Reinforcement learning chaos control using value sensitive vector-quantization

Sabino Gadaleta and Gerhard Dangelmayr
Colorado State University, Dept. of Mathematics
101 Weber Building, Ft. Collins, CO 80523
{sabino,gerhard}@math.colostate.edu

Abstract

A novel algorithm for the control of complex dynamical systems is introduced that extends a previously introduced approach to chaos control (S. Gadaleta and G. Dangelmayr, Chaos, 9, 775-788, 1999) by combining reinforcement learning with a modified version of the growing neural-gas vector-quantization method to approximate optimal control policies. The algorithm places codebook vectors in regions of extreme reinforcement learning values and produces a codebook suitable for efficient solution of the desired control problem.

1 Introduction

Many physical systems exhibit dynamics in certain parameter regimes, where they display a rich variety of different behaviours, including chaotic dynamics. In principle, a chaotic system has access to an unlimited number of states which are, however, unstable and are visited by the system in an unpredictable manner. In many cases of interest, system performance can be improved by controlling the dynamics such that it is constrained to one of these unstable states (see [2] for an overview on chaos control). For an efficient control, model-independent techniques are needed which enforce the system to reside in this state through *small* perturbations and allow fast control from any initial state of the system. Local chaos control techniques [14] combined with targeting methods [11] can satisfy these requirements. In [8] we introduced a flexible chaos control method based on reinforcement learning (RL) which represents a simple algorithm and allowed the control of chaotic systems from any initial state. It is interesting to note that chaos control techniques have been applied to the control of other types of dynamical behaviours such as noisy nonlinear dynamics [3] and multistable behaviour [10], which further motivates to study efficient chaos control techniques.

From a general viewpoint, the control of chaotic systems can be viewed as the problem of stabilization of an unstable fixed point of a nonlinear map. The chaos control problem can be formulated as an optimal control problem and solved through optimization techniques or dynamic programming [15]. Reinforcement learning has been shown to efficiently solve optimal control problems [1] and can be used to control chaotic systems. Der and Herrmann [4] were the first to apply reinforcement learning to the control of a simple chaotic system and introduced the idea to apply reinforcement learning in an approximate discrete state space represented by a codebook computed through a vector quantization (VQ) technique. We generalized this idea and demonstrated control of a variety of chaotic systems [8] and a simple multistable system [10].

In this paper we improve upon the previous algorithm by (1) combining the codebook approximation phase with the reinforcement learning control policy approximation phase, and (2) allowing for state-dependent control sets. In particular we use a modification of Fritzke's growing neural-gas VQ (GNGVQ), as described below, which allows to place new codebooks in regions minimizing or maximizing a certain performance measure. Globally, far from the desired state, we allow for a minimal set of discrete control perturbations, while locally, close to the desired state, the controller can choose from a large set of fine perturbations.

The idea of combining a growing VQ technique with reinforcement learning, is, to our knowledge, new and promising for other "established" reinforcement learning control problems, such as the teaching of roboter arms to move into a well defined final state. In a typical application, the controller must learn to position the arm into a desired final state as quickly as possible from any initial position. Allowing globally coarse, locally fine controls is equivalent to allowing globally only for very abrupt, discontinuous arm movements.

Only when the arm is close to the desired position, the controller will allow for fine control signals to smoothly move the arm into the desired position. This application of a minimal set of discrete controls reduces the reinforcement learning problem since only a minimal set of states has to be explored.

2 RL chaos control through VQ

Many nonlinear systems of interest can be described by a nonlinear system of first-order differential equations $\mathbf{x}' = \mathbf{F}(\mathbf{x})$. It is standard in dynamical system theory to view the continuous system as a discrete map $\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n)$, where the map $\mathbf{f}$ is typically constructed by finite sampling of the trajectory when it intersects a surface-of-section. We assume then in the following that the dynamic system is described by a discrete map with state space X,

$$\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n, u_n), \quad \mathbf{x} \in X, \tag{1}$$

but we do not assume any available analytic knowledge about $\mathbf{f}$. The parameter $u_n = u_0 + \Delta u_n$ represents a control parameter which is chosen such the unperturbed system ($\Delta u_n = 0$) shows chaotic dynamics. Assuming discrete dynamics described by (1), chaos control can be interpreted as stabilizing the dynamics at a fixed point of order p, where p is typically small, through small perturbations Δu (perturbations are small if they lead to only small changes in system dynamics).

The approach to chaos control through RL as discussed in [8] and in detail in [7] can be summarized as in Fig. 1. A vector quantization technique such as the neural-gas algorithm [13] is initially applied to construct a set of codebook vectors $\mathbf{w} \in W$ which approximates the state space X. Given a codebook W, a state space X is approximated by the nearest neighbor $\mathbf{w}(\mathbf{x})$ in W:

$$\mathbf{w}(\mathbf{x}) = \arg\min_{w \in W} ||\mathbf{x} - \mathbf{w}||.$$

To solve the control problem with RL, we associate to each state $\mathbf{w}$ a control set $U(\mathbf{w})$ and choose controls $\Delta u_n \in U(\mathbf{w}(\mathbf{x}_n))$ according to a control policy defined through an action-state value function $Q(\mathbf{w}, u)$. Based on instantaneous rewards r_n received from the RL critic for performing action u_n when the system is in state state $\mathbf{w}(\mathbf{x}_n)$, the action-value function is updated according to the Q-learning update rule [16]

$$\Delta Q(\mathbf{w}_{n-1}, u_{n-1}) = \beta[r_n + \gamma \max_u Q(\mathbf{w}_n, u) - Q(\mathbf{w}_{n-1}, u_{n-1})].$$

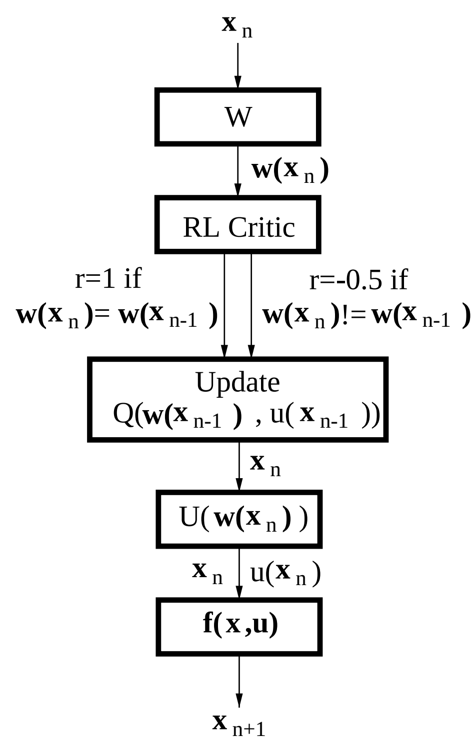


Figure 1: Summary of reinforcement learning chaos control through vector quantization for the control of a period one fixed point.

Although this update rule produces an optimal value function Q^* only under non-realistic operating conditions (infinite exploration of state-action space), the value function will be close to optimal provided that the state-action space (W, U) is explored sufficiently. It follows that the learning time required for the approximation of a good value function depends on the sizes $|W|, |U|$ of the codebook and control set, respectively. In the previous approach [8] the size of the codebook was initially fixed and the codebook was approximated before the policy approximation was initiated. Furthermore the control set was fixed to $|U(\mathbf{w})| = 3, \forall \mathbf{w} \in W$.

Separating state-space discretization and control policy approximation does, however, in general not lead to an optimal codebook for the particular control problem. Specifically the state-space approximation might be too fine, which results in a slow controller, or too coarse, which can result in suboptimal control performance. More generally, a codebook is desired which approximates the action-state space (X, U) and not only the state space X. To this end we develop a vector-quantization technique, based on Fritzke's growing neural-gas algorithm [6], which places clusters in regions of high and low value for the given control problem. In particular, new clusters are placed with nonzero probability into regions of high accumulated instantaneous reward. While globally approximating the state-action space, the resulting codebook approximates regions of high accumulated instantaneous reward to higher accuracy, thus producing a codebook better suitable for the current control problem. In addition, by associating control sets of larger size to codebook vectors placed in regions of high value, the controller has the ability to choose fine tuned signals allowing chaos control through minimal perturbations.

3 Fritzke's growing neural-gas VQ

Many VQ techniques require the size of the codebook as an input algorithm. Growing VQ algorithms initiate the codebook with a minimal number of vectors and increase the size of the codebook by placing new codebooks in regions which minimize or maximize a certain local performance measure. The growing neural-gas algorithm of Fritzke [6] combines a growing cell structure algorithm [5] with competitive Hebbian learning [12]. Each codebook vector accumulates the local distortion error and new units are placed in areas which possess the currently largest accumulated distortion error. In this formulation the algorithm is sensitive to distortion error. The GNGVQ updates a connectivity matrix which approximates the topological structure of the input space and uses this topology to update its codebook vectors. At each iteration step a new input vector $\mathbf{x}$ is presented and the closest reference vector $\mathbf{w}_1$ is updated according to the Hebb rule $\Delta\mathbf{w}_1 = \eta_1(\mathbf{x} - \mathbf{w}_1)$. In addition to the closest reference vector all topological neighbors are updated according to a Hebb rule with an update factor $\eta < \eta_1$. For the specific implementation of the algorithm see [6, 7]. See Fig. 2 for a GNGVQ of the Henon attractor.

The interesting feature of the algorithm is that it allows growing codebooks, can be applied in on-line learning, and can easily be modified to approximate any desired feature of an input space. For our purposes we desire an algorithm which is sensitive to reinforcement learning value of a region in addition to distortion error.

4 RL chaos control through value sensitive VQ

The modification of the GNGVQ for the purpose of chaos control is straightforward. First we need a reward function which rewards controls leading to states in the neighborhood of fixed points of desired period. To this end we introduce the quantity

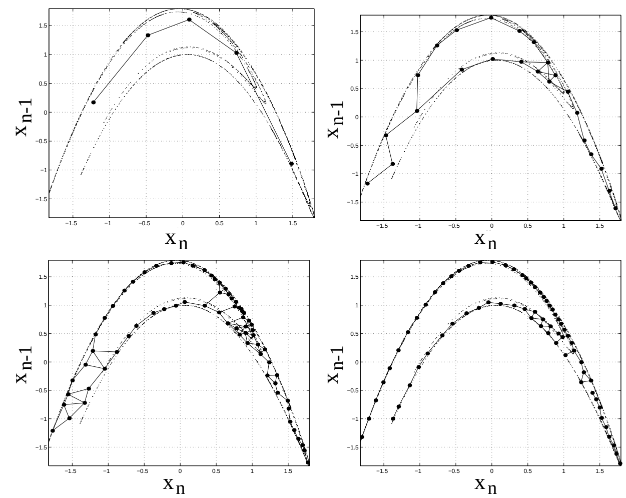


Figure 2: GNGVQ of the Henon attractor. The large filled dots represent the codebook vectors.

$$d_p(\mathbf{x}_n) = \frac{\|\mathbf{x}_n - \mathbf{x}_{n-p}\|}{\exp\left(\sum_{i=1}^{p-1} \ln \|\mathbf{x}_n - \mathbf{x}_{n-i}\|\right)},$$

which is minimal only for fixed points of period p. Fig. 3 shows d_2 for the chaotic logistic map $x_{n+1} = 3.8x_n(1 - x_n)$.

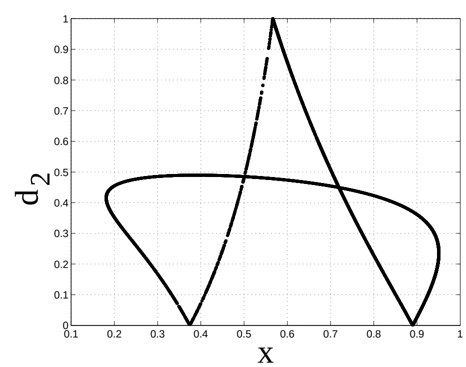


Figure 3: The quantity d_2 for the logistic map.

Based on d_p we introduce the following reward function

$$r_n = -d_p(\mathbf{x}_n) - |u_n| + |u_{min}| + \delta, \quad (2)$$

which allows stabilization of fixed points of period p. The first term $-d_p(\mathbf{x}_n)$ is minimal for a state $\mathbf{x}_n$ close to a fixed point of period p. The two terms $-|u_n| + |u_{min}|$ are introduced to reward minimal action signals, and u_{min} denotes the smallest accessible nonzero control signal. The term δ is added to shift the reward function slightly to positive values for minimal successful control signals. For control of the logistic map, as demonstrated below, we use $\delta = 0.01$.

Based on the reward function (2) we now illustrate value sensitive growing neural-gas (VSGNG) chaos control through the example of on-line control of the logistic map $x_{n+1} = u_0 x_n(1 - x_n)$, with $u_0 = 3.8$. In controlled form the logistic map can be written

$$x_{n+1} = (u_0 + u_n)x_n(1 - x_n),$$

with the control signal u_n chosen from the control set $U(w(x_n))$ greedy from the value function $Q(w(x_n), u)$:

$$u_n = \arg \max_{u \in U(w(x_n))} Q(w(x_n), u).$$

Initially we start iteration from a random state $x_0 \in (0, 1)$ and initialize the codebook with two random states $w_1, w_2 \in (0, 1)$. To each state action pair we associate a state-action value function $Q(w, u)$ with $u \in U(w)$ and initially zero values. To estimate the average RL value of a state, we introduce the quantity $Q^v(w)$ which we update according to

$$Q^v_n(w) = \frac{(n(Q^v(w)) - 1)\, Q^v_{n-1}(w) + r_n}{n(Q^v(w))},$$

where $n(Q^v(w))$ is the number of times the codebook vector w has been winner until iteration n. Using $Q^v(w)$ we can modify the GNG algorithm to obtain a VSGNG chaos control algorithm. We initiate control with a minimal codebook consisting of two reference vectors. Given state x_n, its reference state $w_n = w(x_n)$ is determined from the current codebook. Then a control signal u_n is chosen from the control set $U(w_n)$ and a new state x_{n+1} is obtained. Given the state x_{n+1}, past states, and u_n the reward r_{n+1} can be computed and the values $Q(w_n, u_n)$ and $Q^v(w_n)$ can be updated. This is similar to the previous algorithm. New is now that the codebook is changing. Unless a stopping criterion is reached, every λ iterations (here $\lambda = 200$) the algorithm will insert a new codebook vector in certain regions of state space. In which region the new codebook vector is inserted depends on the maximum average value $Q^v_{max} = \arg\max_w Q^v(w)$. If Q^v_{max} is below a certain threshold (here we used $Q^v_{max} < -0.1$) then units will be inserted as in the original GNGVQ in regions of maximal distortion error. If on the other hand $Q^v_{max} > 0$ units will be inserted in between the reference vector corresponding to Q^v_{max} and its euclidean neighbor, or in other words in regions of maximum RL value. For $-0.1 \leq Q^v_{max} \leq 0$ units will be placed with 50% probability in regions of maximum RL value and with 50% probability in regions of minimum RL value (in between the unit with minimum Q^v and its euclidean neighbor). The algorithm will initially update the codebook like the original GNG algorithm. At each iteration the value function is updated and at a certain iteration the pure spatial codebook will be sufficient to allow the receiving of positive rewards leading eventually to $Q^v_{max} > -0.1$.

To reduce the complexity of the learning problem we allow the size of the set $U(w)$ to vary with w. Let $n_w = (|U(w)| - 1)/2$ and consider the finite control set

$$U^{n_w}_{u_{max}}(w) = \left\{0,\ \pm\frac{u_{max}}{n},\ \pm\frac{2u_{max}}{n},\ \cdots\ ,\ \pm u_{max}\right\}.$$

In this notation, the control set

$$U(w) = \{0, \pm 0.02, \pm 0.04, \pm 0.06, \pm 0.08, \pm 0.1\},$$

would be denoted by $U^5_{0.1}(w)$. For the control of the logistic map, initially every codebook vector and every new inserted codebook vector w has associated the minimal control set $U^1_{0.1}$. However, if a codebook vector is placed in a region of maximum RL value, then we associate a large control set $U^{10}_{0.1}$ to this reference vector. This allows for a larger number of smaller control signals in regions close to the desired state, while allowing only for a minimal set of controls globally, and reduces the complexity of the RL learning problem.

One question must still be answered. How do we decide when to stop the updating and growing of the codebook vectors? Unless we stop this process, the codebook continues changing and the RL algorithm will not be able to converge onto a final state-action value function. As described above, if the clustering is adequate to allow for positive rewards, eventually all new units will be placed in the region in state space of highest average reward. Then, from a histogramm plot of the reference vectors a clear peak will appear in these locations. At this time we can stop the codebook generation phase. After the codebook is fixed, we can continue pure RL on the fixed codebook until the desired state is stabilized.

Fig. 4 shows the codebook resulting from control of the fixed point $x_f \approx 0.7368$ and we see the clear peak in the neighborhood of the fixed point.

We present results for control of the logistic map in the next section.

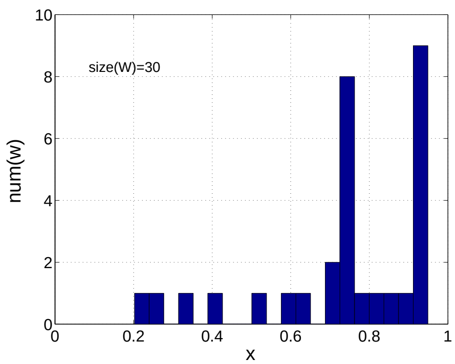


Figure 4: Codebook for the control of the fixed point of the logistic map.

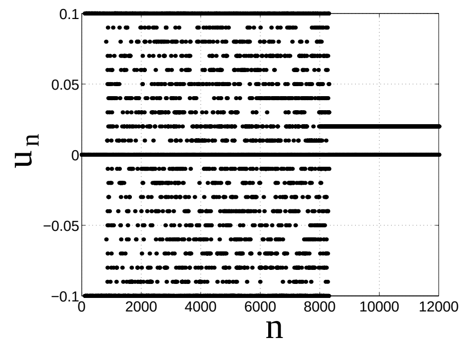
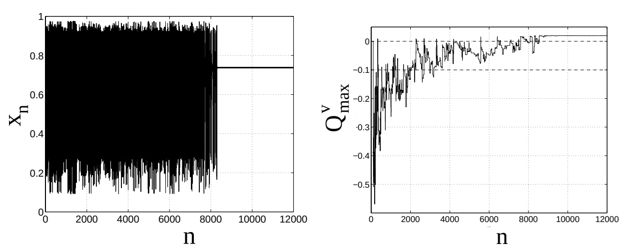


Figure 6: Applied control signals u_n during the online learning process for control of the period one fixed point of the logistic map.

5 Results

Fig. 5a) shows the online control of the fixed point of the logistic map through the VSGNG algorithm as discussed in the last section. Fig. 5b) shows Q^V_{max}. After approximately 6,000 iterations the codebook generation was stopped on the basis of the histogram of Fig. 4. Control was established after about 8,000 iterations.

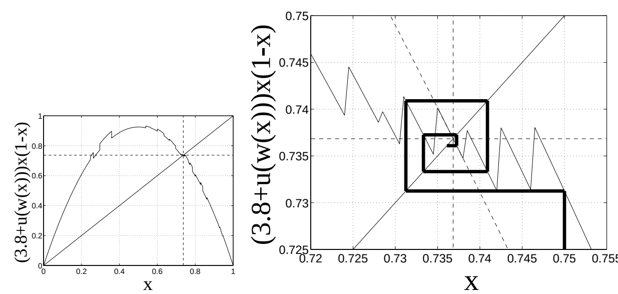


a) b)

Figure 5: a) Online control of the fixed point of the logistic map through the VSGNG algorithm. b) Q^V_{max}.

Fig. 6 shows the applied control signals u_n during the learning process. Initially controls from the set $U^1_{0.1}$ are applied, until after approximately 800 iterations, when the first unit is placed into a valuable neighborhood. Then controls from both $U^1_{0.1}$ and $U^{10}_{0.1}$ appear. Stabilization of the fixed point is established through an almost minimal control perturbation $u_n = 0.02$.

To visualize the approximated control policy, Fig. 7 shows the map $(x_n + u_n(w(x_n)))x_n(1 - x_n)$ approximated for control of the fixed point of the logistic map. Fig. 7b) shows the map and iterated controlled dynamics close to the fixed point. It is clear that the stabilized state is very close to the true fixed point.



a) b)

Figure 7: a) The map $(x_n + u_n(w(x_n)))x_n(1 - x_n)$ approximated for control of the fixed point of the logistic map. b) Zoom of the policy and iteration of the controlled dynamics.

For the logistic map it is a simple task to compute the optimal control signal $u(x)$ which allows to reach the fixed point $x_f = 1 - 1/u_0$ in one iteration:

$$u(x) = \frac{u_0 - 1}{u_0(x(1 - x))} - u_0.$$

Fig. 8 compares this continuous control signal (we refer to it as one-step control) with the discrete one obtained through RL. It is clear that one-step control is only possible for states in the neighborhood of the fixed point or from the neighborhood of states which reach the fixed point naturally in one iteration. One-step control from other regions requires large control signals. The discrete control signals computed through RL approximate the one-step function $u(x)$ in regions from which one-step control with the maximal allowed perturbation $|u(x)| < u_{max} = 0.1$ is possible. The VSGNG algorithm offers a finer approximation of the one-step function compared to the previous algorithm in [8]. In the remaining regions the RL algorithm approximates an optimal policy which brings dynamics in the neighborhood of the desired state in a minimum number of iterations as demonstrated in [8, 7].

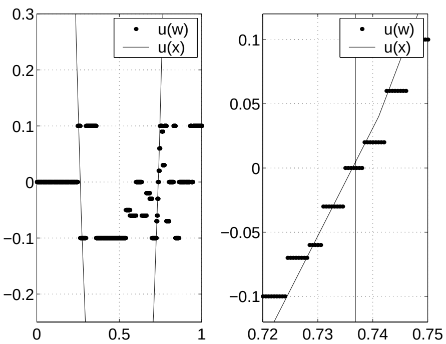


Figure 8: Comparison of optimal one-step control with approximated discrete control.

In similar manner as control of the period-one fixed point is established, control of higher-order fixed points can be achieved.

6 Conclusion

In this paper we combined Fritzke's growing neural-gas algorithm with a reinforcement learning algorithm to obtain a chaos control algorithm. The codebook generation is sensitive to the reinforcement learning value of states. As demonstrated for control of he logistic map, the resulting algorithm is well suited to solve the chaos control problem. This algorithm is a direct improvement of the method suggested in [8], which has successfully been applied to a variety of control problem [8, 9, 7, 10] and is therefore expected to be applicable in a variety of control applications reaching beyond the control of chaotic systems.

References

[1] D. Bertsekas and J. Tsitsiklis. *Neuro-Dynamic Programming.* Athena Scientific, Belmont, MA, 1996.

[2] S. Boccaletti, C. Grebogi, Y.-C. Lai, H. Mancini, and D. Maza. The control of chaos: Theory and applications. *Physics Reports*, **329**:103–197, 2000.

[3] D. Christini and J. Collins. Controlling neuronal noise using chaos control. *Preprint: http://xxx.lanl.gov/find*, 1995.

[4] R. Der and M. Herrmann. Q-learning chaos controller. volume **4**, pages 2472–2475, New York, 1994. IEEE.

[5] B. Fritzke. Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks*, **7 (9)**:1441–1460, 1994.

[6] B. Fritzke. A g rowing neural gas network learns topologies. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems 7*, pages 625–632. MIT Press, Cambridge MA, 1995.

[7] S. Gadaleta. *Optimal chaos control through reinforcement learning.* PhD dissertation, Colorado State University, Department of Mathematics, 2000.

[8] S. Gadaleta and G. Dangelmayr. Optimal chaos control through reinforcement learning. *Chaos*, **9**:775–788, 1999.

[9] S. Gadaleta and G. Dangelmayr. Control of 1-D and 2-D coupled map lattices through reinforcement learning. In F.L. Chernousko and A.L. Fradkov, editors, *Proceedings of Second Int. Conf. "Control of Oscillations and Chaos"*, volume **1**, pages 109–112, St. Petersburg, Russia, 2000. IEEE.

[10] S. Gadaleta and G. Dangelmayr. Learning to control a complex multistable system - art. no 036217. *Physical Review E*, **6303 (3)**:6217–+, 2001.

[11] E. Kostelich, C. Grebogi, E. Ott, and J. Yorke. Higher-dimensional targeting. *Physical Review E*, **47**:305–310, 1993.

[12] T. Martinetz. Competitive hebbian learning rule forms perfectly topology preserving maps. In *ICANN'93: International Conference on Artificial Neural Networks, Amsterdam*, pages 427–434. Springer, 1993.

[13] T. Martinetz, S. Berkovich, and K. Schulten. "Neural-Gas" Network for Vector Quantization and its application to Time-series Prediction. *IEEE Transactions on Neural Networks*, **4**:558–569, 1993.

[14] E. Ott, C. Grebogi, and J.A. Yorke. Controlling chaos. *Physical Review Letters*, **64**:1196–1199, 1990.

[15] M. Paskota, A. Mees, and K. Teo. Directing orbits of chaotic dynamical systems. *Int. J. of Bifurcation and Chaos*, **5**:573–583, 1995.

[16] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, **8**:279–292, 1992.