DREAMING MACHINE LEARNING: QUASI-DISTANCES AND LIPSCHITZ EXTENSIONS FOR MODELING FINANCIAL PROCESSES (WORK IN PROGRESS)

J.M. CALABUIG, H. FALCIANI AND E.A. SÁNCHEZ-PÉREZ

Abstract. We present a concrete application of Lipschitz type extension of reward functions defined on metric spaces. Using some known states of a dynamical system representing the evolution of a financial market, we use our technique to create new ones. This provides an improvement of the investing strategy in the market using less information, giving a new technique for reinforcement learning.

1. Introduction and basic definitions

The aim of this paper is to show a new enviroment for the development of new mathematical tools for reinforcement learning. We model a dynamical system as a sequence of vectors of a finite dimensional space, and characterize certain reward function for a known subset in it. Using well-known theoretical techniques of extension of Lipschitz functions on metric spaces, we construct a framework for understanding and computing improved reward functions in the context of the reinforcement learning.

In this paper we are interested in finding suitable extensions of semi-Lipschitz maps on quasi-metric graphs, with the aim of showing a new method of reinforcement learning for artificial intelligence. Our arguments bring together ideas from abstract topology on quasi-metrics and semi-Lipschitz maps and practical computational tools for extending Lipschitz functions on metric vector spaces. Thus, we use the McShane and the Whitney extension formulas for Lipschitz maps in order to extend reward functions in a particular way. Although our approach is new, the reader can find some related ideas in [1, 2, 5].

We center our attention in a concrete problem: *given a sequence of states of a financial parket and a reward function acting in it, we are interested in extending the reward in a meaningful way for providing a improved tool for decision making.* This allows to mix original known situations with new created states (*dreamed states*), we produce a typical reinforcement learning procedure. Computations are easy, since the extension formulas are simple, so the technique would be applied when a big ammount of data are involved.

Let us present some relevant concepts. A quasi-pseudo-metric on a set M is a function $d : M \times M \to \mathbb{R}^+$ —the set of non-negative real numbers— such that

Date: May 17, 2018.

Key words and phrases. Quasi-pseudo-metric, machine learning.

(1) $d(a,b) = 0$ if $a = b$, and
(2) $d(a,b) \leq d(a,c) + d(c,b)$

for $a, b, c \in M$. A topology is defined by such a function d: the open balls define the basis of neighborhoods. For $\varepsilon > 0$, we define the ball of radius ε and center in $a \in M$ as

$$B_\varepsilon(a) := \Big\{ b \in M : d(a,b) < \varepsilon \Big\}.$$

(M, d) is called a quasi-pseudo-metric space. We will work in this note with metrics, that is $d(a,b) = 0$ if and only if $a = b$, and $d(a,b) = d(b,a)$ for all $a, b \in M$. In this case, the defined topology is Hausdorff. The same technique can be applied in the general quasi-pseudo-metric case, although this property would not be satisfied.

Let us recall now some notions regarding functions. Let (M, d) be a metric space. A function $f : M \to \mathcal{R}$ is a Lipschitz function if there is a positive constant K such that

$$|f(a) - f(b)| \leq K d(a,b), \quad a, b \in M.$$

The infimum of such constants K is called the Lipschitz constant of f.

Regarding extensions of Lipschitz maps, the McShane-Whitney theorem states that if M_0 is a subset of a metric space (M, ρ) and $T : M_0 \to \mathbb{R}$ is a Lipschitz function with Lipschitz constant K, there always exists a Lipschitz function $\tilde{T} : M \to \mathbb{R}$ extending T and with the same Lipschitz constant. The function

$$\tilde{T}(x) := \sup_{u \in B} \{ T(u) - K\, \rho(x,u) \}, \quad x \in M,$$

provides such an extension; it is sometimes called the McShane extension. We will use it for giving a constructive tool for our approximation. The Whitney formula, given by

$$T^W(x) := \inf_{u \in B} \{ T(u) + K\, \rho(x,u) \}, \quad x \in M,$$

provides also an extension. We will use the first one in this paper. The reader can find some information about the theoretical foundations in [3, 4].

2. Metric spaces of states and Lipschitz maps: an algorithm for machine learning

We will work with the following metric space as model of a dynamical system given by a financial market. Consider a subset M_0 of vectors of a finite dimensional linear space M. We define a metric defined by mixing the angular pseudo-distance (geodesic distance) and the Euclidean norm in this space. Thus, if

$$Cos(s_i, s_j) = \frac{s_i \cdot s_j}{\|s_i\|\, \|s_j\|}, \quad s_i, s_j \in M,$$

we define a distance by mixing the angle

$$\Theta(s_i, s_j) = \frac{ArcCos\big(\frac{s_i \cdot s_j}{\|s_i\|\, \|s_j\|}\big)}{\pi},$$

and an Euclidean component

$$E(s_i, s_j) = \|s_j - s_i\|_2 = \sqrt{\sum_{k=1}^{4} |s_i^k - s_j^k|^2}.$$

We define the distance in our space as

$$d_\epsilon(s_i, s_j) = \Theta(s_i, s_j) + \epsilon E(s_j, s_i). \tag{2.1}$$

We can use for defining the distance d for example $\epsilon = 1/10$, that is, we use

$$d(s_i, s_j) = d_{1/10}(s_i, s_j) = \Theta(s_i, s_j) + \frac{1}{10} E(s_j, s_i).$$

We consider (M_0, d) as a subspace of a bigger metric space (M, d), where M is the finite dimensional space fixed at the beginning.

We will consider also a reward function R acting in M_0, and we want to extend it to the whole M. It is supposed to measure "how succesful" is a given state. We will use the MacShane extension formula. We have to compute the Lipschitz constant K for the reward function R in order to get the extension $\hat{R}$, for which the same K works.

Finally, we will use $\hat{R}$ for simulating new time sequences. In order to do this, we generate randomly new states for increasing the set M_0. We create in this way a new seminal set M_1 bigger than M_0, in which we are mixing "known situations" ($s \in M_0$) and new ones, ("dreams", $s^* \in M_1 \setminus M_0$.) The rate of elections of known cases and dreams that we have chosen is $\beta = 50\%$.

We use this procedure for defining a new time series. We can already compare it with the known one. The final proof is the next step.

3. Training and dreaming: a Lipschitz approximation to a real market reward function

Suppose that we are analyzing four markets for the same product. We have the complete behavior of the values of all of them each minute of a day. For the aim of simplicity, we assume that at $t = 0$ the value in all the markets equals 0.

1. A state of the system is given by a five coordinate vector: each minute the vector gives the cumulative increase or decrease of the value in each market; the last coordinate corresponds to the value when nothing is invested.

   We consider series of "bets" applied at each minute: they are described as the % of the money that the decision maker wants to apply in each market this minute (including not investing a certain part). It is supposed he is investing 100 money units at each step. A bet is then given by a five positive coordinate vector summing 100; recall we have five coordinates since the decision maker could decide not to invest a part of the money this time.

2. The reward function is then defined as a two variable function given by the scalar product of the state and the

   $$R(a, s) = a \cdot s^t,$$

   where s^t is the transpose of the vector s.

   We can use all the information for similar situations for computing a reward function depending only on the state, that will be the one which

will be used. This is relevant, since we are going to evaluate the state of the system using this reward function.

In order to define it, we use the following procedure. For a state of the system s, we use the distance defined by Formula (2.1) for $d = 1/10$ for obtaining similar states that have been already checked, and for which we have obtained the reward function. We define then

$$R(s) := mean\, R(a, s),$$

where the mean is computed over two sets, in a relation of 90% and 10%, respectively.

a) The first set —90%— is defined by using bets a applied for states that are similar to s with respect to the distance $d_{1/10}$ that have obtained a good enough values of the reward.

b) The second one —10%— is randomly obtained.

3. We perform in this way a method for obtaining a reward function. We use the first 50% positions of the market (Figure 1) to train the way we define the adequate values of the elements of the system —R, a...—. The complementary 50% is used for checking the behavior of the model (Figure 2).

FIGURE 1. Behavior of the set of states for training the model.

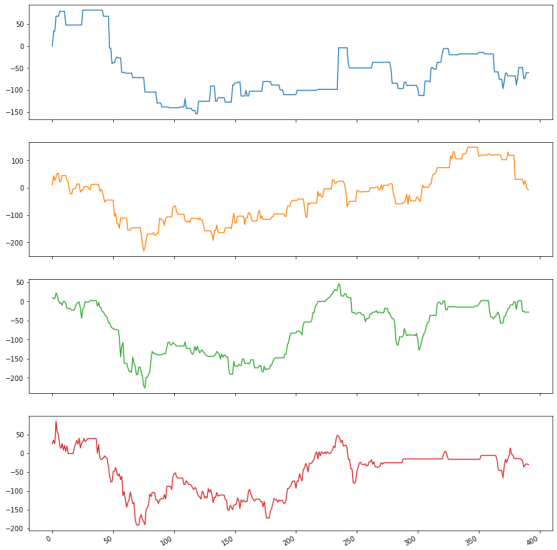


FIGURE 2. Behavior of the set of states for testing the model.

4. Next we sort a number of bets that have already been improved using the observations of the market shown in the previous figures. In Figure 3 and Figure 4 a sequence of bets that have been sorted/optimized as explained before. At each time, the sum of the values in the five graphics sum 100%.

FIGURE 3. Sequence of real states for the construction of the reward function.

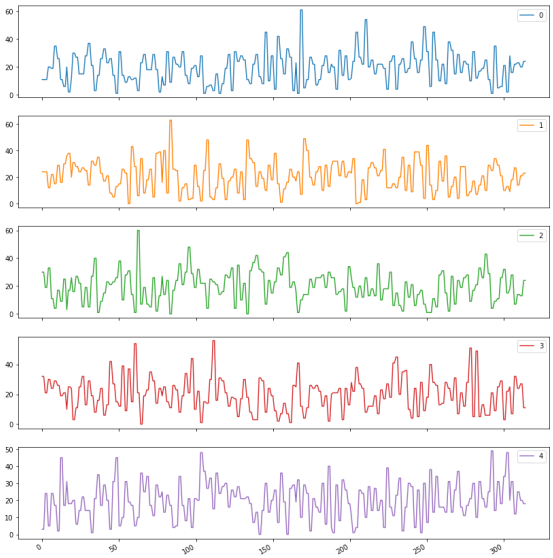
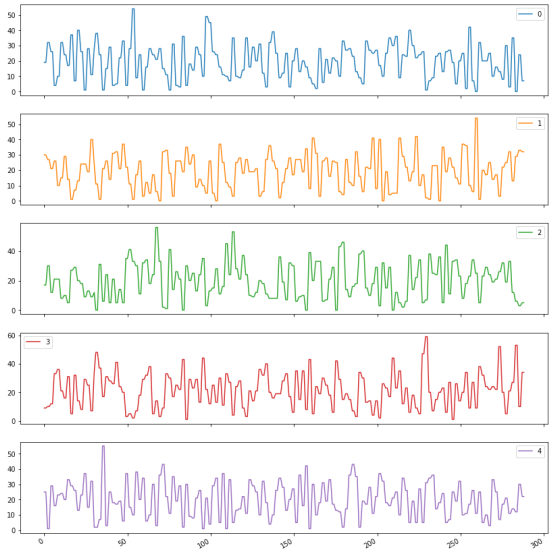


FIGURE 4. Sequence of states including dreamed states for the construction of the reward function.



In the first one (Figure 3), the bets are obtained by using the reward function. They are given by particular five dimensional vectors which satisfy that the reward function takes the value that have been obtained as explained in step 2. In this case, all the bets are associated to values of the reward function that have been computed using real (observed) values of the market, as in Figure 1.

The second one (Figure 4) represents the bets that have been obtained by mixing "real states" an "dreamed states". The "dreamed states" are obtained by using the McShane extension $\hat{R}$ of the reward function R. Both type of states are sorted each time with a probability of $\beta = 50\%$.

5. Finally, we check the success of the model when 50% of the information about the reward function is obtained by the McShane extension $\hat{R}$ of the computed reward function R. In the terminology that we have introduced in the paper, we have 50% of real states and 50% of dreamed states.

   We assume that we start betting in the market at time $t = 0$ with 1000 of money units and we stop when we loose all of them. In order to check the success of the model, we produce a simulation when the reward function is purely obtained by the information of the market (Figure 3), and using 50% of dreamed states (Figure 4). The reader can notice the the success is similar in both cases. That is, the same result can be obtained by using the McShane extension of the 50% of known date instead of 100% of real data.

FIGURE 5. Simulation with real data obtained from the experience.

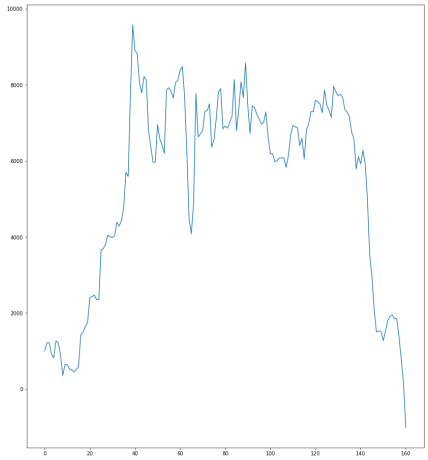
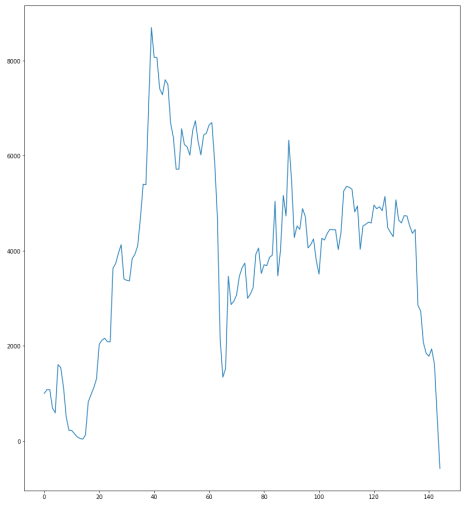


FIGURE 6. Simulation with 50% of real data +50% of dream.



REFERENCES

[1] Asadi, K., Dipendra, M., and M.L. Littman. "Lipschitz Continuity in Model-based Reinforcement Learning." arXiv preprint arXiv:1804.07193 (2018).

[2] Kurt Driessens, Jan Ramon, Thomas Gärtner Graph kernels and Gaussian processes for relational reinforcement learning, Mach Learn (2006) 64:91119 DOI 10.1007/s10994-006-8258-y

[3] Mustata, Costica. "Extensions of semi-Lipschitz functions on quasi-metric spaces." Rev. Anal. Numr. Thor. Approx. 30.1 (2001): 61-67.

[4] Mustata, Costica. "On the extremal semi-Lipschitz functions." Rev. Anal. Numr. Thor. Approx. 31.1 (2002): 103-108.

[5] N'Guyen, S., Moulin-Frier, C., and Droulez, J. (2013). Decision making under uncertainty: a quasimetric approach. PloS one, 8(12), e83411.