ROAD MESH MODELLING USING THE SPATIOTEMPORAL CLUSTERIZATION

Rudolf Marek, Miroslav Skrbek

Czech Technical University in Prague,
Faculty of Electrical Engineering,
Department of Computer Science and Engineering
Karlovo namesti 13
121 35 Prague, Czech Republic

marekr2@fel.cvut.cz(Rudolf Marek)

Abstract

The GPS navigation is widely used aid for travelers. However, good navigation depends on good maps, which are sometimes hard to get. In this paper we explore a method to model a road mesh using self-organizing spatiotemporal data clustering of collected GPS data. The resulting road mesh model is obtained from simulated self organizing neural network, which clusters the individual data vectors and infers the time dependencies between the clusters. This allows to detect one way roads, or slow traffic automatically. To achive this goal we model the road mesh with the Temporal Hebbian Self-organizing map (THSOM). This paper presents a novel training method for the simulated THSOM neural network, which reduces training period and improves model the convergence of original THSOM neural network. The road mesh model is obtained from real GPS data as well as from simulated data set.

Keywords: Neural Networks, Self-Organizing Maps, Computational Intelligence

Presenting Author's Biography

Rudolf Marek is a postgraduate student and researcher at the Department of Computer Science and Engineering. He got his Master's degree in Electronics and Computer Science and Engineering from Faculty of Electrical Engineering Czech Technical University in Prague in 2006. His research is focused on hardware acceleration of computational intelligence algorithms.

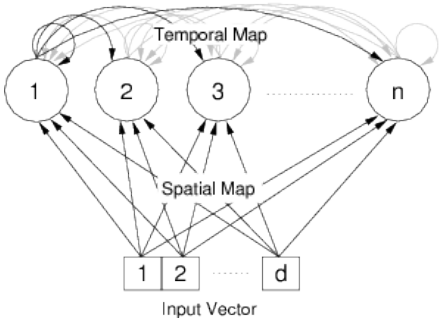

Fig. 1 THSOM network architecture.

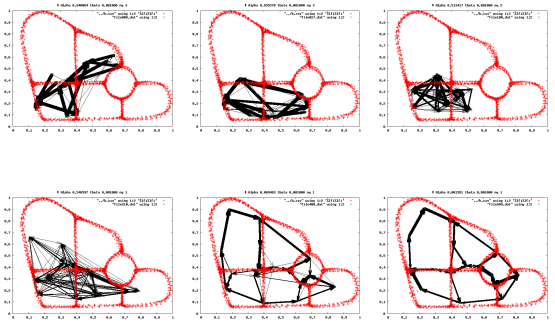

Fig. 2 Original THSOM algorithm after 8, 27, 100 (first row), 210, 400, 999 (second row) epochs. At the beginning convergence is poor and the algorithm seems not to fit and follow the data. After 200 epochs the temporal map starts to fit the roads well and data vectors begin to get covered, the spatial map starts to cover most of the data vectors. After epoch 500, oscillations emerges which tries to refit some neurons in roundabout. The spatial and temporal map gets only minor adjustments

1 Introduction

The GPS car navigation systems are quite common accessory for travelers, however the navigation is as good as the map provided by the GPS system vendor. In this paper, we model a road mesh from a simulated data set and also from raw GPS data recorded periodically during a car ride. The model is constructed by means of self organization which arises during the simulation of the Temporal Hebbian Self-organizing Map (THSOM) [1] neural network.

Unlike classic SOM [2], THSOM contains spatial and temporal maps, extending the clustering to space and time. The THSOM architecture consists of single layer of hybrid neurons as depicted in Fig. 1. Neurons of this layer are fully connected to the input vector of dimension d, connections make up the spatial map. The neurons are connected to each other in the grid using recurrent temporal synapses (*temporal map*). Hybrid neurons contain two types of similarity measures, Euclidean metric for measuring similarities in the input spatial space and the scalar product for measuring similarities in the temporal space. The activation (output) of a neuron is defined in [3] is as follows:

$$y_i^{t+1} = \gamma \left(1 - \sqrt{\frac{\sum_{j=1}^{D}(x_j^t - w_{ij}^t)^2}{D}} \right) \quad (1)$$

$$+ \quad (1-\gamma)\frac{\sum_{k=1}^{n}(y_k^t m_{ik}^t)}{n} \quad (2)$$

where y_i^{t+1} is activation (output) of i-th neuron in time $t+1$ (next time step), D is a dimension of input vector, x_j^t is input of j-th vector in time t, w_{ij}^t is the spatial weight for j-th input in time t, y_k^t is output of k-th neuron in time t, m_{ik}^t is temporal weights for k-th neuron in time t (from neuron k to neuron i) and n is number of of neurons in network.

For $\gamma \to 1$, the THSOM reduces to SOM network. For $\gamma \to 0$ only prediction is done based on the temporal weights.

2 Training algorithm

The original training algorithm of the THSOM is applied individually to each input pattern.

For each data input in epoch:

- Find BMU
- Update spatial weights according to classical Kohonen SOM training rule
- Update the temporal weights

The spatial weights are updated using a SOM (Kohonen's Self-organizing Map) rules. The weights are updated in order to move the neuron in the space closer to the input vector. Temporal weights use modified Hebbian training rule according to the Eq. (3).

We propose new training algorithm which changes the view to the data. Instead of modifing the network state after each input, the data is presented to the network in a batches, similarly to BatchSOM[4]. The spatial and temporal training is separated. This new algorithm is referenced as BatchTHSOM in further text.

For each epoch:

- Apply the BatchSOM[4] training algorithm on data batch (y_i^{t+1} computed with $\gamma \to 1$)
- Compute BMUs (y_i^{t+1} computed with $\gamma \to 0.5$) for individual vectors, if old and new BMU differs then update the temporal weights

The first step, well-known BatchSOM algorithm is applied to spatial map only. The $\gamma \to 1$ reduces the network to the SOM. Second step, which is used to train

the temporal map using Hebbian training combines influence of both maps $\gamma \to 0.5$.

The rules for the temporal weights updates are following:

$$m_{bk}^{t+1} = \begin{cases} \min(\max(m_{bk}^t + \alpha(1 - m_{bk}^t + \beta), K_l), K_h), \\ \text{for } k = \text{argmax}_k(y_k^t) \\ \\ \min(\max(m_{bk}^t - \alpha(m_{bk}^t + \beta), K_l), K_h), \\ \text{otherwise} \end{cases} \tag{3}$$

$$\alpha = 1 - e^{\frac{-2 * \text{epoch}}{\text{epoch}_{\max}}} \tag{4}$$

The coeficients β and α control temporal synapses training rate. The α starts on 0 and is increased in time up to 1 to streghten the temporal map influence during the training process. The β is fixed on the value 0.0001. The K_l and $K_h = 1$ are the weight boundaries (usually 0.0 - 1.0).

The temporal weights are initialized to small 0.00001 on start of the training phase. The recurrent self temporal weight is always zeroed. The temporal weights are reset to 0.0 on the each epoch start.

During the SOM traing phase, it may happen that some neurons never win for any input vector. This happened mostly if there were too many neurons employed. Those neurons will cause arithmetic underflow because their new position is based on their performance to match input data. This situation is detected and all orphaned neurons are re-fit in the the data space to randomly chosen input vector (weights matches directly the input vector).

3 THSOM Simulation

The reference BatchTHSOM simulation was written in C language. If it is compared to the simulation with original non-parallel THSOM learning algorithm[5] it is even faster. For the net with 64 neurons the original algorithm took 23 seconds to complete (1000 epochs) now it takes 18 seconds with same compiler settings and on same PC. It could be explained with the fact that batch processing needs slightly less branching and decision making than original algorithm. Moreover we now need 10 times less epochs so it is simulated considerably faster anyway.

4 The GPS data

Two datasets were used for new algorithm evaluation. First an artificial dataset containing a data from a simulated ride made up using tablet as data source (2217 datapoints). Second batch set was real GPS data from a ride of few blocks of streets (788 datapoints). Both datasets were normalized to match (0.0 - 1.0) float coordinates.

The following figures depicts position of the individual neurons and their time relationships inferred during the neural network learning stage. The thickness of the line corresponds to the strongness of the the time linkage between neurons. In our case, if the neurons A has a linkage to B and not vice versa, it means that the cars passed as one way.

4.1 Original THSOM algorithm

The original THSOM algorithm was evaluated in the past on artifical GPS data. The outstanding difficulty was a poor convergence on the data set. It took 1000 epochs and even not all neurons were properly placed. An animation containing the network state after each epoch was created.

As depicted in Fig. 2 it took quite a lot of epochs for a network to match up the input data with proper spatial and temporal weights. Often the network just oscillated long time in more or less same state (check epoch 8 and 100). Also, the fit of neurons to the roads were not ideal.

It served as a motivation to improve the training algorithm of the THSOM neural network.

4.2 Performance of the improved THSOM algorithm

The performace of our BatchTHSOM algorithm was evaluated on the same data set. The number of epochs was reduced to 100, because it converges much more quickly. In fact it the experiments showed that even 10 or 20 is enough. The reason for this is the Eq. (4) which just depends on number of epochs. If the change of this coeficient is steep, the temporal weights get learn more quickly (neurons slides to the points much faster).

The Fig. 3 depicts the developement of the THSOM neural network over the time. The BatchSOM places the neurons close to the center of the input vector domain. Then the neurons starts spreading to fit onto the data. The temporal connections between the neurons starts to develop early too.

The one way roads were inferred correctly. It is nearly done after 42 epochs. Same experiment was peformed with higher number of neurons as depicted in Fig. 4. The roads were detected and aproximated with all neurons correctly.

The real GPS data sets performed also well. There was 100 epochs. If there was not enough neurons to cover the roads the network approximated them a bit as depicted in Fig. 5.

The real GPS data set with enough neurons covered whole with 64 neurons depicted in Fig. 6. The one way roads were inferred correctly.

5 Conclusion

We introduced new learning algorithm for simulation of THSOM neural network. The algorithm processes data in batches. As the first step, well-known BatchSOM algorithm is applied to spatial map only. Second step, which is used to train the temporal map using Hebbian training combines influence of both maps.

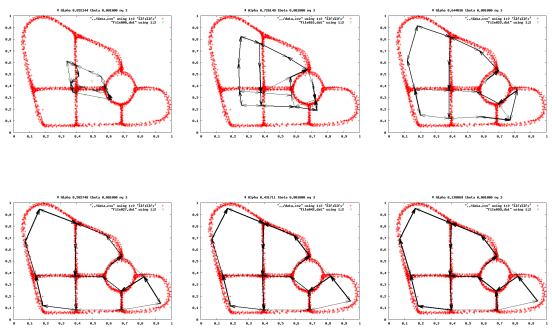

Fig. 3 The progess of BatchTHSOM training algorithm on artificial data during epoch 8,16,22 (first row), 27,42,99 (second row) with 16 neurons total. The road mesh starts to build from vector avg value. On epoch 22 the roundabout starts to emerge. From epoch 27 to 99 not much happens, only the temporal weights got adjusted, no slight move of neurons in spatial map.

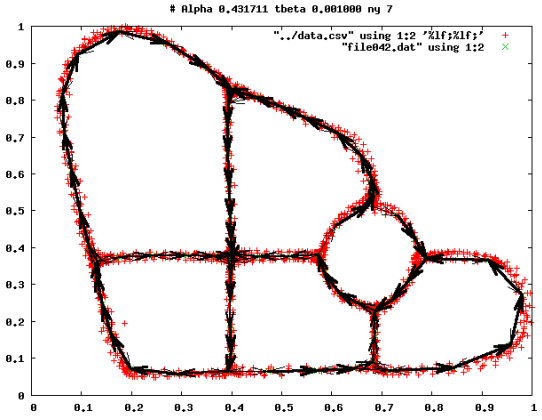

Fig. 4 The progess of BatchTHSOM training algorithm during epoch 42 with 64 neurons total (artificial data). The road mesh is correctly modelled, one way roads correctly infered. The roundabout is modelled well too.

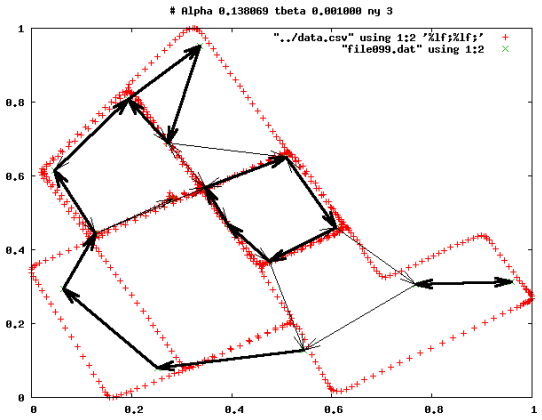

Fig. 5 The trained (100 epochs) BatchTHSOM network with 16 neurons on real GPS data set, the number of neurons is too low, some roads we approximated or dragged to places with more data.

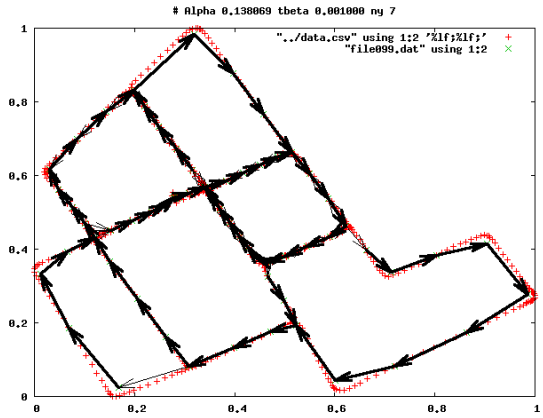

Fig. 6 The trained (100 epochs) BatchTHSOM network with 16 neurons on real GPS data set, all one way roads correctly inferred, normal roads detected too, although they were driven less in both directions.

The models inferred from the simulation of THSOM network were obtained faster, with less number of epochs (10 times). The models converged smoothly inferring correctly the shapes and directions of different roads.

6 Acknowledgments

This work has been supported by the research program "Transdisciplinary Research in the Area of Biomedical Engineering II" (MSM6840770012) sponsored by the Ministry of Education, Youth and Sports of the Czech Republic.

7 References

[1] J. Koutnik and M. Snorek. Temporal hebbian self-organizing map for sequences. In *18th International Conference on Artificial Neural Networks Proceedings (ICANN 2008)*, volume 5163, pages 639–648. Springer Berlin / Heidelberg, 2008.

[2] T. Kohonen, M. R. Schroeder, and T. S. Huang, editors. *Self-Organizing Maps*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.

[3] J. Koutnik. *Self-organizing Maps for Modeling and Recognition of Temporal Sequences*. A thesis submitted to Faculty of Electrical Engineering, Czech Technical University in Prague., 2008.

[4] Juha Vesanto. Neural network tool for data mining: Som toolbox, 2000.

[5] M. Skrbek R. Marek. Hardware acceleration for computational intelligence - thsom neural network on x86 hardware. In *European Simulation and Modelling Conference 2008*, pages 327–331. EUROSIS, 2008.