ESANN'2006 proceedings - European Symposium on Artificial Neural Networks
Bruges (Belgium), 26-28 April 2006, d-side publi., ISBN 2-930307-06-4.

Unsupervised Clustering of Continuous Trajectories of Kinematic Trees with SOM-SD

Jochen J. Steil[1], Risto Koiva[1] and Alessandro Sperduti[2] *

1-Neuroinformatics Group, University of Bielefeld, Germany
{rkoiva,jsteil}@techfak.uni-bielefeld.de
2-University of Padua, Department of Pure and Applied Mathematics
sperduti@math.unipd.it

Abstract. We explore the capability of the Self Organizing Map for structured data (SOM-SD) to compress continuous time data recorded from a kinematic tree, which can represent a robot or an artifical stick-figure. We compare different encodings of this data as tree or sequence, which preserve the structural dependencies introduced by the physical constraints in the model to different degrees. Besides computing a standard quantization error, we propose a new measure to account for the amount of compression in the temporal domain based on correlation of the degree of locality of the tree and the number of winners in the map for this tree. The approach is demonstrated for a stick-figure moving in a physics based simulation world. It turns out that SOM-SD is able to achieve a very exact representation of the data together with a reasonable compression if tree encodings rather than sequence encodings are used.

1 Introduction

In cognitive robotics, the advent of multi degree of freedom (DOF) robots with increasing abilities for smooth human-machine interaction calls for advanced methods for interpretation of movements and action recognition. Among the most promising current approaches are methods to generate from video streams joint angle data for kinematic stick-figure models of the person or robot to be observed [1, 2]. As these approaches will soon lead to large amounts of continuous time trajectory data, machine learning methods for automatic categorization and clustering will apparently be important for grounding successive more symbolic processing steps like action recognition on reduced prototypic descriptions.

Such approaches, however, typically work with sequences of one- or multi-dimensional vectors, where structural interdependencies between the vector components can not be encoded. On the other hand, in kinematic trees describing robots or stick-figures representing humans such dependencies are apparent due to the physical constraints acting through the links. An intuitive way to encode this information is to resort to the graph structure of the kinematic tree and process it as structured data directly in this form. For the current study, we generate such data from a physics based simulation of a stick-figure with fourteen joint angles, which from a defined position falls down to the floor subject to

*This work was supported by the "Vigoni Program", sponsored by DAAD and CRUI.

gravity, to different initial forces, and in the presence of obstacles. While recursive SOM's have already been applied to continuous time sequences [3], a new aspect in the current study is the mixture of continuous time and tree-structure of the data given at a single time step. We explore the application of the SOM-SD introduced in [4] to this data and compare two different tree-like and a more traditional vectorial sequence encoding with respect to the quantization error.

2 The SOM-SD framework

SOM-SD is a SOM capable of processing structured input in the form of a directed acyclic graph, where each vertex of the graph has attached a label (e.g., a real valued vector). Basically, the functioning of SOM-SD can be understood as the recursive application of a standard SOM where the input is properly coded to take in consideration the structural information. In fact, in SOM-SD, a network input vector $\boldsymbol{x}_v$ representing the information of a vertex v of the input graph is built through the concatenation of the data label $\boldsymbol{v}$ and coordinates of the mapping, on the same network, of child nodes $\boldsymbol{y}_{ch[v]}$ so that $\boldsymbol{x}_v = [\boldsymbol{v}^T, \boldsymbol{y}_{ch[v]}^T]^T$. The vectors can be made constant in size if the maximum out-degree of any vertex in the dataset is known. Assuming that the maximum out-degree is o, then for vertexes with less than o children, padding with a default coordinate, typically the "impossible" coordinate (-1,-1), is applied. As a result, the $\mathbf{x}$ are $k = p + 2o$ dimensional vectors, where p is the dimension of the data label. The codebook vectors $\mathbf{m}$ are of the same dimension.

In order to account for the coding of both data label and structural information in the same input item, the winning neuron is selected by using a modified Euclidean distance, i.e. $r = \arg\min_i \|(\mathbf{x}_v - \mathbf{m}_i)\boldsymbol{\Lambda}\|$ where $\boldsymbol{\Lambda}$ is a $k \times k$ dimensional diagonal matrix. Its diagonal elements $\lambda_{11} \cdots \lambda_{pp}$ are set to μ_1, all remaining diagonal elements are set to μ_2. The constant μ_1 influences the contribution of the data label component to the Euclidean distance, and μ_2 controls the influence of the children's coordinates to the Euclidean distance. The difficulty of this approach is the estimation of good values for the parameters μ_i. Using large values for μ_2 increases the mapping precision according to structural information but neglects the importance of the information provided by the data labels. Setting large values for μ_1 increases the focus on the data label. In the current paper, the μ_1 are automatically chosen with respect to the structure of the tree and the variances in the label data, leading to large values for μ_1 (> 0.98). More details on how SOM-SD works can be found in [4].

3 Evaluation and quantization measures

It is very difficult to evaluate the ability of a map to encode relevant spatio-temporal relationships contained in the input trajectories. There are different issues that must be considered. First of all we would like to have a small quantization error, i.e. we would like the weight vectors to retain all the statistical information about the labels distribution. Moreover, we would like the map to preserve also the main temporal (or more in general, structural) features of the trajectories, i.e., the "history" or "contextual" information contained in the trajectories (e.g., that a point in space is reached by a trajectory only passing across a specific subspace and following a specific direction). This means that we expect that the same region of the input space is "covered" by several neurons, each one specialized to recognize from which origin and direction the trajectory reaches that region. Of course, this ability should be obtained with no reduction of the quantization capability of the network.

We start our study by addressing this issue on the representation side, i.e. how can a trajectory of a structured object be represented such that both contextual information is preserved and quantization error is minimized.

In the following, we clarify our notion of quantization error. Consider trees $\boldsymbol{T}$ from a set of trees $\mathcal{T}$ with vertices $v \in V(\boldsymbol{T})$, the set of vertexes of $\boldsymbol{T}$. If with $r(\boldsymbol{x}_v)$ we denote the index of the winning neuron for input $\boldsymbol{x}_v$, the (squared) quantization error for the standard SOM is defined as

$$E_{som} = \sum_{\mathbf{T} \in \mathcal{T}} \sum_{v \in V(\mathbf{T})} (\boldsymbol{x}_v - \boldsymbol{m}_{r(\boldsymbol{x}_v)})^2.$$

This way of computing the quantization error, however, includes also the structural component introduced by the input encoding. We can avoid this contribution, when considering SOM-SD, by computing the quantization error as:

$$E_{som-sd} = \sum_{\mathbf{T} \in \mathcal{T}} \sum_{v \in V(\mathbf{T})} (\boldsymbol{v} - \hat{\boldsymbol{m}}_{r(\boldsymbol{x}_v)})^2,$$

where $\hat{\boldsymbol{m}}$ is the subvector obtained by selecting the first p components of the codevector $\boldsymbol{m}$, i.e., the portion which refers to the input label.

We also study how spatio-temporal information is coded into the maps. For this purpose we need to define a notion of "locality" of a trajectory represented as a tree. The "locality" $L(\boldsymbol{T})$ of a tree $\boldsymbol{T}$ is defined as:

$$L(\boldsymbol{T}) = \sum_{v \in V(\mathbf{T})} \sum_{u \in \downarrow v} \|\boldsymbol{v} - \boldsymbol{u}\|$$

where $\downarrow v = \{u | \exists path(v, u)\}$. Notice that this definition holds also for the special case in which $\boldsymbol{T}$ is a tree with out-degree 1, i.e. a list or sequence. If we define the set of winning neurons n_j for $\boldsymbol{T}$ as

$$win(\boldsymbol{T}) = \{n_j | \exists v \in V(\boldsymbol{T}),\ \text{s.t. } n_j \text{ is the winner for } \boldsymbol{x}_v\}$$

we can study for a set of trees $\mathcal{T}$ the correlation between locality and number of distinct winners. In fact, if we define the random variables L, with realizations in the set $\{L(\boldsymbol{T}) | \boldsymbol{T} \in \mathcal{T}\}$, and win, with realizations in the set $\{|win(\boldsymbol{T})|\ | \boldsymbol{T} \in \mathcal{T}\}$, we can compute the following (squared) correlation coefficient:

$$r^2(L, win) = \frac{\left(\sum_{\mathbf{T} \in \mathcal{T}} (L(\boldsymbol{T}) \cdot |win(\boldsymbol{T})| - \bar{L} \cdot \overline{win})\right)^2}{\left(\sum_{\mathbf{T} \in \mathcal{T}} (L(\boldsymbol{T})^2 - \bar{L}^2)\right) \cdot \left(\sum_{\mathbf{T} \in \mathcal{T}} (|win(\boldsymbol{T})|^2 - \overline{win}^2)\right)} \qquad (1)$$

where $\bar{L}$ and $\overline{win}$ are the mean values of L and win, respectively.

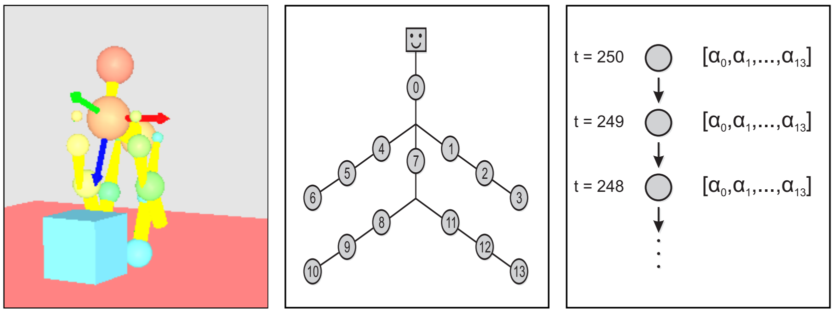
0
1
2
3
4
5
6
7
8
9
10
11
12
13
t = 250
$[\alpha_0,\alpha_1,...,\alpha_{13}]$
t = 249
$[\alpha_0,\alpha_1,...,\alpha_{13}]$
t = 248
$[\alpha_0,\alpha_1,...,\alpha_{13}]$

Fig. 1: Stick-figure model in physics based simulation (left) falling towards an object. Respective kinematic tree with 14 joints (center). Stick-figure movement trajectory in vector-sequence representation, α_i refers to i-th joint (right).

4 Data preparation and experimental setup

We have generated structured data from a physics based simulation of the stick-figure, which is situated in a virtual world and subject to gravity, forces and collisions in the world. The stick-figure is composed of balls representing the body (see Fig.1) connected by fourteen joint angles each constrained by joint limits. For the collision checking links between the balls are ignored, however, the whole figure behaves quite realistically. In initial position the figure is standing straight up and is connected with a spring to the ceiling. Then we apply a random force which pushes the figure either to fall down in some direction, potentially hitting an object, or to return after oscillatory movements to the initial position if the force is too weak and the spring can hold to figure. Thus there are qualitatively different dynamics like stable (force close to zero), returning (small forces), unstable (falls down), and obstacle (dependent on direction). For the experiments described below we use 90 sequences of 250 time steps.

Sequences of movements of the stick-figure have been represented in three different ways. In the first representation, the state of each joint at time t is collected in a real valued vector of dimension 14 (joint vector), and a sequence of movements is represented as a sequence of these vectors. Since we are using SOM-SD, the sequence is actually represented as a list, where the tail is given by the joint vector at time $t = 0$ and the head is the joint vector at time $t = 249$ (see right side of Fig. 1). In the second representation, the state of each joint at time t is collected in a tree with out-degree 4. In this tree, each joint is associated to a node, and the topology of the tree follows the topology of the kinematic tree. Thus, referring to the left side of Fig. 1, joint 0 is the root of the tree, and it has four children: the last child is the root of the tree at time step $t - 1$, i.e., the joint 0 at time step $t - 1$; the other (sorted) children are joints $4, 7, 1$, respectively. Then joint 4 has just one child, the first, which is joint 5, which in turn has a single (first) child, i.e. joint 6 (which is a leaf), and so on as shown in Fig. 2. Notice that in this representation, the first (third) child may

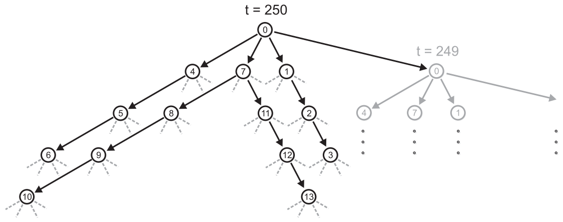

Fig. 2: Data representation as tree with out-degree 4 (details see text).

either represent a joint from the left (right) arm or a joint from the left (right) leg. In the third representation, we have avoided that: there is a child for each left (right) arm and a child for each left (right) leg. This can be obtained by using a tree with out-degree 6. In this last representation, joint values for left (right) arm are not confused with joint values for left (right) leg.

SOM-SD preparation. To evaluate the capacity of the SOM-SD and compare the different encodings, we train for each encoding maps of different sizes from 20×10, 24×12, . . . , 60×30. Map initialization differs for the weights representing node label (a 14-dim vector for the sequence encoding, a scalar for the tree-encodings), where values are uniformly randomized between the maximum and the minimum value of the respective component in the training data. Weights referring to context are initialized to uniformly in $[-1, 0]$. We also compare different weightings of the labels against the context, i.e. of μ_1 vs. μ_2, and – as a non-contextual baseline for the quantization error – a plain SOM with $\mu_1 = 1, \mu_2 = 0$. Automatic values for μ's obtained by computing statistics on the labels, assuming uniform distribution over the pointers, and equalizing the contributions are also considered and showed the best performance. Note that these automatically determined values (always summing to 1) lead to different μ's for each size of the map in Fig. 3 and therefore only the lower bounds are given.

5 Results and discussion

In Fig. 3 it can clearly be seen that for all encodings the quatization errors, i.e. the degree to which the sequence of winner approximates the true sequence, are satisfactory though clearly the tree encodings perform much better. Note that this is a non-trival result, as before only one-dimensional sequences have been investigated to some depth and there have been no results for trees, which use links as well for encoding time order as structure in the data at each timestep. It also seems that there is an optimal moderate map-size, which allows for the best compromize between context and label encoding, in our case this are the 28x14 networks. Control experiments with plain SOM show that the best quantization error for SOM is approximately the same for the same map-size and thus one outcome is that it does not disturb quantization performance to add structure if SOM-SD is used, while gaining the possibility to distinguish identical labels

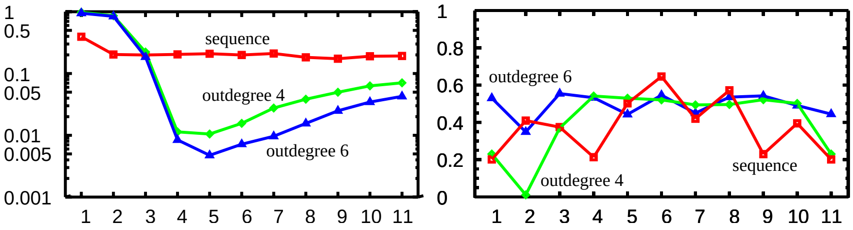

Fig. 3: Left:Best quantization errors E_{som-sd} (in log scale) in function of map sizes for the three studied representations: sequences (in red, $\mu_1 = 0.9$), tree with out-degree equal to 4 (in green, $\mu_1 > 0.98$), tree with out-degree equal to 6 (in blue, $\mu_1 > 0.99$). The x-axis refers to map sizes according to the law $(2x+20)\times(10+x)$, i.e. for $x = 2$ the map size 24×12 is referred to. Results are averaged over 10 different initialized maps. Right: Correlation between locality of the graph and number of distinct winners measured as in Eq. 1.

with respect to the context in the tree. The right plot shows that there is a clear correlation between the number of the winners used in the map and the degree of locality in the tree, which means that time is effectively compressed. In the experiment that is: if the stick-figure stays close to the original position or temporarily in the same region, then only very few winner nodes are needed. Further inspection of the map showed that qualitative similar trajectories end in similar regions of the map, as has to be expected of a SOM type algorithm. Concluding, we believe that SOM-SD shows a high potential for unsupervised clustering of complex trajectories like they soon will be generated in evaluating videos and observing people. Our future work will therefore be directed towards utilizing SOM-SD as a preprocessing framework for trajectory classification and action recognition in such contexts and to further evaluate the time and structure compression abilities in a more quantitative way.

Acknowledgment

We would like to thank M. Hagenbuchner for providing us with an implementation of the SOM-SD as described in more detail in [5].

References

[1] C. Chang and R. Ansari. Kernel particle filter for visual tracking. *IEEE Signal Processing Letters*, 12(3):242–245, 2005.

[2] H. Sidenbladh. *Probabilistic Tracking and Reconstruction of 3D Human Motion in Monocular Video Sequences.* PhD thesis, KTH Sweden, 2001.

[3] T. Voegtlin. Recursive self-organizing maps. *Neural Networks*, 15:979–992, 2002.

[4] M. Hagenbuchner, A. Sperduti, and A.C. Tsoi. A self-organizing map for adaptive processing of structured data. *IEEE Trans. on Neural Networks*, 14(3):491–505, May 2003.

[5] M. Hagenbuchner and A.C. Tsoi. A supervised self-organizing map for structures. In *IJCNN*, 2004.