# Vector Quantization Algorithm for Time Series Prediction and Visuo-Motor Control of Robots

Stan Berkovitch, Philippe Dalger, Ted Hesselroth,
Thomas Martinetz, Benoît Noël, Jörg Walter
and Klaus Schulten
Beckman-Institute and Department of Physics
University of Illinois
Urbana,IL 61801, USA

*We describe a new algorithm for vector quantization and control. The algorithm, in addition to generating a discrete representation of input data by means of Voronoï polyhedra, also generates a tesselation associated with these polyhedra. The tesselation corresponds to a graph which connects neighbouring Voronoï polyhedra and, hence, reflects neighborhood relationships of the embedding space of the data. The algorithm can be extended to approximate through 'training' arbitrary functions defined on the data points. The tesselation allows one to speed up the 'training' through cooperative learning involving nearest, next-nearest, etc. Voronoï polyhedra, reducing the range of cooperation progressively during training. The algorithm produces a table look-up program, assigning optimally tables to inputs and generating rapidly optimal table entries. The entries can be complex data structures, e.g., combinations of scalars, vectors, and tensors. The use of the algorithm has been demonstrated for time series prediction, surpassing existing algorithms, and for visuo-motor control of an industrial robot, e.g., for precise end effector position control. We will attempt to demonstrate by the time of the lecture also an application of the algorithm for visuo-motor control of a pneumatically driven robot arm, a Bridgestone 'RUBBERTUATOR'. This light-weight robot, capable of compliant motion, can be operated in contact with humans. The presented algorithm can acquire the complex response characteristics of this arm through training and, thereby, allows accurate and swift control of pneumatic robot motion.*

### The Control Problem Addressed

In our lecture we want to introduce a novel algorithm for time series prediction and control tasks. We want to provide an overview of the key aspects of this algorithm. For this purpose we describe the algorithm in the context of a particular control problem, namely visuo-motor control of a pneumatically driven robot arm.

The algorithm we seek, in its simplest ramification, should move the end-effector of a multi-jointed robot arm to specified positions $\mathbf{v}$ in the robots 3-dimensional, finite work space $V$, i.e., $\mathrm{v} \in V$. In its simplest form, the $N$ angles $\theta_1, \theta_2, \ldots \theta_N$ at the robots $N$ joints need to be specified such as to achieve the desired position $\mathbf{v}$ of the arms end, i.e., one needs to learn the vector valued function $\mathbf{v}(\vec{\theta})$, where $\vec{\theta}$ represents the $N$-dimensional column vector $(\theta_1, \theta_2, \ldots, \theta_N)^T$. In case of $N = 3$ the functional dependence represented by $\mathbf{v}(\vec{\theta})$ is unique (actually, for wide intervals from which values for $\theta_j, j = 1, 2, \ldots, N$ can be taken, the function can assume two or more discrete branches), for $N > 3$ a continuum of possibilities exists to realize end effector positions $\mathbf{v}$. In the latter case one wants to select $\vec{\theta}$ such that certain conditions are met, e.g., that the arm reaches around obstacles. The issues involved are discussed at length in [1].

### Why a Learning Scheme is Needed

The control problem just stated can be solved by means of conventional robot algorithms. The situation becomes more difficult, and more interesting from our perspective, in case that the control signals actually employed do not specify directly the joint angles. An example is a novel robot arm design which moves arm joints pneumatically through pairs of tubes: inflating and deflating the tubes leads to forces along the tubes which, hence, can move the joints according to the same agonist–antagonist principle realized in the familiar muscle–joint systems of vertebrates. The advantage of such systems is that the motion of joint $j$ is controlled by two pressure variables, the average pressure $\bar{p}_j$ in the two tubes and the pressure difference $\Delta p_j$ between the two tubes. Pressure difference drives the joints, average pressure controls the force with which the motion is executed. This latter feature allows operation at low average pressures and, thereby, allows one to carry out compliant motion of the arm. This makes such robots suitable for operation in fragile environments, in particular, allows direct contact with human operators. The price to be paid for this advantage is that the response of the arm to signals $(\bar{p}_1, \bar{p}_2, \ldots, \bar{p}_N)^T$ and $(\Delta p_1, \Delta p_2, \ldots, \Delta p_N)^T$ cannot be described by 'a priori' known mathematical equations, but rather must be acquired heuristically. One expects that the response characteristsics change during the life time of an arm through wear, after replacements of parts and, in particular, are subject to hysteretic effects.

### The RUBBERTUATOR – A Pneumatically Driven Robot

To master the control of a pneumatically driven robot arm is a worthwhile challenge in two respects. First, the mentioned robot, presently built by Bridgestone under the brand name 'RUBBERTUATOR', through its light weight, its relatively low price and its capacity for compliant motion and direct robot–human contact, might constitute a new robot generation for which control programs need to be furnished; presently, the robot is controlled through a feed-back cycle involving joint angle sensors, the control being slow and relatively imprecise. Second, the close analogy between the joint motions of the RUBBERTUATOR and biological vertebrates opens the possibility that through mastering this robot system we may gain understanding on animal motion, a subject matter which from a theoretical perspective is still ill understood.

How can one obtain information on the response characteristics of the robot arm. We have suggested earlier (see [1] and references quoted therein) to employ a pair of stereo cameras. We have demonstrated in conjunction with an industrial robot (PUMA 560, see [2]) that the signals from the two camera backplanes can be employed for the purpose, i.e., a robot–camera–computer system learns, in fact, to control the arm solely on account of camera images.

### Employing a Linear Feedback Loop

At this point a rather straight forward concept of utmost practical importance needs to be introduced, the linearly controlled feed-back loop. The idea is that rather than to learn directly the precise relationship between joint angles (or other control signals) and end effector positions one learns such relationship only approximately and only for a coarse set $\{\mathbf{v_s},\ \mathbf{s} \in A\}$ of end effector position, i.e., one learns a set of joint angles $\vec{\theta}_\mathbf{s}$, $\mathbf{s} \in A$ for some set A (to be specified later) such that

$$\mathbf{v_s} = \mathbf{v}(\theta_\mathbf{s}) \tag{1}$$

and assigns the remaining control to linear feed-back loops which are based on the expansion

$$\mathbf{v}^{(n)} = \mathbf{v}^{(n-1)} + \mathbf{v}\left(\mathbf{A_s}\left(\mathbf{v}_{\text{target}} - \mathbf{v}^{(n-1)}\right)\right) \tag{2}$$

where $\mathbf{A_s}$ is the Jacobian tensor $\partial\vec{\theta}/\partial\mathbf{v}$ evaluated at the locations $\theta_\mathbf{s}$, $\mathbf{s} \in A$. This expansion attempts to move the end effector to the target location $\mathbf{v}_{\text{target}}$ by linearly correcting the joint angles on account of the remaining deviation $\mathbf{v}_{\text{target}} - \mathbf{v}^{(n-1)}$. Repeated application of (2) starting with $\mathbf{v}^{(0)} = \mathbf{v}(\theta_\mathbf{s})$ leads to a series of end effector positions $\mathbf{v}^{(1)}, \mathbf{v}^{(2)}, \ldots$ which approaches $\mathbf{v}_{\text{target}}$ for suitable $\mathbf{A_s}$.

Schemes for acquiring $\theta_s$ and $\mathbf{A_s}$ have been presented in [1] and their capacity for real applications has been demonstrated in [2,3]. Rather than learning $\theta_s$, $s \in A$ on a very fine mesh $A$ one can learn $\theta_s$ and $\mathbf{A_s}$ on a coarse mesh. For the control of the end effector position of a PUMA 560 through stereo cameras a few hundred mesh points suffice [2,3]. Further control, e.g., grasping motions, require submeshes, which (using a corresponding principle) can be limited to significantly less than hundred mesh points [4]. Obviously, the mesh points must be judiciously chosen, a subject matter which constitutes another important aspect of the algorithm.

### Vector Quantization Scheme

In fact, the choice of mesh points is the most cardinal part of the proposed algorithm as we like to explain now. This part of the algorithms entails two aspects, the aspect of a vector quantization algorithm and the aspect of a graph matching algorithm. We like to explain these two aspects now.

The control algorithm suggested here actually generates, in a training period, a table look-up program. Our discussion above has been mainly concerned with the generation of the table entries. The following discussion is concerned with the assignement of table entries to control tasks. In case of end effector control the tasks can be designated simply by the target positions $v_{\mathrm{target}}$. However, the algorithm can be applied to more general tasks.

The essential property which we require for the task space $V$ is the existence of a distance metric, i.e., for all $\mathbf{u}, \mathbf{v} \in V$ exists a real, positive, etc. distance $d(\mathbf{u}, \mathbf{v})$ such that a *small* $d(\mathbf{u}, \mathbf{v})$ (in most cases) implies that the tasks $\mathbf{u}$ and $\mathbf{v}$ are *similar*, a *large* $d(\mathbf{u}, \mathbf{v})$ implies that the tasks are *dissimilar*. The algorithm determines now a set $\{\mathbf{v_s}, \mathbf{s} \in A\}$ of points $\mathbf{v_s} \in V$ which assign table entries, labelled by $\mathbf{s} \in A$ to tasks. This assignment works as follows. The table entry, labelled $\mathbf{s}$, is connected with the Voronoï polyhedron

$$Vor_1(\mathbf{s}) = \{\mathbf{v} \in V \,|\, \forall \mathbf{r} \in A, \mathbf{r} \neq \mathbf{s}, d(\mathbf{v_s}, \mathbf{v}) \leq d(\mathbf{v_r}, \mathbf{v})\} \tag{3}$$

The Voronoï polyhedra provide a complete partition of the task space $V$, i.e., $V = \cup_{\mathbf{s} \in A} Vor_1(\mathbf{s})$. Hence, any $\mathbf{v} \in V$ can be assigned to a table entry $\mathbf{s}(\mathbf{v})$, specified through the label $\mathbf{s}$ of the Voronoï polyhedron to which $\mathbf{v}$ belongs. (The fact, that a $\mathbf{v}$ may belong to several Voronoï polyhedra is not a nuisance, but rather a great benefit, as we will see shortly.) The question arises how the 'centers' $\mathbf{v_s}$ of the Voronoï polyhedra should be chosen. A suitable criterion is to choose the centers according to the distribution of tasks $P(\mathbf{v})$ encountered in a training episode, i.e., to select more centers in regions of $V$ where $P(\mathbf{v})$ is large and *vice versa*. A possible criterion would be to assign $\{\mathbf{v_s}, \mathbf{s} \in A\}$ such that

$$E(\{\mathbf{v_s}, \mathbf{s} \in A\}) = \int d\mathbf{v} P(\mathbf{v}) d(\mathbf{v}, v_{\mathbf{s}(\mathbf{v})}) \tag{4}$$

assumes a minimum. Such criteria are well-known in the theory of vector quantization algorithms (for a more detailed discussion and references see [1]). We will explain below how the minimization of (4) is achieved. Details can be also found in [5].

### Learning a Neighborhood Graph

So far the algorithm assigning table entries to tasks has been of a rather conventional vector quantization type. A crucial new feature of the algorithm is that a graph is being developed, the nodes of which are the elements of $A$, the edges being defined below. This graph can be exploited to enhance training results and training speeds. In fact, without exploiting such graph structures many control problems cannot be learned (see [1,2,3]). Also the gain in training speed can be very considerable (see [1,2,3]). The assignement of edges can be achieved in principle (a practical algorithm is presented in [5]) as follows. One considers so-called second order Voronoï polyhedra defined through

$$Vor_2(\mathbf{r}, \mathbf{s}) = \{\mathbf{v} \in V \,|\, \forall \mathbf{t} \in A, \mathbf{t} \notin \{\mathbf{r}, \mathbf{s}\}, d(\mathbf{v}, \mathbf{v_r}) \leq d(\mathbf{v}, \mathbf{v_t}) \wedge \mathbf{v_s}) \leq d(\mathbf{v}, \mathbf{v_t})\} \tag{5}$$

One assigns now edges between all pairs of nodes $(\mathbf{r}, \mathbf{s})$ the associated second order Voronoï polyhedra (5) of which are not empty. One obtains, thereby, a graph which reflects the neighborhood relationships of the first order Voronoï polyhedra (4).

Unfortunately, actual algorithms [5] can achieve assignements of edges only if the volume of the second order Voronoï diagrams is large enough, a condition which is not necessarily met in Euclidean tasks spaces of dimension three or larger. However, the algorithm presented in [5] usually captures a large fraction of neighborhood relationships through edge assignement.

A particularly straightforward interpretation of the graph described above can be given in case of a two-dimensional Euclidean space. In this case the Voronoï polyhedra are actually polygons and the structure of edges are the dual of these polygons, called the Delaunay tesselation.

We like to comment finally on the reason why the graph structure described can improve the generation of table entries for control tasks. The reason is that the edges of the graph structure connect those tables which are closest with respect to the metric of the task space. The edges provide a hierarchy of nearest neighbors, next nearest neighbors (connected through at least two edges), etc. During training one can asume then that tables, which are neighboring, have to learn similar entries. One can exploit this by incorporating a cooperative learning scheme involving nearest, next-nearest, etc. Voronoï polyhedra, reducing the range of cooperation progressively to achieve asymptotically an optimal resolution of table content.

## Summary

The algorithm described above in the context of a control problem, has the important feature that it employs a nodes $\mathbf{v_s}$, $\mathbf{s} \in A$ together with a self-generated graph (edges between nodes). Previous algorithm (the extended Kohonen algorithm, see [1]) employed a lattice of nodes (which also corresponds to a node–edge, i.e., graph, structure) which was fixed 'a priori'. The new algorithm 'learns' the topology (neighborhood relationships) of the task space and, hence, does not require that the topology of the task space is known before hand and it can deal with complex topologies, like those of disjoint task spaces of mixed dimensionalities.

The algorithm can also be used in a somewhat simpler context of time series prediction. In this case the input data, corresponding to the tasks in the aforementioned example, are time series $y(t_1), y(t_2), \ldots, y(t_n)$ of a function $y(t)$, and the algorithm is asked to 'predict' the function value $y(t_{n+1}), t_{n+1} > t_1, t_2, \ldots t_n$. The problem, in principle, corresponds to learning to approximate the function $Y = y(t_{n+1})$ for vector-valued arguments $Y_r = (y(t_1), y(t_2), \ldots, y(t_n))^T$. We have developed a suitable learning rule for this purpose and applied it succesfully to a function $y(t)$ described by the Mackey-Glass equation [6]. This application will be presented in our lecture, in particularly, it will be demonstrated that the algorithm compares very favourably with existing algorithms, e.g., those of Moody and Darken and of Lapedes and Farber.

The algorithm is described in detail in [5], the original report, as well as in [2-4, 6-7].

## References

[1] *Neuronale Netze: Eine Einführung in die Neuroinformatik selbstorganisierender Abbildungen*, H. Ritter, Th. Martinetz, and K. Schulten (2nd enlarged edition, Addison-Wesley, Bonn, 1990)
*Neural Computation and Self-Organizing Maps: An Introduction*, H. Ritter, Th. Martinetz, and K. Schulten (revised, English edition, Addison-Wesley, New York, 1991)

[2] Industrial Robot Learns Visuo-Motor Coordination by Means of 'Neural Gas' Network, J.A.Walter, Th.M.Martinetz, and K.Schulten, *Proceedings of the International Conference on Artificial Neural Networks, Helsinki, 1991)*

[3] Neural Network with Hebbian-like Adaptation Rules Learning Control of a PUMA Robot, Th.Martinetz and K.Schulten, (submitted to NIPS-91)

[4] Hierarchical Neural Net for Learning Control of a Robot's Arm and Gripper, Th.Martinetz and K.Schulten, *IJCNN International Joint Conference on Neural Networks, San Diego, California, July 1990*, pp. II–747 to II–752 (The Institute of Electrical and Electronics Engineers, New York, 1990)

[5] A 'Neural Gas' Network Learns Topologies, Th.Martinetz and K.Schulten, *Proceedings of the International Conference on Artificial Neural Networks, Helsinki, 1991)*
[6] S. Berkovitch, Th, Martinetz and K. Schulten, application to time series prediction, manuscript in preparation
[7] Ph. Dalger, B. Noël, Th. Martinetz and K. Schulten, mathematical analysis of Delaunay tesselation on the basis of 1st and 2nd order Voronoï polygons in case of 2-dimensional task spaces, manuscript in preparation