Gradient Descent Approaches to Neural-Net-Based Solutions of the Hamilton-Jacobi-Bellman Equation

Remi Munos, Leemon C. Baird and Andrew W. Moore
Robotics Institute and Computer Science Department,
Carnegie Mellon University
5000 Forbes Ave. Pittsburgh, PA 15213, USA.
{munos, leemon, awm}@cs.cmu.edu
http://www.cs.cmu.edu/~AUTON/

Abstract

In this paper we investigate new approaches to dynamic-programming-based optimal control of continuous time-and-space systems. We use neural networks to approximate the solution to the Hamilton-Jacobi-Bellman (HJB) equation which is, in the deterministic case studied here, a first-order, non-linear, partial differential equation. We derive the gradient descent rule for integrating this equation inside the domain, given the conditions on the boundary. We apply this approach to the "Car-on-the-hill" which is a two-dimensional highly non-linear control problem. We discuss the results obtained and point out a low quality of approximation of the value function and of the derived control. We attribute this bad approximation to the fact that the HJB equation has many generalized solutions (i.e. differentiable almost everywhere) other than the value function, and our gradient descent method converges to one among these functions, thus possibly failing to find the correct value function. We illustrate this limitation on a simple one-dimensional control problem.

1 Introduction

In this paper we consider the method of dynamic programming for solving optimal control problems [5; 3] for continuous time-and-space systems. The state dynamics are defined by some controlled differential equation :

$$\frac{dx(t)}{dt} = f(x(t), u(t))$$

where $x(t)$ is the state of the system ($x(t) \in X$, the state space, subset of $\mathbb{R}^d$) and $u(t) \in U$ is the control. Here we assume that the control space U is discrete.

We define the **value function** $V(x)$ as the maximal value (for all possible Lebesgue-measurable control functions $u(t)$) of the *infinite-time discounted gain* :

$$V(x) = \sup_{u(t)} \left\{ \int_0^\tau \gamma^t . r(x(t), u(t)) . dt + \gamma^\tau . R(x(\tau)) \right\}$$

where $r : X \times U \to \mathbb{R}$ and $R : \partial X \to \mathbb{R}$ are the *current* and *boundary reinforcement* functions, γ is the *discount factor* (with $0 < \gamma < 1$) and τ is the *exit time* of the trajectory from the state space (with the convention that $\tau = \infty$ if the trajectory stays infinitely inside X).

From the principle of Dynamic Programming [3], we derive that V satisfies the following **Hamilton-Jacobi-Bellman equation**, which is (for deterministic systems) a first-order non-linear partial differential equation :

- For $x \in X$, if V is differentiable at x, then

$$V(x) \ln \gamma + \max_u [\nabla_x V(x) . f(x, u) + r(x, u)] = 0 \quad (1)$$

- For $x \in \partial X$, we have the following boundary condition :

$$V(x) \geq R(x) \quad (2)$$

These notations are justified in the framework of Reinforcement Learning [7]. More conventional (but equivalent) notations usually consider the minimization of a cost functional [3].

Remark 1 *The boundary condition is an inequality because at some boundary point $x \in \partial X$, there may exist a trajectory going inside X whose gain is strictly greater than the immediate boundary reinforcement $R(x)$.*

We observe that the HJB equation holds only where V is differentiable. However, in general the value function is not differentiable everywhere, even for smooth boundary conditions.

Besides, we can prove that there is an infinity of functions satisfying HJB almost everywhere [3]. So this equation is not easy to solve in the usual sense, because this would lead to either *no classical solution* (i.e. differentiable everywhere) or an *infinity of generalized solutions* (i.e. differentiable almost everywhere).

The formalism of *Viscosity Solutions* [2], which will not be used here, overcomes this problem by introducing a weak formulation of the HJB equation, thus defining an adequate class of solutions : the viscosity solution of the HJB equation (1) with the boundary condition (2) exists, is unique and is the value function of the control problem. See [3] for applications of viscosity solutions to Markov Diffusion Processes and [7] to Reinforcement Learning.

In this paper, we intend to approximate the value function with smooth neural-network-based functions. We use neural networks to approximate the solution V to the Hamilton-Jacobi-Bellman equation. We directly solve this partial differential equation subject to representing $V(x)$ by a neural net.

However, as this will be illustrated by the numerical simulations, the gradient descent method converges to an approximation of some generalized solution (among an infinity) of the HJB equation, thus possibly failing to find the desired value function.

Section 2 presents the architecture of the network and the method used ; *section 3* derives the gradient descent updating rules ; *section 4* proposes a numerical simulation for a highly non-linear two dimensional control problem : the "Car-on-the-Hill" and compares the results to an almost optimal solution obtained with discretization techniques ; and *section 5* illustrates the problems encountered on a simple one dimensional example.

2 The gradient descent method

An approximate V_W neural-net representation of V can be improved by gradient descent on the error :

$$E(W) = \frac{1}{2}\int_x (H(V_W, x))^2 dx \tag{3}$$

where $H(V_W, x)$ is the HJB-Residual :

$$H(V_W, x) \stackrel{def}{=} \begin{cases} V_W(x)\ln\gamma + \max_u[\nabla_x V_W(x).f(x,u) + r(x,u)] & \text{for } x \in X \\ V_W(x) - R(x) & \text{for } x \in \partial X \end{cases}$$

On the t^{th} step, we generate a random state x_t (either on the boundary or inside the state space), and perform a stochastic gradient descent move in the direction that reduces $(H(V_W, x_t))^2$. This gives an unbiased estimate of the gradient, with bounded variance for bounded weights, just as in standard back propagation.

Assume that we use a one-hidden-layer network, with N hidden units (see figure 1) whose output is :

$$V_W(x) = \sum_{j=1}^{N} w_j.\sigma(\sum_{i=0}^{d} w_{ij}.x_i)$$

where the w_{ij}'s $(i = 0..d, j = 1..N)$ are the *input weights*, the w_j's are the *output weights* and x_i $(i = 1..d)$ are the coordinates of x. The w_{0j}'s are the *bias weights* (and we set $x_0 = 1$), and $\sigma(s) = 1/(1+e^{-s})$ is the sigmoid function.

3 Derivation of the updating equations

In order to train the network, we randomly choose states, either inside the state space or on the boundary, and update the weights each time. We deduce the following updating rules :

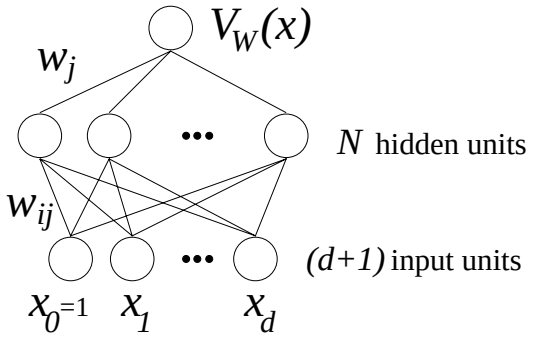

Figure 1: The architecture of the network

3.1 Inside the state space

Here we consider a state $x \in X$.

Let $u^*_W(x)$ be the optimal control with respect to the current approximated function V_W, i.e. $u^*_W(x)$ satisfies :

$$u^*_W(x) = \arg\max_u[\nabla_x V_W(x).f(x,u) + r(x,u)] \tag{4}$$

Since we have a discrete control space, the derivative of the optimal control $u^*_W(x)$ with respect to the weights is zero almost everywhere :

$$\frac{\partial u^*_W(x)}{\partial w} = 0$$

So the gradient rule that applies to a particular weight w (where w means either w_j or w_{ij}) is :

$$\begin{aligned} \Delta w &= -\alpha \frac{\partial\left(\frac{1}{2}H(V_W,x)^2\right)}{\partial w} \\ &= -\alpha\left[\frac{\partial V_W(x)}{\partial w}\ln\gamma + \sum_{k=1}^{d}\left(\frac{\partial^2 V_W(x)}{\partial w \partial x_k}.f_k(x, u^*_W(x))\right)\right]. \\ &\left[V_W(x)\ln\gamma + \sum_{i=1}^{d}\left(\frac{\partial V_W(x)}{\partial x_i}.f_i(x, u^*_W(x))\right) + r(x, u^*_W(x))\right] \end{aligned} \tag{5}$$

By denoting $\sigma_j = \sigma\left(\sum_{k=0}^{d} w_{kj}x_k\right)$, we have the derivative of $V_W(x)$ with respect to the i^{th} coordinates x_i $(i = 1..d)$:

$$\frac{\partial V_W(x)}{\partial x_i} = \sum_{j=1}^{N} w_j.w_{ij}.\sigma_j(1-\sigma_j)$$

Let us derive $\frac{\partial V_W(x)}{\partial w}$:

- For the output weights $w = w_j$, we have :

$$\frac{\partial V_W(x)}{\partial w_j} = \sigma_j$$

- For the input weights $w = w_{ij}$, we have :

$$\frac{\partial V_W(x)}{\partial w_j} = w_j.x_i.\sigma_j(1-\sigma_j)$$

Now, let us derive $\frac{\partial^2 V_W(x)}{\partial w \partial x_k}$:

- For the output weights $w = w_j$, we have :

$$\frac{\partial^2 V_W(x)}{\partial w_j \partial x_k} = w_{kj}.\sigma_j(1-\sigma_j)$$

- For the input weights $w = w_{ij}$, we have :

$$\frac{\partial^2 V_W(x)}{\partial w_{ij} \partial x_k} = \frac{\partial}{\partial w_{ij}}\left[w_j.w_{kj}.\sigma_j(1-\sigma_j)\right]$$
$$= w_j\left[\sigma_j \frac{\partial w_{kj}}{\partial w_{ij}} + w_{kj}\frac{\partial \sigma_j}{\partial w_{ij}} - \sigma_j^2 \frac{\partial w_{kj}}{\partial w_{ij}} - 2w_{kj}\sigma_j \frac{\partial \sigma_j}{\partial w_{ij}}\right]$$
$$= w_j.\sigma_j(1-\sigma_j)\left[w_{kj}.x_i.(1-2\sigma_j) + \begin{cases} 1 \text{ if } i=k \\ 0 \text{ otherwise} \end{cases}\right]$$

3.2 On the boundary

Here we consider a state $x \in \partial X$. From the boundary condition (2), we deduce that the value function satisfies :

$$V(x) = \max\left\{R(x), \frac{-1}{\ln \gamma}\max_u[\nabla_x V(x).f(x,u) + r(x,u)]\right\}$$

So the updating rule will be : for $x \in \partial X$, find $u_W^*(x)$ satisfying (4), then :

- if :

$$\frac{-1}{\ln \gamma}[\nabla_x V_W(x).f(x,u_W^*(x)) + r(x,u_W^*(x))] > R(x)$$

then use the updating rule of the previous section.

- otherwise, use a regular gradient descent on the error $\frac{1}{2}[V_W(x) - R(x)]^2$:

$$\Delta w = -\alpha[V_W(x) - R(x)]\frac{\partial V_W(x)}{\partial w}$$

4 The "Car on the Hill" control problem

We show how this approach applies in the case of the car-on-the-hill problem [6], a highly non-linear, non-minimum-phase plant with a two dimensional state space : the position and velocity of the car. Here, the current reinforcement $r(x,u)$ is zero everywhere. The terminal reinforcement $R(x)$ is -1 if the car exits from the left side of the state-space, and varies linearly between $+1$ and -1 depending on the velocity of the car when it exits from the right side of the state-space. The best reinforcement $+1$ occurs when the car reaches the right boundary with a null velocity (see Figure 2). The control u has only 2 possible values : maximal positive or negative thrust. Because of the discount factor, we are encouraged to get to the goal as quickly as possible.

Figure 3 shows the value function V_W obtained by a neural network with 200 hidden units with a learning rate $\alpha = 10^{-5}$ (the fraction of the training points chosen to be on the boundaries is 0.5).

As a comparison, figure 4 shows the (almost optimal) value function obtained by discretization methods (based

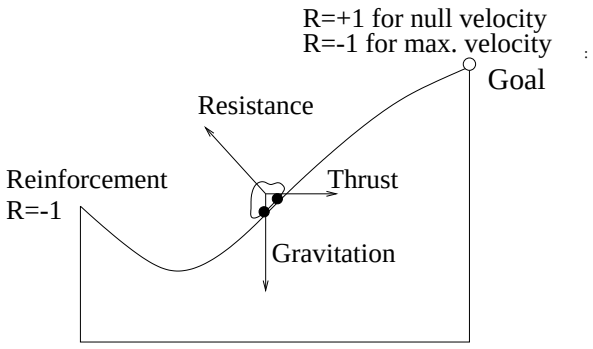

Figure 2: The "Car on the Hill" control problem.

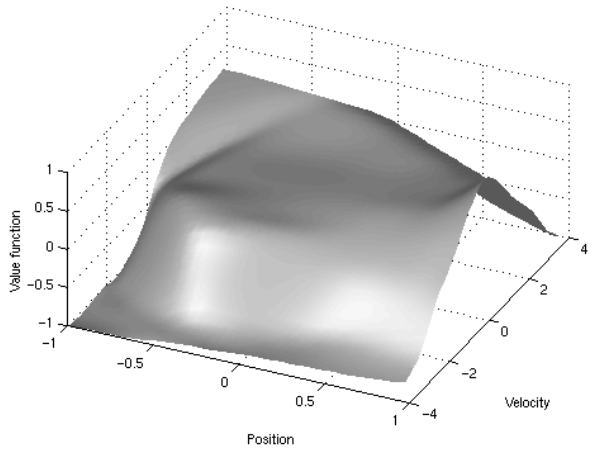

Figure 3: Value function V_W obtained by the neural network.

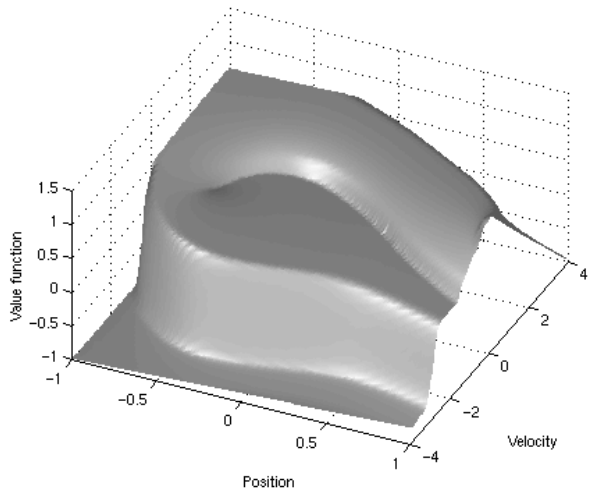

Figure 4: Value function obtained by discretization techniques.

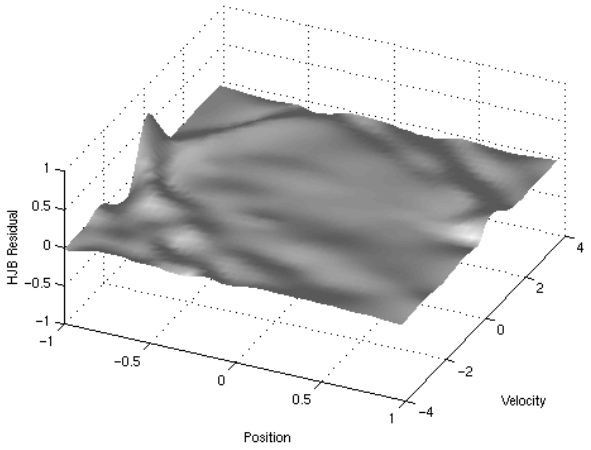

Figure 5: The HJB residual $H(V_W, x)$ of neural network approximated function V_W.

on finite-element methods) using a very refined grid (see [8]).

The HJB-residual $H(V_W, x)$ of the approximated function V_W is plotted in figure 5.

Some sub-optimal trajectories (computed with respect to the neural-net approximated value function, by choosing at each instant the control according to (4)) are plotted in figure 6.

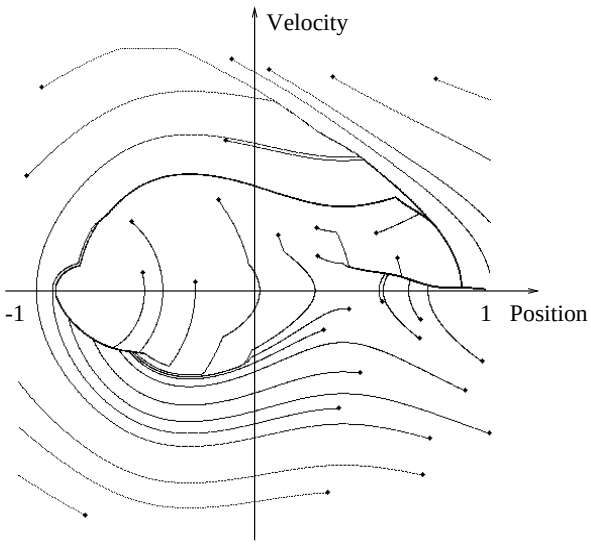

Figure 6: Some trajectories derived by the neural network. The dots are the starting points. At each instant the control is chosen according to (4).

As a comparison, some almost-optimal trajectories (with respect to the fine grid approximated value function) are plotted in figure 7.

During our experiments, we observed that the function approximated by the neural network considerably rely on many parameters : the number of hidden units, the learning rate, the initial weights and the random

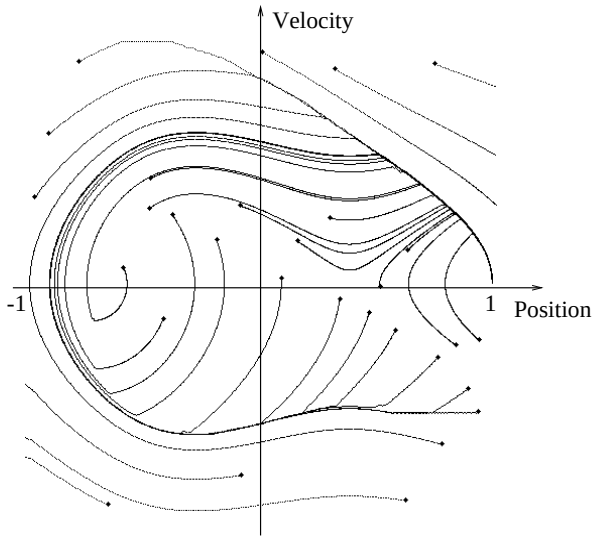

Figure 7: Optimal trajectories derived by the grid.

training set x_t. Small changes in these parameters lead to different functions.

Of course we know that gradient descent methods only ensure local optimality thus these functions are different local optima of the problem of minimizing the error (3).

However, we noticed that many of these functions were very dissimilar but yet had a very low HJB-residual almost everywhere, which means that they minimize the error quite well, even if their general shape differs. This is surprising. And since the optimal control deduced from the value function directly depends on its "shape" (the control is chosen according to V and $\nabla_x V$ in (4)), we observe that the control is very sensitive to the initial conditions and the training data and differs among approximated functions of same error.

We believe that this problem is related to the fact that the HJB equation has an infinity of generalized solutions (i.e. differentiable almost everywhere), and that the gradient descent method converge to one of these generalized functions, thus possibly failing to approximate the desired value function.

Indeed, these generalized functions have a HJB-residual equal to zero almost everywhere, thus are *global* solutions to the problem of minimizing the error (3). Thus each such generalized function may "attract" the gradient descent method to converge towards it, thus failing to approximate the value function.

In the next section, we illustrate this important problem with a very simple one-dimensional control problem.

5 A simple 1d control problem

Let the state $x(t) \in [0, 1]$, the control $u(t) = -1$ or $+1$ and the state dynamics be : $\frac{dx}{dt} = u$.

Consider the current reinforcement :

$$r(x) = \begin{cases} 0 & \text{for } x < 0.4 \\ -\ln \gamma & \text{for } x \in \Omega = [0.4, 0.6] \\ 0 & \text{for } x > 0.6 \end{cases}$$

and a boundary reinforcement defined by $R(0) = 1$ and $R(1) = 1$. In this example, we deduce that the value function satisfies the HJB equation :

$$V(x) \ln \gamma + |V'(x)| + r(x) = 0 \qquad (6)$$

For $x \in \Omega$, there exists a control $u(t)$ such that $x(t)$ stays infinitely inside Ω, leading to a gain of $\int_{t=0}^{\infty} \gamma^t.(-\ln \gamma)dt = 1$.

Thus for $x < 0.2$, the optimal control is $u^* = -1$ which leads to get the boundary reinforcement $R(0) = 1$, whereas for $x \in [0.2, 0.4]$, the optimal control is $u^* = 1$ which leads to the area Ω where the infinite-time discounted reward is also 1.

A similar argument holds for $x \in [0.6, 1]$ and the value function (plotted in figure 8 for $\gamma = 0.5$) of this control problem is :

$$V(x) = \begin{cases} \gamma^x & \text{for } x \leq 0.2 \\ \gamma^{0.4-x} & \text{for } 0.2 \leq x \leq 0.4 \\ 1 & \text{for } x \in \Omega \\ \gamma^{x-0.6} & \text{for } 0.6 \leq x \leq 0.8 \\ \gamma^{1-x} & \text{for } x \geq 0.8 \end{cases}$$

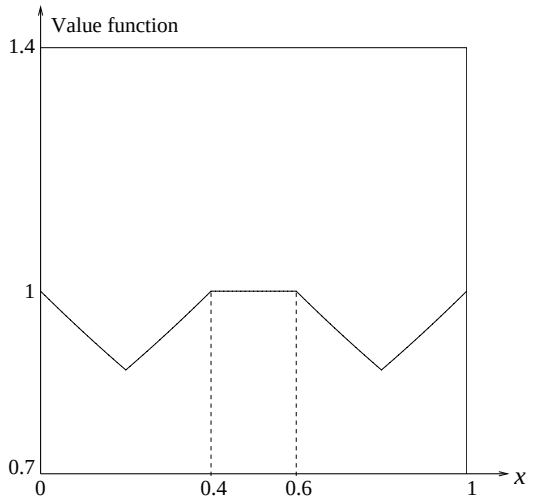

Figure 8: The value function V.

We verify that $V(x)$ satisfies the HJB equation (6) everywhere except at $x = 0.2$, $x = 0.4$, $x = 0.6$ and $x = 0.8$ where it is not differentiable.

We already claimed that the HJB equations have many generalized solutions (see [3; 7]), so we provide an example of such a generalized solution V_g (plotted in figure 9) :

$$V_g(x) = \begin{cases} \gamma^{-x} & \text{for } x \leq 0.4 \\ 1 + (1 - \gamma^{0.4})\gamma^{-x} & \text{for } 0.4 \leq x \leq 0.5 \\ 1 + (1 - \gamma^{0.4})\gamma^{x-1} & \text{for } 0.5 \leq x \leq 0.6 \\ \gamma^{x-1} & \text{for } x \geq 0.6 \end{cases}$$

which satisfies the HJB equation (6) everywhere except at $x = 0.4$, $x = 0.5$ and $x = 0.6$ where it is not differentiable.

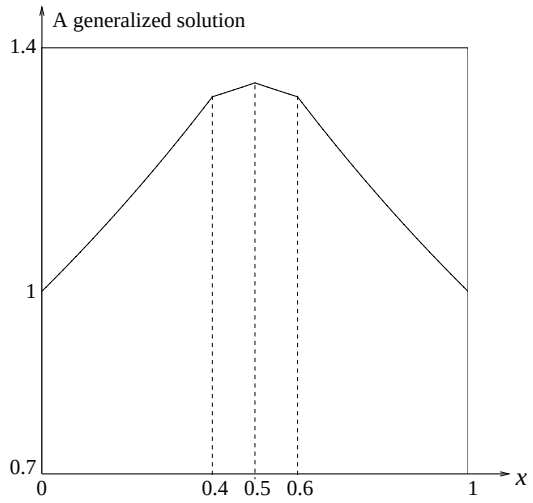

Figure 9: A generalized solution V_g to HJB (6).

Figure 10 shows the function V_W computed by the neural network (with 100 hidden units and a learning rate of 10^{-5}) trained by our algorithm of gradient descent on the error (3). **We observe that the function learned by the network completely differs from the value function** V (shown in figure 8). Actually, the function learned by the network converges to the generalized solution V_g shown in figure 9.

Moreover, the control derived by this function (here $u^*_W(x) = \text{sign}(V'_W(x))$) is different from the optimal control (derived by V) (for $x \in [0, 0.2]$ and $x \in [0.8, 1]$).

We notice that the HJB-residual (plotted in figure 11) of this function is very low, which means that the gradient descent method worked well and the error (3) is almost minimized.

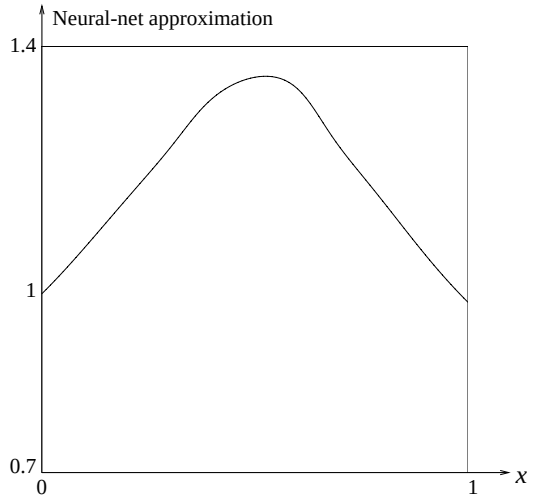

Figure 10: A solution V_W learned by the network.

Knowing that the HJB-residual is zero almost everywhere for both the value function V and the general-

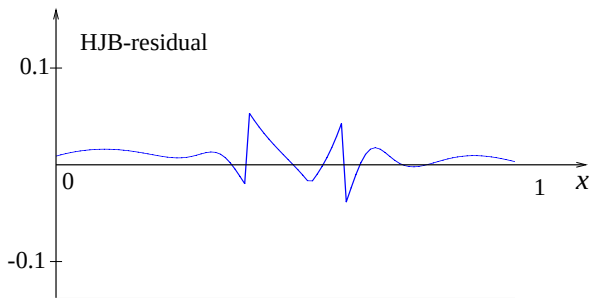

Figure 11: The HJB-residual of the net function V_W.

ized solution V_g, in this case, the network tends to approximate V_g instead of V because its global shape is smoother. Indeed, in our experiments, none of our simulations converged to the expected value function, even when we increased the number of hidden units.

This simple example illustrates the fact (stated theoretically in [7]) that gradient-descent methods may approximate any (among a possible infinity of) generalized solution of the HJB equation, because these functions are global optimum of the problem of minimizing the error (3) (i.e. the HJB-residual of generalized solutions is zero almost everywhere thus these functions have an error of zero).

The insight gained in studying this simple problem provides us with a better understanding of the cause of the bad accuracy of the value function and the policy, as well as the very sensitivity to initial conditions observed for the simulations on the "Car on the Hill".

6 Conclusion

The algorithm described in this paper combines the gradient descent approach of [1] with the HJB equations from conventional optimal control. This combination allows us to use function approximators such as neural networks to represent the value function for continuous-time problems.

This permits a reinforcement learning algorithm that had previously been restricted to discrete time to now be applicable in continuous time.

However, this method suffers from a major limitation because the gradient-descent might lead to approximate any generalized solutions of the HJB equation, thus failing to find the value function. Moreover the control induced by such functions might be very different from the optimal control (derived from the value function). The cause of these troubles is that there exists many global optima (i.e. the generalized solutions of HJB) to the problem of minimizing the error (3).

Nevertheless, this approach may be extended and more successfully applied to approximate the solution of second-order HJB equations that arise when the dynamics are stochastic. Indeed, in the particular case of uniform parabolicity (i.e. when there is some stochasticity in every direction), we know that the value function is smooth (see [4]) thus we can integrate the HJB equation in the usual sense and there is a unique solution to the problem of minimizing the error (3), which can be approximated by the gradient-descent method described in this paper.

References

[1] L. C. Baird. Residual Algorithms: Reinforcement Learning with Function Approximation. In *Machine Learning: Proceedings of the Twelfth International Conference*. Morgan Kaufman, 1995.

[2] M.G. Crandall, Hitoshi Ishii, and P.L. Lions. User's guide to viscosity solutions of second order partial differential equations. *Bulletin of the American Mathematical Society*, 27(1), 1992.

[3] Wendell H. Fleming and H. Mete Soner. *Controlled Markov Processes and Viscosity Solutions*. Applications of Mathematics. Springer-Verlag, 1993.

[4] N.V. Krylov. *Controlled Diffusion Processes*. Springer-Verlag, New York, 1980.

[5] H. Kushner and D. Clark. *Stochastic Approximation Methods for Constrained and Unconstrained Systems*. Springer-Verlag, 1978.

[6] Andrew W. Moore. Variable resolution dynamic programming: Efficiently learning action maps in multivariate real-valued state-spaces. *Machine Learning : Poceedings of the Eight International Workshop*, pages 333–337, 1991.

[7] Rémi Munos. A study of reinforcement learning in the continuous case by the means of viscosity solutions. *To appear in Machine Learning Journal*, 1999.

[8] Rémi Munos and Andrew Moore. Variable resolution discretization for high-accuracy solutions of optimal control problems. *International Joint Conference on Artificial Intelligence*, 1999.