Gesture Classification with Hierarchically Structured Recurrent Self-Organizing Maps

Volker Baier
Neuro Cognitive Psychology,
Universität München
Leoplodstr. 13, 80802 Munich, Germany
baierv@cs.tum.edu

Lorenz Mösenlechner
Institut für Informatik,
Technische Universität München
Boltzmannstrasse 3, 85748 Garching b. München, Germany
moesenle@cs.tum.edu

Matthias Kranz
Institut für Betriebssysteme und Rechnerverbund,
TU Braunschweig
Mühlenpfordtstr. 23, 38106 Braunschweig, Germany
matthias@ibr.cs.tu-bs.de

***Abstract*—New input devices need clever algorithms to process input information. We constructed a hierarchically structured neural network assembly based on recurrent self-organizing maps which is able to process and to classify motion data. We derived motion data using a so called Gesture Cube [1], a cubic tangible user interface developed for one-handed control of media appliances in a home environment. This previously recorded data was automatically pre-processed by our biologically inspired neural network and classified by a improved k-nearest neighborhood classifier. In this paper we shortly describe the platform used for data acquisition but focus on the novel algorithms used for classification.**

***Index Terms*—Gesture Cube, Sequence Processing, Sequence Classification, Multi Layer Neural Networks**

I. Introduction

Dealing with continuous streamed motion data is a quite difficult task for neural networks. We will introduce an assembly which is able to sub-sample the stream on a temporal basis in a self driven manner. The sub-sampling network structures act as temporal filter combining activation events of single neurons to static activation patterns. These patterns represent the input of the classification algorithm.

II. Related Work

Cubic user interfaces have attracted many researchers in the field of pervasive computing. The compelling properties and implications of cubes have been discussed by Sheridan [2] and Terrengi and Kranz [3]. The system used for data acquisition has been developed by Freund et al. [1]. The Gesture Cube is based on the same wireless sensor node platform, as the Display Cube of Kranz et al. [4], [3], employing the Particle Computer Platform [5], [6], a common sensor network platform. The displays have been removed and the housing was changed to a more compelling form factor. This allows the usage of a well designed user interface as input to other systems which can be manipulated with only one hand due to its size. We therefore will not discuss the platform in greater detail but will focus on the algorithm and the classification of the previously recorded sensor data.

Early systems discussed the cube as tangible user interface for input (e.g. Block et al. [7]) and combined I/O (e.g Laerhoven et al. [8]).

The importance of good classification of acceleration sensor data for context derivation has been shown, for example, by Laerhoven et al. [9], [10]. Another example of a cubic user interface is Laerhoven's [11] Fair Dice. It is an example of sensory augmentation of an object that perceives rolls and records what face it lands on. Neural networks are used for classification here.

To distinguish our work from other classification systems using acceleration sensor data, our data intentionally was not tuned: The sensor data we acquired has been sampled at a much lower frequency and with high noise on the sensor data - absolutely no preprocessing has been done on the local sensor node, nor has the measurement range of the ADXL acceleration sensor been adjusted to actual measurement range. This makes classification and sense-making of sensor data, e.g. for context-aware applications, much more challenging. Thus, this setup is much more real-world like and the algorithm presented in this work therefore can be applied to a greater range of scenarios.

III. Gesture Cube

The data sets we used for classification were gathered with the above introduced Gesture Cube, containing three orthogonal acceleration axis (two perpendicular mounted two-axes acceleration sensors from Analog device). Fig. 1 shows the cube and the embedded sensor node.

The sensor readings are transfered via a wireless RF communication channel in near-real time to a PC via a so-called XBridge (RF-to-UDP bridge). The limitations of the embedded PIC microprocessor is unsuitable for preprocessing, therefore this process cannot be dealt with on the sensor node platform. Also, the timing of the communication slots [6] interrupts the micro-controller during the data transmission process. Thus, some sensor readings may be lost already during data sampling and acquisition. In our work, we explicitly exploit the shortcoming in processing power and transmission bandwidth and feed our neural network with this noisy and erroneous sensor data.

The sensor data we use as input for the classification system was captured from a user who performed gestures with the wooden cube in his hand. By performing gestures, the user controlled a media player [1].

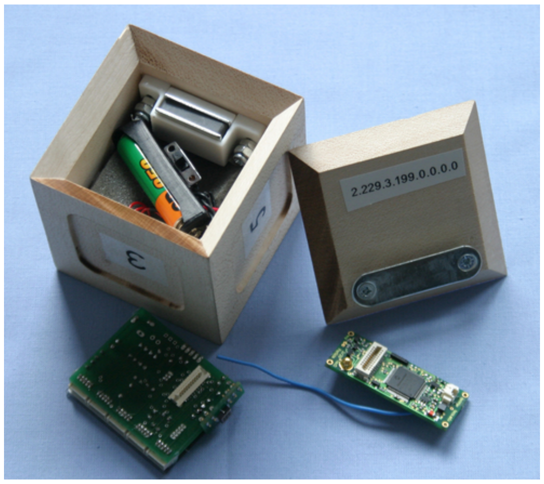

Figure 1. The Gesture Cube is a small wooden cube just big enough to contain a wireless sensor system. The system is networked with the environment with a unreliable RF communication channel. The data is noisy and sampled at a very low rate. Additionally, data can get lost on the RF channel - no securing against data loss has been implemented or used.

For training purpose we combined all gestures in a single data file plotted in Fig. 2.

IV. Neural Network

A. Single Processing Layers

To understand the network behavior it is necessary to provide a brief introduction to the well established model of Kohonens Self-Organizing Maps (SOM).

SOMs can be defined as a two dimensional mesh-grid of neurons. Each neuron holds a weight vector w_i and its position on the mesh r_i.

Training is done in an unsupervised scheme. Initially, all weight vectors are randomly occupied. After providing the network with an input value, we calculate the distance between input value $x(n)$ and weight vector of each neuron y_i. The best matching neuron (often called best matching unit BMU) is the one with the smallest euclidian distance to the input vector and denominated with y_b.

$$y_i(n) = ||x(n) - w_i(n)||_p \quad (1)$$
$$y_b(n) = \min ||y_i(n)|| \quad (2)$$

The BMU is then adapted with its neighboring neurons towards the input value. The neighborhood is determined by a neighborhood function:

$$N_{ib}(n) = \exp\left(-\frac{||r_i - r_b(n)||_2^2}{2\sigma(n)^2}\right) \quad (3)$$

$$w_i(n+1) = w_i(n)\gamma(n)N_{ib}(n)((x(n) - w_i(n)) \quad (4)$$

The width of the neighborhood is a decreasing function of time as well as the learning rate $\gamma(n)$.

To enable this algorithm to process temporal sequence information like the Gesture Cube motion data, we extended the standard formulation of the neuron with respect to a temporal recurrent activation handling. An existing formulation, namely RSOM [12], used a feedback of the activation back to the input. Since we want that the temporal ordering of activation events is preserved along the whole processing chain, we introduce a nonlinear mapping of the excitation of the BMU to a standardized activation value of 1 and therefore called the algorithm Strict Temporally Ordered Recurrent Map (STORM). If we take the activation strength as the time reference, the euclidian distance calculation in the RSOM formulation is responsible for the problems in the temporal ordering. Additionally we introduced a quite biological limitation, namely a period within a once activated neuron is not able to be activated. This limitation is the so called refractory period.

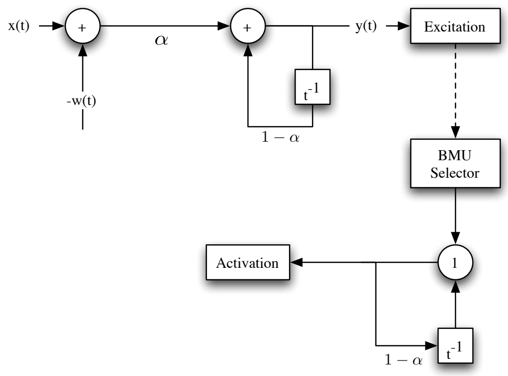

Figure 3. STORM Neuron with separated excitation and activation

The excitation of a recurrent neuron is a combination of the damped output of the last time step, with the current output:

$$y_i(t) = (1-\alpha)y_i(t-1) + \alpha(x(t) - w_i(t)) \quad (5)$$

Based on the excitation we calculate the activation of the neuron with:

$$z_b(t) = 1.0 \quad (6)$$
$$z_{i\neq b}(t) = z_{i\neq b}(t-1)(1-\alpha)$$

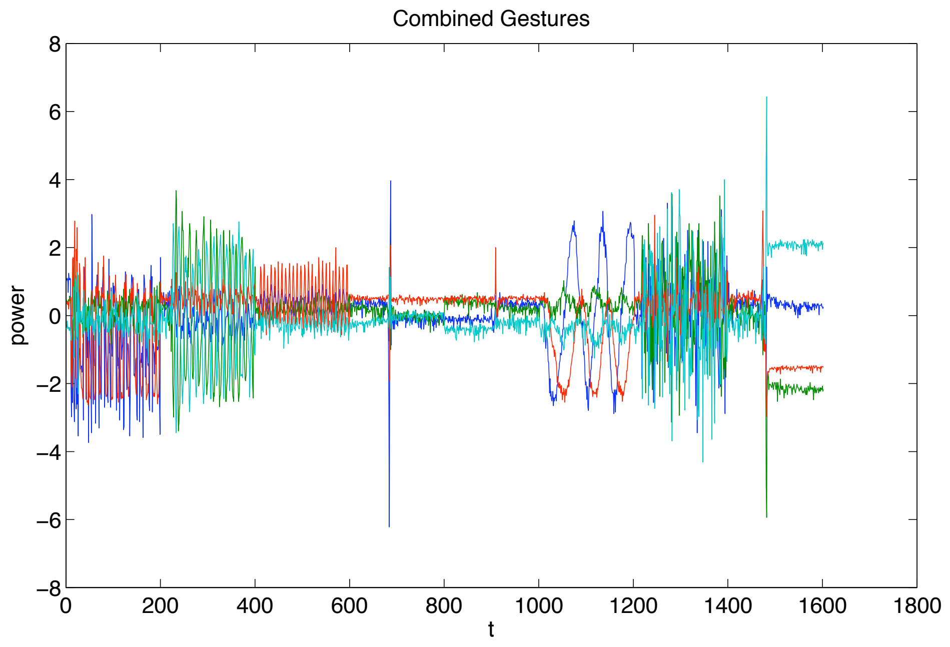

Figure 2. Combined Gestures Plot

By applying an unified activation value to the first non refractory neuron, we establish a nonlinear projection from the input to the output of the neuron. Calculating the activation in this way ensures a strict temporally ordering of the *firing* state of the neurons. The result of this scheme is an activation pattern of all neurons whereas the level of activation also codes it temporal occurrence in the past.

The period, during which a neuron is stated refractory is determined with the system response and the significance threshold θ_s.

$$k_\theta = min_k|\theta_s \geq \alpha(1-\alpha)^k \tag{7}$$

We also tried to add some extra time steps to the refractory period. These extra time steps resulted in a better unfolding of the network and a better representation of the input data on the network. The trials described below where gathered with two thirds extra time steps:

$$k_\Theta = k_\theta + k_\theta \cdot 2/3 \tag{8}$$

This variant of STORM is called inhibited STORM (iSTORM) and is used as one type of the used structures in the neural network assembly.

B. The Neural Network Assembly

Based on the single processing layers introduced above we constructed a multi-layer network. This compound is a derivation of the spatio temporal Strict Temporally Ordered Map (stSTORM) [13] lacking the prediction instances.

The assembly consisted of three iSTORM layers consisting of 20×20 neurons filtering the input information to get a temporally more coarse representation of the input data stream. Each layer collects several activation events before it feeds its activation state, a single vector consisting of the activation values of all neurons, to the next higher processing layer. This scheme applies for the three first processing layer.

The fourth layer is a classical SOM layer (10×10 neurons) building a static time space representation of the last seen activation. The fifth layer sums up the last seen activation events of the fourth layer establishing an activation density mapping.

Finally, we fed the information of the fifth layer into a k-nearest neighbor classification algorithm.

C. The k-Nearest Neighborhood Classificatory

For classification we used the k-nearest neighborhood algorithm. For each (unknown) data vector to be classified, the distances to all learned data vectors are calculated. The class of the input data vector is then determined by a majority decision of the k closest vector's classes.

In contrast to the original formulation of the k-nearest neighborhood algorithm, we used a distance measurement derived from the Kullback-Leibler-Divergence (9).

$$D(v^{(1)}, v^{(2)}) = \sum_i v_i^{(1)} \log \frac{v_i^{(1)}+1}{v_i^{(2)}+1} \tag{9}$$

This distance measurement resulted in a classification result gain of 5 percent compared to the euclidian distance.

V. Results

We tested our neural network assembly on a data set consisting of the acceleration values of ten gestures executed with the Gesture Cube. For each gesture we had only ten samples, consisting of about 200 samples. The ten samples for each gesture varied in frequency and magnitude, which forced the network to generalize in order to achieve good classification results.

The data was preprocessed by shifting and scaling the values to a range between around $[-1, 1]$. This was simply done to have better control of the random initialization of the weight vectors. The initialization was chosen to be normal distributed with mean 0 and a variance of 1.

We used a feedback factor $\alpha = 0.3$ leading to memory of 8 time steps and a refractory period of 11 time steps in total.

The activation pattern of layer 1 was passed to layer 2 after five time steps ($\Delta t_1 = 5$). The subsequent layers collected information for 3 time steps each ($\Delta t_2 = 3$).

Our results were gathered with a 5 fold cross-validation using a 5-nearest neighborhood classificator as described above. The performance of our classification algorithm is illustrated in the confusion matrix in table I. The overall classification rate was 80%.

Table I
CONFUSION MATRIX WITH 5-NEAREST-NEIGHBORHOOD AND 5-FOLD-CROSSVALIDATION

	1	2	3	4	5	6	7	8	9	10
1	**9**	0	0	0	0	1	0	0	0	0
2	0	**9**	1	0	0	0	0	0	0	0
3	0	0	**10**	0	0	0	0	0	0	0
4	0	0	3	**7**	0	0	0	0	0	0
5	0	0	0	0	**10**	0	0	0	0	0
6	1	0	2	0	0	**5**	0	0	2	0
7	0	0	2	1	3	0	**4**	0	0	0
8	0	0	0	0	0	0	0	**9**	0	1
9	0	0	0	0	0	0	0	0	**10**	0
10	0	2	0	0	1	0	0	0	0	**7**

The mapping of the class numbers and gestures is shown in table II. We cross checked several classification algorithms

Table II
MAPPING OF GESTURES TO CLASS NUMBERS.

Class	**Gesture name**
1	Shake up down
2	Shake right left
3	Shift backward
4	Shift up
5	Spin forward
6	Spin horizontal
7	Throw catch
8	Turn forward
9	Turn right
10	Wiggling forward backward

like support vector machines and RBF networks for the classification instance in our assembly. Even the support vector machine had a slightly worse classification result (76%) than the k-nn classificator with KL divergence. The RBF network achieved 73%.

VI. CONCLUSIONS AND DISCUSSION

With our work we showed that an algorithm developed to explain biological phenomena within the visual processing apparatus of the human brain can be successfully used for technical pattern processing and classification. Furthermore our approach is highly successful even on small data sets and noisy information without special adaptation for the specific task. The number of training cycles was surprisingly small. Our algorithm showed the excellent classification results after only 90 training cycles.
In future work we will continue to integrate results from fundamental research in neuro cognitive modeling into neural network assemblies addressing real world applications.

VII. ACKNOWLEDGEMENTS

Part of the work was funded by the CoTeSys Cluster of Excellence.

REFERENCES

[1] M. Kranz, S. Freund, P. Holleis, A. Schmidt, and H. Arndt, “Developing gestural input,” *icdcsw*, vol. 0, p. 63, 2006.

[2] J. Sheridan, B. Short, K. V. Laerhoven, N. Villar, and G. Kortuem, “Exploring cube affordance: Towards a classification of non-verbal dynamics of physical interfaces for wearable computing,” in *In Proceedings of the IEE Eurowearable 2003*. IEE Press, 2003, pp. pp 113–118.

[3] L. Terrenghi, M. Kranz, P. Holleis, and A. Schmidt, “A cube to learn: a tangible user interface for the design of a learning appliance,” in *Personal and Ubiquitous Computing*, November 2005, pp. pp. 1–6. [Online]. Available: http://dx.doi.org/10.1007/s00779-005-0025-8

[4] M. Kranz, D. Schmidt, P. Holleis, and A. Schmidt, “A display cube as tangible user interface,” in *In Adjunct Proceedings of the Seventh Internation Conference on Ubiquitous Computing (Demo 22)*, September 2005.

[5] C. Decker, A. Krohn, M. Beigl, and T. Zimmer, “The particle computer system,” in *IPSN Track on Sensor Platform, Tools and Design Methods for Networked Embedded Systems (SPOTS), Proceedings of the ACM/IEEE Fourth International Conference on Information Processing in Sensor Networks*, April 2005.

[6] M. Beigl, A. Krohn, T. Zimmer, C. Decker, and P. Robinson, “AwareCon: Situation Aware Context Communication,” in *Proceedings of Ubicomp 2003*, Seattle, USA, October 2003.

[7] F. Block, A. Schmidt, N. Villar, and H.-W. Gellersen, “Towards a playful user interface for home entertainment systems,” in *European Symposium on Ambient Intelligence (EUSAI 2004), Springer LNCS 3295*. Springer, 2004, pp. pp207–217.

[8] K. V. Laerhoven, N. Villar, A. Schmidt, G. Kortuem, and H. Gellersen, “Using an autonomous cube for basic navigation and input,” in *ICMI ’03: Proceedings of the 5th international conference on Multimodal interfaces*. New York, NY, USA: ACM Press, 2003, pp. 203–210.

[9] K. V. Laerhoven and H.-W. Gellersen, “Spine versus porcupine: A study in distributed wearable activity recognition,” in *ISWC ’04: Proceedings of the Eighth International Symposium on Wearable Computers (ISWC’04)*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 142–149.

[10] K. V. Laerhoven, K. A. Aidoo, and S. Lowette, “Real-time analysis of data from many sensors with neural networks,” in *ISWC ’01: Proceedings of the 5th IEEE International Symposium on Wearable Computers*. Washington, DC, USA: IEEE Computer Society, 2001, p. 115.

[11] K. V. Laerhoven and H.-W. Gellersen, “Fair dice: A tilt and motion-aware cube with a conscience.” in *ICDCS Workshops*. IEEE Computer Society, 2006, p. 66.

[12] T. Koskela, M. Varsta, J. Heikkonen, and K. Kaski, “Temporal sequence processing using recurrent som,” in *KES’98 2nd Int. Conf on Knowledge-Based Intelligent Engineering Systems, Adelaide, Australia*, vol. 1, April 1998, pp. 290–297.

[13] V. Baier, “Motion perception and prediction: A subsymbolic approach,” Electronic, Chair for Theoretical Computer Science and Foundations of Artificial Intelligence, TU München, November 2006.