Sensory Flow Segmentation using a Resource Allocating Vector Quantizer

Fredrik Linåker[1,2] and Lars Niklasson[1]

[1] Department of Computer Science, University of Skövde,
P.O. Box 408, SE-541 28 Skövde, Sweden
[2] Department of Computer Science, University of Sheffield,
Regent Court, 211 Portobello Street, Sheffield S1 4DP, United Kingdom
fredrik.linaker@ida.his.se
lars.niklasson@ida.his.se

Abstract. We present a very simple unsupervised vector quantizer which extracts higher order concepts from time series generated from sensors on a mobile robot as it moves through an environment. The vector quantizer is constructive, i.e. it adds new model vectors, each one encoding a separate higher order concept, to account for any novel situation the robot encounters. The number of higher order concepts is determined dynamically, depending on the complexity of the sensed environment, without the need of any user intervention. We show how the vector quantizer elegantly handles many of the problems faced by an existing architecture by Nolfi and Tani, and note some directions for future work.

1 Introduction

As a mobile robot moves through an environment, it receives a sequence of inputs through its sensory equipment, this sequence of inputs is called the 'sensory flow'. The sensory flow can easily be in the order of thousands, or even millions, of discrete samples. Finding relations and reoccurring phenomena in this sequence is a computationally intractable task, especially when we receive noisy or even faulty inputs which need to be filtered out. However, instead of working on the sensory input sequence directly, abstractions can be formed which capture the general characteristics of the inputs instead of each individual input. For instance, when the robot is moving down a corridor, it receives basically the same type of inputs time step after time step; a wall to the left and a wall to the right. As the robot encounters a fork in the corridor, the inputs change radically, suddenly the front sensors may become active and the left and right sensors no longer sense walls. These distinct changes in the sensory inputs can be exploited by the robot. If the task of the robot requires it to remember the path it just took through a maze it could, instead of storing each individual input it received, requiring extensive memory capabilities, store an abstraction of the inputs, e.g. 'corridor, left turn, room, corridor', requiring only a fraction of storage space. It could later use this stored abstraction to navigate through the maze, or to output a description to the user of what the robot has perceived. Similarly, if the task involves finding relations between events that are far apart in time, an error-signal will vanish as it passes through thousands and thousands of intermediate sequence elements, making it practically impossible to find such relations. This problem becomes more manageable if the input sequence is segmented into a sequence of higher order concepts, each represented by a unique symbol. Finding relations between these symbols is a much more viable task than working directly on the input sequence.

The task of finding reoccurring sub-sequences is however a quite complex task if there are no clearly defined boundaries in the input sequence. However, the sequence used in this paper has easily identifiable and stable regions with fairly sharp transition borders, similar to the sequence shown in Figure 1. It is generated from the sensors of a mobile wall-following Khepera robot, which moves through a simulated environment.

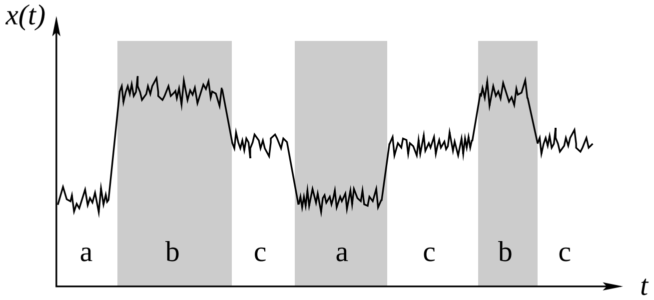

Fig. 1. The input signal $x(t)$ from a distance sensor on a wall-following mobile robot jumps between regions of fixed signal mean with added noise; the task is to find the different signal levels and to detect the transitions between them. Labelling each segment with a symbol, the entire sequence can be stored (with some loss of information) using just seven symbols instead of hundreds, or even thousands, of distinct input values. The length of each sub-sequence is however lost during the mapping

Nolfi and Tani [7, 9] conducted experiments using a similar wall-following robot and segmented the sensory flow using a hierarchical neural network architecture consisting of several prediction and segmentation networks. While their system managed to extract higher order concepts from the sensory flow, such as 'walls', 'corners' and 'corridors', it had problems finding sub-sequences which did not occur very often, yet were quite distinct. Furthermore, they needed to manually specify exactly how many higher order concepts the system should split the sequence into instead of letting the system decide this on its own, depending on the complexity of the sequence.

In the following, a simple constructive vector quantizer which solves these problems efficiently is designed. It finds an 'appropriate' segmentation using just a single presentation of the input sequence, unlike Nolfi and Tani's methods which require repeated presentations of the input sequence. Higher order concepts which do not occur very frequently, but are distinct, are successfully extracted by the system. Furthermore, the system determines the number of higher order concepts automatically, based on the sequence characteristics.

In section 2, Nolfi and Tani's original experiments are summarized, and a number of problems with their system are highlighted. In section 3, we describe the resource allocating vector quantizer, which solves these problems. It is defined mathematically and results of the experiments conducted with this architecture are presented. Finally, in section 4, the main advantages of the new approach are summarized.

2 Existing Methods

In experiments carried out by Nolfi and Tani [7,9], different neural network architectures were investigated which segmented sensory input sequences from mobile robots. In the former paper [9], they designed a modular system of gated experts, where each module represented a sub-sequence. In the latter paper [7] they presented an altered architecture which involved a simpler, yet still quite complex, hierarchical architecture which is described below.

2.1 Architecture

The hierarchical architecture Nolfi and Tani [7] used consisted of a first level input prediction network, a segmentation network, and a second level sub-sequence prediction network.

The first level prediction network was a recurrent network with 10 input units (encoding the 8 sensor values and 2 motor values at time t), 3 hidden units, and 8 output units (encoding the expected 8 sensor values at time $t+1$). The activation of the hidden units at the previous time step was fed into 3 additional input units the succeeding time step, providing a memory of previous events which could help in predicting the next inputs.

The activation of the hidden units of the first level prediction network constituted the input to the segmentation network, which thus had 3 input units. These input units were connected to a pre-defined number of 'winner take all' output units which each represented a different higher order concept. Nolfi and Tani used 3 such output units. They argued that a segmentation based on the hidden unit activation of a prediction network, instead of using the input sequence directly, allowed enhancement of less frequent sub-sequences.

The segmentation network was updated in an unsupervised manner, similar to a Self-Organizing Map [4] with neighbourhood range set to zero (i.e. no neighbours were updated). There was also a second level prediction network which tried to find regularities in the sequence of extracted higher order concepts. This particular aspect is however not relevant here, but the reader is encouraged to read the original paper for more details about this.

2.2 Experiments

The environment consisted of two simulated rooms of different sizes, connected together by a short corridor. The robot, a simulated Khepera robot, was controlled by a fixed wall-following behaviour which was not affected by the prediction and segmentation networks. The networks merely were idle observers, trying to find regularities in the sequence of inputs. Nolfi and Tani's experiments are here replicated, using Olivier Michel's publicly available Khepera Simulator [5], and the resulting segmentation is depicted in Figure 2.

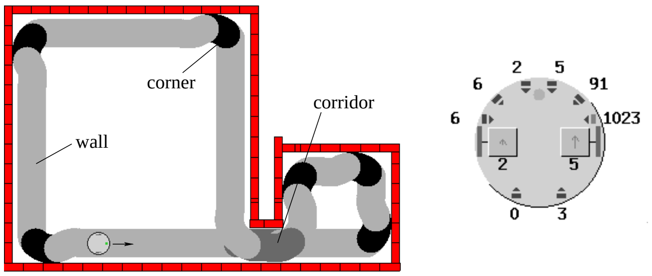

Fig. 2. The simulated environment and the segmentation acquired using the Nolfi and Tani approach. Each unit in the segmentation network has been assigned a different shade; the shade of the winning unit at each time step is shown. The simulated Khepera robot is shown to the right; it has eight distance sensors and two motors, distance sensor values are in the range [0,1023] and motor values are in the range [-10,10]

As noted by Nolfi and Tani, the extracted sub-sequences can be described as 'walls' (light gray), 'corridors' (gray) and 'corners' (black).

2.3 Problems

The architecture used in Nolfi and Tani's experiments required many repeated presentations of the same input sequence (Nolfi and Tani used over 300 laps in the environment) in order to extract the sub-sequences, which made it computationally intractable to repeat the experiments with different parameter sets. This is a problem since they had many user-specified parameters which all influenced the outcome of the segmentation, e.g. number of hidden units in the prediction nets, choice of learning rates, weight initialization, decay value, etc., the choice of which could lead to very different segmentations.

Moreover, the training had to be split into different phases, one for training the first level prediction network, another for training the segmentation network. The duration of these phases also needed to be decided by the user.

Further, the system could not detect situations which had low density, i.e. that did not occur very often or sustain for a long period of time, for example very short corridors. Even more severe, the system could suffer from catastrophic forgetting if new situations arose which lead to a reallocation of the hidden activation space or the first layer prediction network. Thus making the existing segmentation layer weights inappropriate or even invalid.

Finally, the user was forced to manually specify the number of categories which should be extracted, instead of letting the system determine this on its own as the robot negotiated the world.

The system suggested in the following, deals with all of the above identified problems. It has the ability to segment the input sequence using a single presentation, without having to split the training into different phases. It also operates directly on the input sequence, avoiding the risk of catastrophic forgetting. Finally, the system determines the number of higher order concepts automatically depending on the complexity of the input sequence. It is also able to handle situations where new concepts are introduced dynamically.

3 The Resource Allocating Vector Quantizer

The resource allocating vector quantizer (RAVQ) represents higher order concepts using model vectors. Additional model vectors are allocated dynamically when new and stable situations are encountered.

3.1 Related Work

The process of using model vectors to represent categories has been successfully employed in a number of different architectures, e.g. in Kohonen's Learning Vector Quantization and Self-Organizing Maps [4]. Such networks however rely on a fixed number of model vectors. Careful analysis of the complexity of the input signal has to be performed by the user in order to specify an appropriate number of units. Using too few units will force the system to disregard some of the possible categories, while too many may create unwanted, or spurious, categories.

This problem has been alleviated through the development of constructive systems, e.g. systems by Platt [8], Fritzke [3], Chan and Vetterli [2], i.e. systems which are able to allocate further resources whenever deemed necessary. While a variety of such systems are employed today, the family of adaptive resonance theory (ART) networks by Carpenter and Grossberg [1] are most relevant to the design of the RAVQ.

The ART networks classify inputs into categories, where each category has its own prototype. The prototype depicts the typical input pattern which is associated with the category. When new input patterns are encountered which do not closely match any of the existing categories, further categories might be created. This corresponds to the allocation of an additional output unit and a corresponding prototype. The new prototype is initialized to match the present, unfamiliar, input pattern. The ART networks do however have problems coping with noisy input patterns which can result in the incorporation of spurious categories ([6] page 138). Many such noisy inputs can however be filtered using a simple method employed by the RAVQ, which stores a number of previous inputs to provide a context for the current input pattern.

3.2 Theory

The RAVQ is specifically constructed for sequentially ordered inputs where each model vector comes to represent a different sub-sequence. The model vectors can be viewed as *points* in the input space, not as sequences or paths. This implies that only sub-sequences with stable signal values can be represented accurately using such a model vector. This means that the RAVQ is limited to input sequences where the signal mean basically remains fixed for a period of time with some occasional transition to a new signal mean. As is shown here, this is adequate for the segmentation of a sequence generated from the sensors of a mobile Khepera robot.

The RAVQ has only three user specified parameters: a window size n, a mismatch reduction requirement δ, and a stability criterion ϵ. The system forms a set of model vectors which are placed approximately in the centre signal-value of the sub-sequence it is meant to represent. Initially empty, the set of model vectors is increased as soon as novel, stable, situations are encountered. This is done as follows:

A moving average of the last n inputs is calculated in order to filter out noise in the input signal. This moving average is incorporated as a new model vector (we denote the moving average as a 'model vector' as soon as it has been incorporated) if it characterizes a stable mismatch reducing situation, i.e. it meets both the mismatch reduction criterion and the stability criterion:

- the reduction criterion is that the moving average alone should account for the inputs better than the existing model vectors do, reducing the mismatch at least by δ,
- the stability criterion is that the deviation between the inputs and the moving average should stay below a certain threshold ϵ, otherwise the situation is not characterized as being stable (the robot may be switching between existing model vectors or experiencing a temporary sensor fluctuation).

Each of the above criteria is not sufficient on its own, as only having a reduction criterion can lead to the incorporation of model vectors for erratic inputs (i.e. suffering from the same problems as the ART networks). Only having a stability criterion, on the other hand, can lead to the incorporation of new model vectors which are virtually identical to already incorporated model vectors.

3.3 Definition

The key part of the RAVQ is an input buffer of size n. At each time step this buffer stores the last n input vectors $x(t) \in X$. In the first n time steps ($t = 0, \ldots, n-1$), inputs are simply recorded into the buffer and the RAVQ is not activated until time step $t = n - 1$, when the input buffer has been filled. At this time, the set $M(t)$ of model vectors m, is initialized to the empty set:

$$M(n-1) = \emptyset \ , \tag{1}$$

and at each successive time step t, the finite moving average $\overline{x}(t)$ is calculated:

$$\overline{x}(t) = \frac{1}{n} \sum_{i=0}^{n-1} x(t-i) \ . \tag{2}$$

We define a distance metric $d(V, X)$ which specifies the mean of the shortest distances between a set of (model / moving average) vectors $v_j \in V$ and a set of input vectors $x_i \in X$, i.e. the average error for the inputs given a set of vectors:

$$d(V, X) = \frac{1}{|X|} \sum_{i=1}^{|X|} \min_{1 \leq j \leq |V|} \{||x_i - v_j||\}; x_i \in X, v_j \in V \ , \tag{3}$$

where $||.||$ denotes the Euclidean distance. That is, it returns the best match between the model vectors and each given input vector. This distance metric can be used to calculate the mean distance $d_{\overline{x}(t)}$ from each of the last n inputs to the moving average $\overline{x}(t)$:

$$d_{\overline{x}(t)} = d(\{\overline{x}(t)\}, \{x(t), \ldots, x(t-n+1)\}) \ , \tag{4}$$

The same distance metric is used to calculate the distance $d_{M(t)}$ between each of the last n inputs and the best matching model vector at each time step:

$$d_{M(t)} = \begin{cases} d(M(t), \{x(t), \ldots, x(t-n+1)\}) & |M(t)| > 0 \\ \epsilon + \delta & \text{otherwise} \ . \end{cases} \tag{5}$$

If there are no model vectors in $M(t)$, i.e. the RAVQ has just started, the distance $d_{M(t)}$ is set to be sufficient for incorporation of this moving average into the set of model vectors. For the moving average to be incorporated as a model vector, the mean distance between the last n inputs and the moving average $\overline{x}(t)$ must stay below the threshold ϵ, i.e. it must constitute a stable input location, and the improvement which is possible through an incorporation of the current moving average into the set of model vectors must exceed the minimum improvement requirement δ:

$$M(t+1) = \begin{cases} M(t) \cup \overline{x}(t) & d_{\overline{x}(t)} \leq \min(\epsilon, d_{M(t)} - \delta) \\ M(t) & \text{otherwise} \ . \end{cases} \tag{6}$$

The higher order concept the system is in at time t, is indicated by the selection of the best matching model vector $win(t)$ in relation to the moving average $\overline{x}(t)$ at that time step:

$$win(t) = \arg \min_{1 \leq j \leq |M(t)|} \{||\overline{x}(t) - m_j||\}; m_j \in M(t) \ . \tag{7}$$

That is, $win(t)$ specifies the index of the best matching model vector at time step t.

3.4 Experiments

Replicating the segmentation part of the Nolfi and Tani [7] experiments, but in Olivier Michel's publicly available Khepera Simulator [5], and this time using the RAVQ, the same segmentation was achieved in only a fraction of the time (Figure 3). The system operates directly on the input sequence. The input vector to the RAVQ had 10 elements, 8 distance sensor values and 2 motor values, all normalized to the range [0.0,1.0]. The parameters used were $n = 10$, $\delta = 0.9$ and $\epsilon = 0.2$.

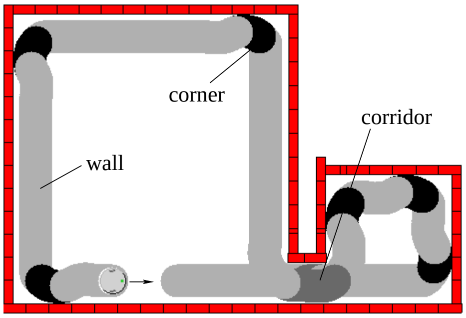

Fig. 3. The resulting segmentation directly from the first lap (about 1,900 time steps) in the environment. There is a slight delay of n inputs before the first corridor and corner segments are detected. The winning model vector $win(t)$ at each time step is indicated using different shades

The system is also capable of instantaneous learning of new, previously unencountered, situations at any time of the simulation since there is no decaying learning rate or similar parameter which would make such learning harder at later points in the simulation. For instance, if a turning corridor is added (something which was not present in Nolfi and Tani's original experiments) the RAVQ first tries to handle the situation using the existing model vectors; the best match is the 'corner' model vector. But this model vector does not exactly match the situation since there now is a wall on the left side, the RAVQ swiftly adds a new model vector for 'turning corridor' (Figure 4a), and the next time this situation occurs, this model vector is used accordingly (Figure 4b). (There is no sensor pointing back to the sides, this is why the system produces a 'wall' segment just before the 'corridor' / 'turning corridor' since the inputs depict a wall to the right but nothing to the left.)

The architecture used by Nolfi and Tani [7] is incapable of detecting new situations, there is no mechanism for adding more units, representing sub-sequences, when such units are needed. Further, starting out with 'extra' units would not help either since they would soon be allocated to better cover the more dense input regions, and would subsequently be hard to reallocate to handle new situations instead.

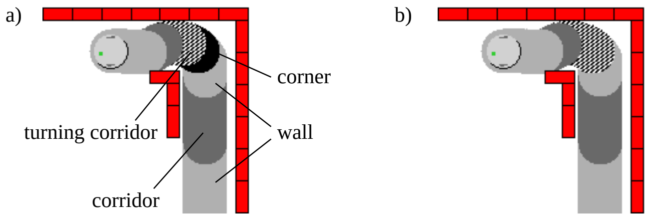

Fig. 4. A new model vector is incorporated as soon as a new situation is detected, for instance a corridor which turns (a). The uptake region of the new model vector steals space from the existing model vectors; the 'corner' model vector is no longer activated the next time this particular situation is encountered (b)

4 Conclusions

We have described a resource allocating vector quantizer (RAVQ) which is capable of single presentation segmentation of the sensory flow of a mobile robot. Compared to the earlier models of Nolfi and Tani [7], the RAVQ approach for segmentation is considerably simpler, requiring no division of the training into separate phases, and with just three user specified parameters: a window size n, a mismatch reduction requirement δ and a stability criterion ϵ. The RAVQ elegantly handles low density inputs; not basing the placement of model vectors according to input frequency but novelty and stability. Further, and more importantly, the RAVQ dynamically determines the appropriate number of categories to use, without the need for any user intervention, and finally, since the RAVQ works directly on the input sequence, there is no risk of catastrophic forgetting due to a reallocation of the hidden space which could be very damaging in previously used methods for this task.

The RAVQ is limited to sequences where the signal mean basically remains fixed for a period of time with some occasional transition to a new signal mean. The cause of this limitation is that each higher order concept is represented using a single model vector, placed directly in input space. If instead regions or paths could be identified in the input space, and represented using model vectors, sequences of inputs could form higher order concepts. This could, for instance, correspond to situations where a mobile robot travels through a broadening corridor or moves diagonally through a room. Future extensions should also involve the incorporation of adaptation of the model vectors to better account for the higher order concepts which they represent. This could be performed using a learning rule similar to that employed in Self-Organizing Maps [4].

The extracted sequence of higher order concepts can be viewed as an abstract representation of the environment, created through the eyes of the robot. The granularity of the segmentation can be controlled indirectly through the mismatch reduction and the stability criteria. As different levels of segmentations may be needed for different tasks, a series of robot tasks should be designed, testing how useful the higher order concepts actually are in tackling memoryintensive tasks such as path learning through mazes and finding long time dependencies in the sensory-motor flow.

Acknowledgements

This paper was made possible by a grant from The Foundation for Knowledge and Competence Development (1507/97), Sweden, and a grant to both authors from the University of Skövde, Sweden. We would also like to thank Kim Laurio and Tom Ziemke for providing insightful comments on an early draft of this paper.

References

1. Carpenter G. A., Grossberg S.: ART 2: Self-organization of stable category recognition codes for analog input patterns, *Applied Optics* (1987) 26: 4919-4930.
2. Chan C., Vetterli M.: Lossy compression of individual signals based on string matching and one pass codebook design, In *Proceedings ICASSP* (1995) 2491-2494.
3. Fritzke B.: Vector quantization with a growing and splitting elastic network, In *ICANN'93: International Conference on Artificial Neural Networks*, Springer (1993) 580-585.
4. Kohonen T.: *Self-Organizing Maps (second edition)*, Springer (1995).
5. Michel O.: *Khepera simulator package version 2.0: Freeware mobile robot simulator* (1996), `http://diwww.epfl.ch/w3lami/team/michel/khep-sim/`
6. Nehmzow U.: *Mobile Robotics: A Practical Introduction*, Springer (2000).
7. Nolfi S., Tani J.: Extracting Regularities in Space and Time Through a Cascade of Prediction Networks: The Case of a Mobile Robot Navigating in a Structured Environment, *Connection Science* (1999), 11(2).
8. Platt J.: A Resource-Allocating Network for Function Interpolation, *Neural Computation* (1991) 3(2): 213-225.
9. Tani J., Nolfi S.: Learning to perceive the world as articulated: an approach for hierarchical learning in sensory-motor systems, In *Proc. of the Fifth Int. Conf. on Simulation of Adaptive Behavior*, R. Pfeifer, B. Blumberg, J.A. Meyer and S.W. Wilson (Eds.), MIT Press (1998) 270-279.