Real-time Analysis of Data from Many Sensors with Neural Networks

Kristof Van Laerhoven, Kofi A. Aidoo and Steven Lowette
StarlabResearch
Englandstraat555
B-1180Brussels,Belgium
{kristof, kofi}@starlab.net, slowette@vub.ac.be

Abstract

Much research has been conducted that uses sensor-based modules with dedicated software to automatically distinguish the user's situation or context. The best results were obtained when powerful sensors (such as cameras or GPS systems) and/or sensor-specific algorithms (like sound analysis) were applied. A somewhat new approach is to replace the one smart sensor by many simple sensors. We argue that neural networks are ideal algorithms to analyze the data coming from these sensors and describe how we came to one specific algorithm that gives good results, by giving an overview of several requirements. Finally, wearable implementations are given to show the feasibility and benefits of this approach and its implications.

1. Introduction

One of the elements one can always find in definitions of wearable computers is the presence of one or more sensors to act as a second input. As the MIT's wearable computing FAQ states: "In addition to user-inputs, a wearable should have sensors for the physical environment. Such sensors might include wireless communications, GPS, cameras, or microphones." [16]. This approach is in the fields of human-computer interaction, ubiquitous computing and wearable computing also known as context awareness [3].

Context awareness deals with granting a device sensors so that it can autonomously detect what state (internal or external) the user is in. This would provide a valuable service to the device, which it can use in taking certain decisions without any user interaction. There are two distinct methods for distinguishing the context, depending on the amount and complexity of the sensors.

The first approach is a well-established one where one sensor (or a set of a few sensors) is combined with an algorithm that is usually specific to that particular sensor. Research on wearable cameras with computer vision algorithms [17,20], microphones with sound-specific preprocessing [2], or beacon-based systems [12] for example give excellent results.

The alternative approach is based on a large number of small and simple sensors. This direction was taken in research at MERL [8], Philips [5], and in the TEA project [4,19], but is still rather new and unexplored. The combination of many simple sensors that individually give information on just a small aspect of the context, results in a total picture that might be richer than the one-smart-sensor approach. The distinction between both approaches is depicted in Figure 1.

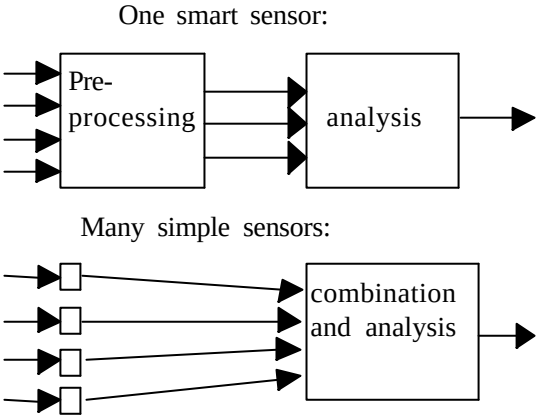

Figure 1. Diagram of the two approaches.

Most of the benefits of the latter approach were mentioned in previous research (see for instance [8]):

- **Cheap.** The small simple sensors require generally less resources and cost less than for instance cameras and GPS systems. An extremely large amount of simple sensors could of course invalidate this.
- **Robust.** Since the sensors we use are small, they can smoothly be distributed over a larger area which makes the sensing system less prone to errors. In case a sensor gets blocked or damaged, other sensors will still capture context-relevant information due to the redundancy in the sensors.

- **Distributed.** The size also allows the sensors to be integrated into clothing much easier.
- **Flexible.** The richness and complexity of the identifiable contexts is directly linked to the amount, position and kind of sensors. Adding, moving, or improving sensors hence increases the performance of the system.

The real bottleneck in this method is the software algorithm that has to combine and analyze all the data. This paper defines the choices one has to make in finding a suitable algorithm and clarifies our choices.

The remainder of the paper will be organized as a step-by-step search for a suitable algorithm. After elaborating on the many-simple-sensors approach and giving a concrete hardware example, we will motivate that a neural network approach will be a logical next step to analyze the incoming data. A comparison study of the most likely candidate algorithms will be given motivating the choice for the self-organizing map. Some shortcomings of this algorithm will be discussed, leading to a modification of the self-organized map. Finally some applications illustrate the implications of using such an algorithm.

2. Combining many simple sensors

2.1. What are simple sensors?

It is very hard to construct a formal definition to describe a simple sensor. One can say that it needs to be small, cheap and yet give a value that could allow distinguishing different contexts easily without expensive processing power. To be more specific we give the details of a sensorboard that we build during the TEA project (Technology for Enabling Awareness, see [4] and [19]).

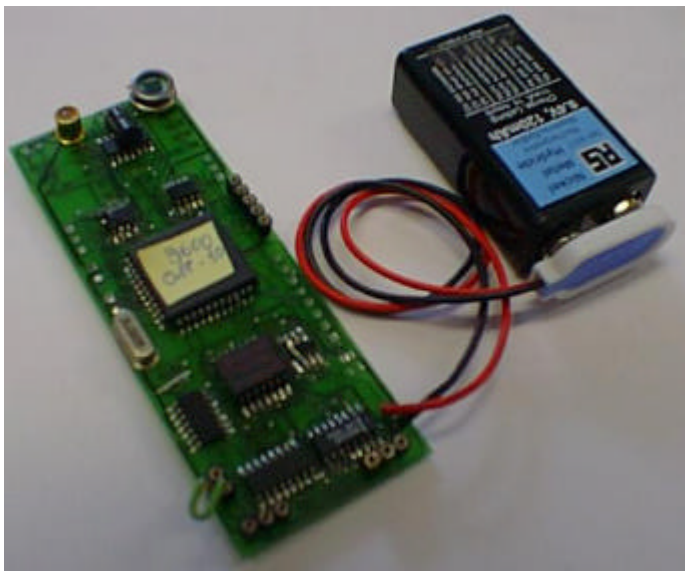

Figure 2. The TEA2 sensorboard with 8 integrated hardware sensors and 2 communication ports.

The TEA2 sensor board (Figure 2) was used much during our experiments and in our wearable system. This sensorboard has been tailored to slip into the back of an enlarged battery casing which fits into the Nokia 6110-6150 mobile phones series. The dimensions of the board are approximately 85mm x 35mm x 0.6mm. It is essentially a sensor board stuffed with 8 sensors, 3 of them doubly redundant, PIC micro-controller and serial communication. It has been designed with low power IC's to receive power by a mobile phone battery outputting 3.6 volts, or a standard 9 volts battery. The board has two communication slots: one standard serial RS232 port to make communication possible with a (wearable) computer or PDA and one Nokia FBUS port for communication with a Mobile Phone.

The sensors on the board are comprised of two photodiodes, two microphones, a dual axis accelerometer, a digital temperature sensor and a touch sensor. The microphones are miniature electrolet capsules regularly used in mobile phone applications. The accelerometer IC is the ADXL202 from Analog Devices and the digital temperature is the Dallas Semiconductor DS1820. These signals are then fed either through direct digital inputs or analog lines into a Microchip PIC16F877 microprocessor, which runs at 20MHz. Apart from these sensors, additional slots for other sensors are available as well.

2.2. Algorithm requirements and characteristics

Apart from having many simple sensors, other requirements were also taken into account, mostly to make the final system more usable:

On-the-spot training. Neural network and machine learning approaches are not new in context aware applications. However, the training phase, where the algorithm actually learns the new contexts, are usually governed by the designer of the application. Giving the user the chance to train the system for new contexts would result in a flexible solution.

Cluster based learning. Every adaptive system tries to store information to distinguish the data it is learning better and better. As opposed to *boundary-based clustering*, where the stored information is an approximation of the boundaries between the learned instances, *cluster-based learning* approximates the data itself. In the latter it is considerably easier to add new contexts afterwards, so this is a favored approach for our needs.

Minimal amount of pre-processing. The strength of the system is based on the redundancy and diversity coming from the large number of sensors. As a consequence, it is feasible to reduce the pre-processing per sensor. In our approach we restrict ourselves even to using the same

four features for every sensor: the average, standard deviation, minimum and maximum over a sliding window. There is thus no need for build-in knowledge of which sensors the incoming values belong to, making the whole system very flexible. Sensors can be added, removed or replaced without replacing the algorithm.

Overlapping contexts. It should be possible to give multiple labels to the same context, such that more than one context can be active at the same time. This requirement favors clustering-based methods.

Outputs are probabilities per context. The output of the algorithm is a list of all learned context plus their probability of appearing. This is a necessity in order to be able to handle overlapping contexts.

2.3. (Artificial) Neural Networks

Neural networks are a group of algorithms that have typically a lot of small and simple, interconnected components (or neurons). This networking enables the entire algorithm to perform much more powerful computations by combining the limited processing power of the separate components. This fits perfectly into the multiple-simple-sensors approach, but there is another reason to choose for neural networks as well: neural networks generally have a better track record on noisy data than statistical methods or expert systems.

3. Finding the right neural network

Neural network research has done a lot of work on the real-time categorization of data coming from many sensors. Especially in the related field of robotics, where autonomous robots wander around and try to learn a representative model of their environment, very similar research has been going on for many years (see for example [18]). The major difference is that in the robotics' case, the problem is not restricted to modeling the environment, further linking the model to actuators or actions is necessary as well. We will adopt this fields' terminology to make our goals more concrete.

3.1. Further neural network specifications

Unsupervised training. The most popular neural networks like the (multilayer) perceptron or networks using radial basis functions are supervised, i.e. learning is only possible during training. Since the user of the wearable computer should not spend too much time in training it, and since it makes more sense to exploit the long-term usage of the wearable, unsupervised learning is favored. While the user, with his wearable, goes from context to context, the algorithm should learn autonomously what kind of input is common and how these inputs relate to each other.

Topology preservation. Safeguarding the relation between inputs in the clustering space is known as topology preservation. This means that inputs from similar contexts will be mapped closely to each other onto the neural network and those of different contexts will be mapped further away from each other. Figure 3 shows an example of the output of a topology preserving clustering algorithm (the Kohonen self-organizing map). This property makes it easier to inspect, debug, and visualize the state of the algorithm and its performance.

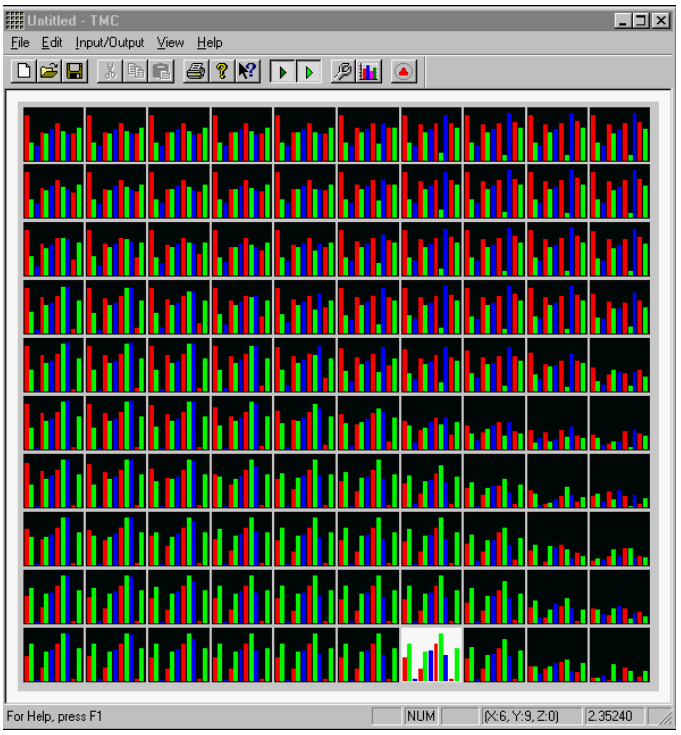

Figure 3. Screenshot of a two-dimensional map of a topology preserving algorithm working on sensordata. The bar-graph in each cell depicts a stored prototype for the input, it is clear that neighboring cells are more similar.

On-line adaptation. Each time the neural network receives a new input, it should recalculate its internal representation of the contexts. We have found no justification for using batch-line training, which waits with adaptation until more input vectors have passed, since it requires more resources.

3.2. Comparison of clustering algorithms

A study was conducted that involved implementation and comparison of the most common on-line clustering algorithms in the neural networks field. It involved measuring the clustering-performance and convergence speed on recorded datasets, as well as required resources.

A general list of the algorithms can be seen in Table 1, and we direct the reader to [14] for a much more detailed report.

The different properties listed in Table 1 give an indication on the feasibility of the algorithm's implementation. An algorithm with fixed network topology is easier to implement since the amount of required storage is known from the start. Soft competitive algorithms generally need more processing time than hard competitive algorithms since they update multiple prototypes for each input signal, however this property is needed when the algorithm has to preserve the topology of the input space.

Table 1. General overview of the algorithms implemented and tested for unsupervised clustering of our sensor data: the on-line variants of: Kohonen's self-organizing map (SOM), the recurrent self-organizing map (RSOM), K-means clustering, Hartigan's sequential leader clustering, growing K-means clustering, neural gas, and neural gas with competitive Hebbian learning (NG+CHL). See [7] for an in-depth overview of most of these algorithms.

On-line Algorithm	*Network Topology*	*Topology preserving*	*Memory required*	*Competitive*	*Real-time execution*
SOM	Fixed	yes	++	soft	++
RSOM	Fixed	yes	+++	soft	+
K-Means	Fixed	no	+	hard	+++
Leader	Variable	no	++++	hard	+++
G K-Means	Variable	no	++++	hard	+++
Neural Gas	Variable	no	++	soft	+
NG+CHL	Variable	no	+++	soft	+
GNG	Variable	no	++++	soft	+

The Kohonen self-organizing map was chosen for a variety of reasons. As a neural network, it is fairly flexible and robust for noisy data. In addition, it is computationally low, and uses the same amount of storage at all times (which makes it easier to implement for small microprocessors and real-time applications). Finally, it is topology-preserving, which gives it an advantage over many other algorithms that tend to act as 'black boxes'.

4. A Stable Self-Organizing Map

Once the choice for the Kohonen self-organizing map is made, several shortcomings of the algorithm prevent direct usage in real-world applications. Before the weaknesses are reviewed, an introduction on the algorithm focuses on the benefits.

4.1. Kohonen's Self-Organizing Map

The Kohonen self-organizing map [11] is an algorithm that has been used in many applications, usually for clustering and/or visualization of high-dimensional data onto two- or three-dimensional grids. It has two fundamental steps to project an incoming input vector (containing the sensorvalues, in this case) onto a certain position in its output grid. We will give a brief introduction into the algorithm below, while [11] provides much more detailed information and [6] gives a more critical evaluation.

The first step controls to what cell (or neuron) the input vector is projected to. The distance, usually the Euclidean, between the input vector and the prototype vector of each cell is measured, and the cell with the closest prototype vector is assigned as the target cell. This cell is also often called the winner or winning neuron. The second step replaces the prototype vectors of the winner and its neighbors with vectors that are (a bit) closer to the input. The components of the prototype vectors are often initiated as random values.

The Kohonen self-organizing map is a computationally efficient algorithm that is located somewhere between a multidimensional scaling algorithm and a clustering/vector quantization algorithm. On top of that, it also inherently tries to preserve the topology of the input space.

4.2. Stability

One of the biggest problems that the Kohonen self-organizing map suffered from (as mentioned in [22]) was related to the stability-plasticity dilemma [9]. It prohibited longer-term functioning of the algorithm since it could become unstable due to the algorithm forgetting previously learned contexts. Figure 5a shows the overwriting behavior of the self-organizing map and its detrimental effects in the initial phase.

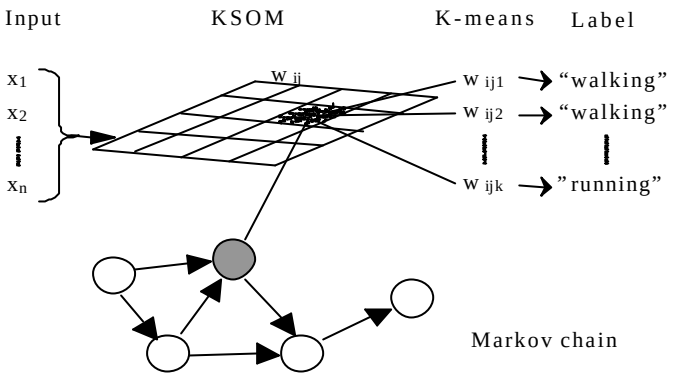

Figure 4. The structure of the algorithm: prototypes of the input are stored on the Kohonen self-organizing map(KSOM) and the K-means clustering layer. Transition probabilities are stored in the Markov Chain.

The search for a solution to this led to a guaranteed stable algorithm by combining the self-organizing map with the k-means-clustering algorithm, a statistical clustering algorithm [15]. The basic concept is to refine the representative capabilities of the self-organizing map with those of a specific k-means clustering mechanism, especially in the initial learning phase. Figure 4 shows a diagram of the complete algorithm, while Figures 5a and 5b demonstrate the superior behavior of the algorithm. The complete algorithm and its details are published in [21].

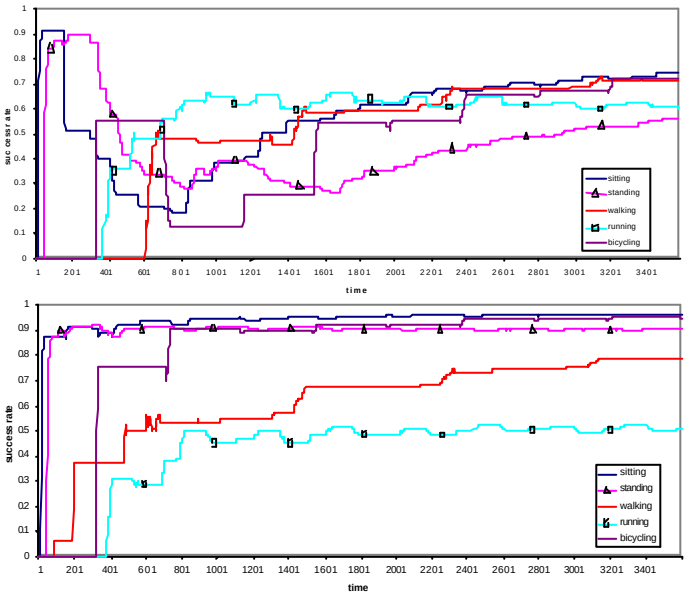

Figure 5 a) and b). Performance of the normal KSOM (top) versus that of the new, stable algorithm (bottom). The plots show success rate over time per context: The initially good success rate of the top plot drops for the first contexts because the prototype vectors were overwritten by those of other contexts (from [21]).

On top of this algorithm, a Markov model can calculate the probabilities that a transition between contexts can occur (as proposed in [22]). Figure 6 shows part of the output of the graphing module that runs in real-time during our experiments on a server.

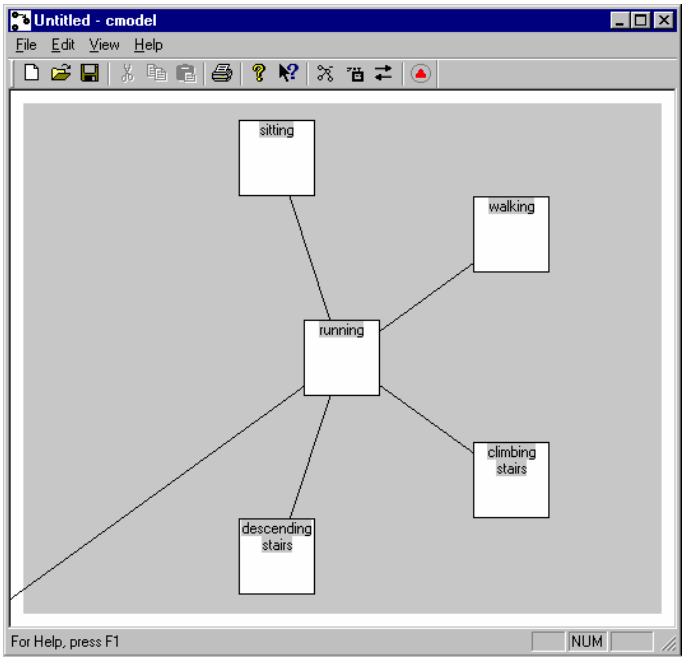

Figure 6. Partial transition graph of the Markov model; the shorter the distance between contexts, the more likely that context will be the next one. This model introduces a confidence measure in transitions of contexts in the algorithm.

Although there is still one obstacle preventing us from finalizing the algorithm (see section 4.4), we can already apply what we have so far to a few typical applications where only a few simple contexts are adequate. The next section will discuss two types of applications that were developed some time ago and were (and still are) used for long periods (several days up to a week). Even though movement-related activities were easiest to distinguish, several indoor locations with different lighting conditions were successfully identified.

These application descriptions should not be treated as anything more than initial usability tests. Comparison with other systems and applications mentioned in the introduction is very hard: recognition rates alone are not enough to measure the performance of the algorithm, since the behavior over time is important as well for application purposes. Most other approaches have a predefined training phase, after which the algorithm is not adaptive, whereas our approach allows the user to train the system at any time.

4.3. Applications

The wearable system, that was used to run the applications on, is based on a sensor board that sends the values from several simple sensors via a serial interface to a Compaq iPAQ handheld computer (based on the ARM processor from Intel), which is belt-worn. The sensorboard contains 2 accelerometers, 2 directional microphones (without high-level processing), photodiodes, a temperature sensor and a touch sensor, and is worn just above the knee. A picture of the setup can be found in Figure 7.

The iPAQ runs the Familiar distribution of Linux, with Blackbox as the X-windows manager. The recognition and logging software is mainly written in C++ and cross-compiled with gcc for Intel's ARM platform. For the graphic al front-end, a Python script was chosen since it allows rapid user-interface construction (see www.handhelds.org for extended information on any of these software packages).

4.3.1. Context logging. The first system can learn different, simple activities like sitting, standing, walking, running and bicycling, on the spot by pressing a designated button (as introduced in [22]) on the iPAQs' touch-sensitive screen. Whenever a change is detected in the context (for instance, when the context changes from "standing" to "sitting") the system logs this, together with the current time and date. This way, a diary of daily actions is automatically made. A short example is shown in Table 2.

Table 2. Excerpt from a context log, where the algorithm was trained on activities.

```
sitting    : Fri Jan   5 20:41:23 2001
sitting    : Fri Jan   5 20:41:31 2001
standing   : Fri Jan   5 20:41:31 2001
standing   : Fri Jan   5 20:41:43 2001
walking    : Fri Jan   5 20:41:44 2001
walking    : Fri Jan   5 20:42:31 2001
running    : Fri Jan   5 20:42:31 2001
standing   : Fri Jan   5 20:52:48 2001
walking    : Fri Jan   5 20:52:48 2001
standing   : Fri Jan   5 21:20:00 2001
sitting    : Fri Jan   5 21:20:01 2001
sitting    : Fri Jan   5 22:00:35 2001
standing   : Fri Jan   5 22:00:35 2001
standing   : Fri Jan   5 22:00:37 2001
```

This application was tested daily (usually from 9:00am till 12:00am) during several weeks, and the recognition rate indeed showed no signs of degrading during this period (the training was done during the first day). The log-file was reset after each time (although it could of course be uploaded onto another computer), but the algorithm's state and data were saved to enable the long-term reliability testing. Although the logging is at the moment done locally on the iPAQ, it is preferable to send this data at regular intervals to a server via wireless communications, making 'manual' uploading needless. Logging was only done for two hours a day, since the iPAQ, in the configuration mentioned above, can only run a few hours before its batteries are empty.

Figure 7. The wearable system setup is simple (two components) and fairly unobtrusive.

4.3.2. Autonomous starting of applications. A more complex chore is to automatically start processes or tasks depending on the current context. An interesting finding in our research is that if the recognition algorithm operates with some kind of confidence measure (or energy function) on the prediction, this value could also be used as a parameter for the task.

Outputting context-tailored music to play more appropriate songs, for instance, can become annoying if some error causes abrupt transitions. Using the confidence measure to change the volume proves to be an easy and a workable solution. This system has already been build on a desktop computer using system calls to the winamp mp3 player and we are currently implementing

a linux-variant to run on the iPAQ. Tests already proved that it has no problems running both an mp3 player and the recognition software.

4.4. What is missing?

The other obstacle mentioned in [22] is known in machine learning as the *curse of dimensionality* [1]. The more sensors, the more data has to be handled and stored by the algorithm, which leads to a slow or even unworkable algorithm as more sensors are added. Since the number of sensors in our experiments hasn't reached the critical limit yet, current versions of the algorithm are still workable. However, to really benefit from the multitude of sensors to distinguish more rich contexts, this must be solved as well.

One of the more obvious and feasible solutions to this problem might be to detect to what extent a sensor contributes to detecting or distinguishing a context. Storing and comparing only the relevant sensors for a specific context would decrease the necessary resources tremendously. Relevant feature detection, as this is generally called in machine learning [13], has many appearances, though, and will be the focus of our research in the near future.

5. Future work

Apart from investigating feature relevance detection algorithms to comply with all conditions necessary to establish a powerful algorithm, several other issues are on our agenda. The building of a network of sensorboards is required to increase the number of sensors that we can read. At the moment we are restricted to the processing limitations of one PIC microcontroller, building a network of these will allow us to distribute the sensors in a better way and use larger number of sensors.

6. Conclusions

Distinguishing the different contexts a wearable computer can encounter, by merely labeling them when they occur is still hard to realize without setting harsh constraints, usually on the available contexts. We believe the approach where the combination of many simple sensors provides enough information to categorize complex contexts is a promising one. Furthermore, clustering the data from multiple sensors in real-time is an ideal application for neural network algorithms.

After an overview of common algorithms, relevant to our problem, a stable neural network algorithm was deduced by combining the Kohonen self-organizing map with a sublayer of k-means clusters. This allows early implementation of the algorithm for usage in real-world context aware applications, although the number of sensors, as well as the complexity of the distinguishable contexts, is restricted.

As the dimensionality increases due to the large number of sensors, the learning is slowed down exponentially, but deducing the relevance of sensors might enable the usage of more appropriate numbers of sensors.

7. Acknowledgements

We would like to thank the people at Starlab's hardware and robotics lab, as well as the anonymous reviewers from ISWC for their suggestions and comments during the writing process of this paper. This research became possible thanks to the i-Wear [10] project; we thank all partners for their vital contributions.

8. References

[1] R. Bellman, *Adaptive Control Processes*. Princeton University Press (1961).

[2] B. Clarkson and A. Pentland, "Extracting Context from Environmental Audio". In *Proceedings of the Second International Symposium on Wearable Computers*, IEEE Press, pp.154-156, 1998.

[3] A. Dey and G. Abowd, "Towards a Better Understanding of Context and Context-Awareness". VU Technical Report GIT-GVU-99-22. 1999.

[4] Esprit Project 26900. Technology for Enabling Awareness (TEA). http://www.teco.edu/tea, 1999.

[5] J. Farringdon, A. Moore, N. Tilbury, J. Church and P. Biemond "Wearable Sensor Badge & Sensor Jacket for Context Awareness". In *Proceedings of the Third International Symposium on Wearable Computers* (ISWC'99), San Francisco, pp.107-113.

[6] A. Flexer, "Limitations of Self-organizing Maps for Vector Quantization and Multidimensional Scaling", in Mozer M.C., et al.(eds.), *Advances in Neural Information Processing Systems 9* , MIT Press/Bradford Books, pp. 445-451. 1997.

[7] B. Fritzke, "Some Competitive Learning Methods", http://www.neuroinformatik.ruhr-uni-bochum.de/ini/VDM/research/gsn/ , 1997.

[8] A. R. Golding and N. Lesh, "Indoor navigation using a diverse set of cheap, wearable sensors". In *Proceedings of the third International Symposium on Wearable Computers*, 1999, pp. 29-36. 1999.

[9] S. Grossberg, "Adaptive pattern classification and universal recoding, I: Parallel development and coding of neural feature detectors". *Biological Cybernetics* 23. pp.121-134. 1976.

[10] i-Wear, see http://www.iwear.com.

[11] T. Kohonen, *Self-Organizing Maps*, Second Edition, Springer-Verlag Heidelberg, 1997.

[12] G. Kortuem, Z. Segall, and M. Bauer, "Context-Aware, Adaptive Wearable Computers as Remote Interfaces to Intelligent Environments" In *Proceedings of the Second International Symposium on Wearable Computers*, pp. 58-66, IEEE Press., 1998.

[13] P. Langley, "Selection of relevant features in machine learning". In *Proceedings of the AAAI Fall Symposium on Relevance*, New Orleans, LA: AAAI Press., 1994.

[14] "Unsupervised Clustering Algorithms Comparison for On-line Sensor Clustering". http://www.teco.edu/tea/unsup/overview.html

[15] J. MacQueen, "On convergence of k-means and partitions with minimum average variance", Ann. Math. Statist. 36., 1084, 1965.

[16] MIT Wearable Computer FAQ: http://wearables.www.media.mit.edu/projects/wearables/FAQ/.

[17] W. Rungsarityotin and T. H. Starner, "Finding location using omnidirectional video on a wearable computing platform". In *Proceedings of the Fourth International Symposium on WearableComputers*, IEEE Press, pp.61-68, 2000.

[18] M. T. Rosenstein and P. R. Cohen, "Continuous Categories For a Mobile Robot". In *Proceedings of the AAAI '99*, AAAI Press, 1999.

[19] A. Schmidt, K.A. Aidoo, A. Takaluoma, U. Tuomela, K. Van Laerhoven & W. Van de Velde, "Advanced Interaction in Context", in H. Gellersen (Ed.) *Handheld and Ubiquitous Computing, Lecture Notes in Computer Science* No. 1707, Springer-Verlag Heidelberg, 1999, pp. 89-101.

[20] T. Starner, B. Schiele, and A. Pentland, "Visual Contextual Awareness in Wearable Computing". In *Proceedings of the Second International Symposium on Wearable Computers*, IEEE Press, pp.50-58, 1998.

[21] K. Van Laerhoven, "Combining the self-organizing map and k-means clustering for on-line classification of sensordata". In *Proceedings of the International Conference on Artificial Neural Networks 2001* (ICANN'01), Vienna, 2001.

[22] K. Van Laerhoven and O. Cakmakci, "What shall we teach our pants?". In *Proceedings of the fourth International Symposium on Wearable Computers (ISWC 2000)*, Atlanta (GA), IEEE Press, pp.77-83, 2000.