Person Movement Prediction, Smart Environment,
Artificial Neural Networks

Tomáš MIKLUŠČAK[*], Michal GREGOR[**]

PERSON MOVEMENT PREDICTION USING ARTIFICIAL NEURAL NETWORKS WITH DYNAMIC TRAINING ON A FIXED-SIZE TRAINING DATA SET

Abstract

Significant technical development over the last years has lately been showing more and more promise of making the vision of smart environments come true. The role of future smart environments lies in proactive interaction. Prediction of user's actions plays a vital role in such interaction. This paper presents a method based on artificial neural networks designed to accommodate the problem of person movement prediction. The paper explores the importance of dynamic training in prediction of nonstationary time series. An approach to dynamic training, based on the so-called on-the-fly training, is presented.

1. INTRODUCTION

Prediction of user's actions is likely to play a vital role in future smart environments, be these smart homes, smart offices, smart workplaces or other environments. This is due to the fact that prediction of user's actions most definitely represents the cornerstone of proactive interaction. Considerable effort has so far been invested into time series prediction in general as well as into prediction of person movement specifically.

Among the artificial intelligence methods often being applied to this problem are artificial neural networks (ANNs). These have now been shown to be able to approach the problem of time series prediction. However, should the time series be nonstationary, the classical training approaches that do not provide for continuous adaptation of the ANN (static training) will usually exhibit unreasonable prediction error and thus fail to provide a viable solution for the problem. Thus, dynamic learning methods are called for in order to provide the ANN with the ability to adapt online and learn from new data once they become available. This paper discusses some of the existing approaches and presents an approach to dynamic learning based on the notion of the so-called on-the-fly training.

* Ing. Tomáš Mikluščak – Department of Control and Information Systems, Faculty of Electrical Engineering, University of Žilina, Univerzitná 1, 010 26 Žilina, Slovak Republic, tomas.mikluscak@fel.uniza.sk

** Ing. Michal Gregor – Department of Control and Information Systems, Faculty of Electrical Engineering, University of Žilina, Univerzitná 1, 010 26 Žilina, Slovak Republic, michal.gregor@fel.uniza.sk

2. THE VISION OF UBIQUITOUS COMPUTING AND SMART ENVIRONMENTS

Ubiquitous computing is a post-desktop model of human-machine interaction, which expects computing to be integrated into everyday objects. This model considers traditional desktop model to be superseded by interaction with devices without even being aware of their functioning. It can also be characterized by the significant change of perspective – we are now striving to rebuild the machine interfaces in such way that they operate in the environment native to humans, instead of forcing humans to change their way of thinking, or use complex interfaces in order to interact with them. Ubiquitous computing is closely related and sometimes overlapping with pervasive computing, ambient intelligence, or the Internet of Things. However, the Internet of Things is much more physically oriented than pervasive and ubiquitous computing and ambient intelligence, which are more abstract paradigms.

Smart environment is a technological concept – Mark Weiser sees ubiquitous computing and the smart environment as “a physical world that is richly and invisibly interwoven with sensors, actuators, displays, and computational elements, embedded seamlessly in the everyday objects of our lives, and connected through a continuous network” [1]. The smart environment is a small world, where different kinds of smart devices are continuously working to make inhabitants' lives more comfortable [2]. Smart environment is sensing, predicting, making decisions and acting proactively, in order to improve user comfort and automate repetitive tasks. Proactive interaction is an essential part of the vision of future smart environments.

3. PERSON MOVEMENT PREDICTION

The ability to recognize and learn patterns in user behaviour forms the base of proactive interaction, and especially of smart environments like intelligent houses, offices, industrial buildings, etc. In general, human machine interaction in smart environment depends on their current and near-future location. Successful movement prediction and other related types of prediction can save energy and improve user comfort by saving their time. People waste a lot of time by routine operations such as control of heating, light, ventilation, various other devices, etc. The automation of routine operations is the current trend and it provides a way to save a lot of time and energy.

The need to save more time and to use our time in an increasingly productive way is nowadays perceived with extraordinary clarity. As for energy savings, these may arise from more optimal control – experiments have proved that prediction of user actions can significantly lower energy consumption [3].

Another important application of person movement prediction and other related predictions presents itself in care for elderly people. According to demographic data, population is ageing fast and the number of elderly people increases all over the world. Population ageing is a great challenge for humanity, and especially so for developed countries. Home care systems may represent one of the viable solutions as it may be able to substitute for the work of a nurse to a certain extent in the future. A home care system able to recognize and learn patterns in user's behaviour would be able to recognize unusual and potentially dangerous situations and call for help.

3.1 To what extent it is possible to predict movement

The assumption underlying all movement prediction systems is that human behaviour and particularly human movement in the smart environment can in principle be predicted. It is also of much importance that prediction of human behaviour, especially of movement, has been shown to lead to energy and time savings. The analysis of inhabitant’s daily lifestyle will tend to show identifiable patterns: these can be learned and therefore actions can, to a certain extent, be predicted. This, in turn, leads to the assumption that the inhabitant’s mobility is a piecewise stationary ergodic stochastic process, as hypothesized by Bhattacharya [4], whose work concerns tracking mobile users in a wireless cellular network. This research indicates that in spite of the fact, that there exists a number of possible routes in the house, the inhabitant will tend to have their own routine, and their own routes.

What is more, the underlying hypothesis that human behaviour and human movement can be predicted has to date been proven by many experiments, for example by smart house projects [3], [5] or smart office projects [6].

3.2 How to track inhabitants' movement

There exists a lot of technical solutions for mobility tracking, which can, in principle, be divide into two main categories. The first one utilizes recognition based on physical characteristics. The second one makes use of recognition based on user's possession of an object – for example an identification chip.

In large buildings with a lot of people, like hospitals, it will usually be better to use identification by the way of possession. An example of such a solution can be found in Priyantha's work: “The Cricket Indoor Location System” [7] developed at the Massachusetts Institute of Technology. There is a wide range of applications for this kind of a system, navigation of robotic devices and indoor navigation being among them.

In smaller buildings with no changing inhabitants it may be better to use a sensor network and cameras. An example of using cameras can be found in [8], a work coming from Microsoft. An example of using sensors, on the other hand, can be found in [9], which presents an approach to tracking inhabitants' movement using pressure sensors in the floor. What is more, current technologies can even recognize particular activities by use of a great numbers of sensors, as shown in [10] and [11]. A major concern when using physical recognition is privacy. This is especially an issue when using cameras.

3.3 An overview of existing solutions for movement prediction

There exist a number of approaches and methods based on artificial intelligence and mathematical methods, which have to date been employed in recognition of patterns in human behaviour in general as well as in the task of movement prediction itself. Among the most obvious candidates are Markov chains, Bayesian networks, and neural networks. This paper presents and ANN-based approach. Some of the existing ANN-based approaches will be discussed in a separate section. Concerning the other approaches, let us only mention prediction by partial matching (PPM), which comes from the theory of data compression. More specifically, PPM finds its base in Markov models theory, and especially in dynamic Markov compression.

PPM has successfully been used to predict movement in a smart office building project [12]. An application of PPM to this task has been demonstrated in MavHome project designed at university of Texas at Arlington as well [3]. MavHome uses the LeZi-Update algorithm. LeZi is based on information theory and constitutes a framework for location-aware resource management. More specifically, the concept of the typical set and the asymptotic equipartition property are utilized to find the inhabitant's most likely routes with some degree of accuracy and predict near-future movement and activities in an indoor environment.

3.4 Single-inhabitant vs. Multi-inhabitant movement prediction

There are some considerable differences between single and multiple inhabitant movement prediction. More people sharing the same smart environment may interact with the environment and with each other, and what is more, conflicting situations may arise. Optimal movement prediction for multiple inhabitant environments is proved to be an NP-hard problem. However, there are some methods that are able to deal with it to a certain extent. Roy uses game theory – cooperative and non-cooperative – in his dissertation thesis to approach the problem [13].

Multi-inhabitant movement prediction is more challenging in two ways. Firstly it is the above mentioned interaction between inhabitants. Secondly, it is very difficult to distinguish motion of every one inhabitant. In other words, if there is just one person in the environment, a simple sensing of motion is enough. However, if there are multiple inhabitants, it becomes much more difficult to track motion of inhabitant 1 and distinguish it from that of inhabitant 2. Inhabitant recognition would require use of cameras or id chips. It is obvious though that neither cameras nor the id chips can be counted among the most popular of solutions. The other way is to only sense motion and not recognize inhabitants at all. This of course has to be paid for by sacrificing some prediction accuracy. This paper focuses on multi-inhabitant movement prediction in such environments where recognition of inhabitants is not available or desirable.

4. THE SIMULATION MODEL

In our case study, the smart environment is represented by a house shown in Fig. 1. We divide the house into 14 zones: every room is represented by a separate zone. Every zone is assigned a unique number from $\{1,2,...,14\}$. Movement of a person in such environment can then be expressed by a sequence of numeric ids such as 1, 2, 3, 6, 10, 12…

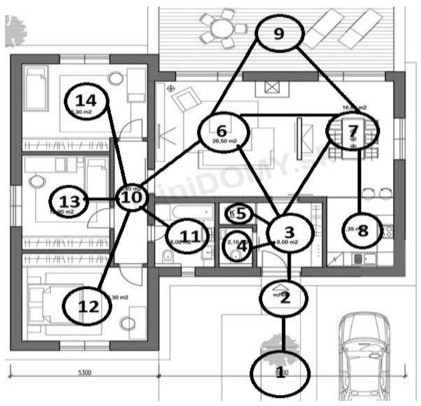

Fig. 1. Structure of the simulated environment

There are two independent inhabitants (agents) living in our simulated environment. The simulation runs in discrete time steps and does not take duration of being in various rooms into account. The name of the first software agent is Adam. Adam is 50, lives alone in the house and works at an office. The name of the second agent is Jane. Jane is 20 and she is Adam's daughter. Jane studies in another city, thus she occupies the smart environment only at weekends. We want to explore the ability of the neural network to adapt to such changes.

The simulation of behaviour for Adam and Jane is based on a combination of predefined scenarios and random decision making based on empirical and uniform probability distributions. Empirical probability distribution arranges that Adam and Jane visit some rooms (such as their bedrooms, toilet, ...) more often than other (like other bedrooms and kitchen). These probabilities are different on working days and at weekends. We use the uniform distribution of probability in situations with non-predicated behaviour, which accounts for approximately 5-10% of simulation time.

Adam acts according to predefined scenarios in the morning; we have several different scenarios for working days and for weekends. Morning scenarios are based on the assumption that morning behaviour is mostly routine, and is therefore the most easy to predict. Other scenarios include: after work scenario; after beer scenario (when Adam comes home after Friday's night beer); after weekend scenario – when Adam comes home from a weekend trip. As for night scenarios, every fifth night Adam wakes up between 2 a.m. and 4 a.m. and goes to the toilet with 90% probability, to the kitchen with 8% probability or to the front door with 2% probability. Jane's predefined scenarios are similar.

5. ARTIFICIAL NEURAL NETWORKS

Let us now provide some fundamentals concerning the theory of artificial neural networks. We will, for the sake of brevity, forebear from giving a more detailed overview of the basic theory, such as is to be found in [14] and other works. We will instead confine ourselves to discussing those features of artificial neural networks (ANNs) that are most closely related to the problem at hand.

5.1 Architecture of the artificial neural network

The way in which artificial neurons in an ANN are connected is often being referred to as the *architecture* of the ANN. There are several special types of architectures, most notably:

- Layered architecture;
- Non-layered architecture.

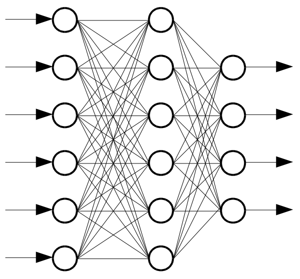
Fig. 2. A feed-forward neural network

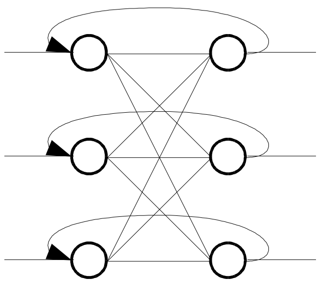
Fig. 3. An example of a recurrent ANN

When an ANN has the layered architecture, several layers of artificial neurons can be identified within its structure and, typically, only the adjacent layers are interconnected. Fig. 2 shows an example of such ANN. In the case of non-layered architecture, obviously, no layers of neurons can be discerned in the ANN.

There is another important distinction, concerning the architecture, this one being related to the way in which signals propagate through the network, namely whether there is any recurrence or not. With regard to this, we distinguish between:

- Feed-forward networks;
- Recurrent networks.

An example of a feed-forward network is shown in Fig. 2, an example of a recurrent ANN in Fig. 3. Implications of using recurrent or feed-forward networks in forecasting will be discussed in more detail shortly.

5.2 Learning methods

There is a multitude of methods designed to accommodate the learning problem in the theory of ANNs. When it comes to supervised learning, there is no doubt that the backpropagation algorithm is the best known approach. It is based on gradient descent and available in several flavours – most notably the batch and the incremental version. Several authors have strived to investigate which of these two versions is the more efficient one, e.g. [15].

There is a variety of other methods, some of which, however, are closely related, or even built atop backpropagation. An instance of such is the Rprop algorithm [16] (resilient backpropagation), which is, as the matter of fact, used to implement training in this work (or, to be precise, one of its enhanced versions is: the iRprop+ [17] as implemented in the FANN library).

ANNs can also be built and trained using genetic algorithms or similar evolutionary techniques. There are several works that discuss this topic and there is a multitude of approaches with different ways of representing ANNs genome. These methods generally tend to be more demanding computationally. However, they may prove to be useful in tasks where gradient descent is of limited use due to its tendency of getting trapped in local extremes [18].

6. ON USE OF ANNS IN FORECASTING

There is a number of papers that investigate applications of artificial neural networks to forecasting, or, if you like, to time series prediction. Several points are to be addressed when investigating such applications, including:

- Selecting an appropriate architecture of the ANN;
- Finding an appropriate data representation;
- Specifying a way in which the data is to be coded into inputs for artificial neurons.

6.1 Architecture of the artificial neural network

As mentioned hereinbefore, regarding the way in which signals are allowed to propagate through the network, there are two distinct types of architecture that an ANN may possess – the feed-forward architecture and the recurrent architecture.

Recurrent networks would seem to be especially well-suited to the task at hand as it is quite obvious that knowledge of several previous values (states) will generally be required in order to determine the forecast. There is a number of works that utilize this approach. It should be noted, however, that recurrent networks tend to be significantly more difficult to train than their more simplistic feed-forward counterparts. For this reason many of these works use various heuristic learning methods.

To name just a few, in [19] a recurrent neural network is applied to forecasting stock markets. The approach is based on some more complex preprocessing based on wavelet transforms. Learning is based on the artificial bee colony algorithm. Paper [20], on the other hand, optimizes the recurrent neural network through the use of genetic algorithms.

However, when investigating forecasting tasks, we are by no means confined to use of recurrent neural networks. If we are willing to provide for time delay of input signals explicitly, feed-forward networks represent a viable alternative. In fact, according to [21], of these two, this is the more widely used approach. Work [22] investigates the problem of time series forecasting using feed-forward networks and it also gives an overview of existing work up to that point. Another example, applying the feed-forward artificial neural network to movement prediction is presented in [5]. This work also investigates prediction of the expected occupancy patterns in the house over the next few hours, the expected hot water usage, and the likelihood that a zone will be entered in the next few seconds.

The data is fed to the ANN in the following fashion: let $f(k)$ be a time series we are trying to predict (and k be the discrete time step). Then, if we leave coding of the values out of consideration for the time being, the following is to hold for data fed into the ANN at point k:

$$X(k) = \{f(k-1), f(k-2), \ldots, f(k-n)\} \tag{1}$$

where n is the number of previous values of $f(k)$ we present to the network, that is, the order of the forecaster (or, if you like, the size of its memory). When specifying the value of n, one should therefore be reasonably certain that the system producing the outputs is of order equal or smaller than n. Should this condition not be satisfied, the ANN-based forecaster cannot be expected to provide us with accurate predictions.

In most practical applications the ANNs have a single hidden layer in addition to the input and output layer. As to the output layer, there would typically be (again not considering coding) a single neuron for an ANN that is to forecast one time step ahead. For ANNs supposed to provide predictions of m future values, there are several options:

- Train an ANN with a single output neuron to forecast a single future value $\bar{f}(k)$. Then use this forecast to compute the forecast for the next time step $\bar{f}(k+1)$. This kind of forecasting is known as *iterative forecasting* [22].
- Train an ANN with m output neurons to predict m future values. Such approach is known as *direct forecasting* [22].
- Use a combination of both – iterative and direct forecasting.

6.2 Data representation and coding

Several notes concerning the input data have been made in the previous section. If we use a feed-forward network to forecast time series, we are bound to present the network with n previous values of the time series. Formula (1) implies that such ANN-based forecaster will possess some sort of a container, effectively a FIFO buffer of size n and feed its elements to the input layer of the network.

However, this notion is not quite correct as we have chosen to ignore the issue of coding. Let us now consider this issue in more detail. Many works concerning ANN-based forecasting deal with continuous data such as currency exchange rates, river flow, rainfall, financial and demographic indicators, etc. For such applications what we have said so far is quite accurate. The only additional measure that we would probably have to take is to scale the numeric values appropriately – especially when using an activation function such as the sigmoid function: so as to avoid saturation [22].

It should be obvious that the problem of person movement prediction is not of the same nature – the observation, that is, the numeric id of the room into which the person moves next, is discrete. There is a finite number of rooms, each with its separate and unique numeric code. There are several approaches one might take when devising a representation fit for such observations:

- Feed the numeric code, such as it is, into a single artificial neuron (in the input layer);
- Have a separate input (and output) neuron for every room times every time step;
- Convert the numeric code into its binary form and assign one neuron to every bit of the resulting binary code.

The first option, as we can see, is the very same as used for forecasting of continuous numeric variables. Such approach has some significant downsides, perhaps the most important of which is the fact that the ANN will not generalize correctly – when we present it with an unknown input similar to both – such input as produces prediction of room 1 as well as such as produces prediction of room 8, we may reasonably expect that the output will be close to number 4, or some similar result in between 1 and 8. It is, in any case, improbable that it will be close enough to either 1 or 8 for the result to be evaluated correctly. Thus, it seems that such representation runs a risk of turning the one thing that ANNs are singularly good at – providing generalization capabilities – into its disadvantage.

If we were to select the second option, we would provide the ANN with $n.r$ input neurons, where r is the number of rooms. Input patterns would then be generated in such way that a given room would be represented by setting the input of its neuron to 1 while setting the inputs of all the other input neurons to 0. Output patterns would be generated in a similar fashion. This approach is the least likely to induce the problem described in the previous paragraph. However, it is rather impractical – size of the network increases rapidly with adding rooms and increasing the order of the forecaster.

The third approach is the one we have actually selected as it provides something of a middle ground between the other two methods.

7. DYNAMIC TRAINING ON A FIXED-SIZE TRAINING DATA SET

This section will provide a brief summary, or rather simply a list of the building blocks employed in our system. As mentioned earlier, our approach is based on a feed-forward ANN. We use static and dynamic training (the difference being described in the next section) both of which make use of the iRprop+ training mechanism mentioned in section . As to coding and representation, we have already mentioned that we transform the numeric code of the room into its binary form and feed the binary digits to the input neurons.

7.1 Static and dynamic training

Among the problems we are facing when trying to predict person movement is the fact that such time series will more often than not prove to be nonstationary. That is, the series are not only random for a large part, but even their statistical properties may change over time – users may change their habits, or some other factors that are not explicitly tracked may be at work like change of seasons, weather, etc.

In the case that the time series is nonstationary, it obviously does not suffice to select a portion of the collected data on which the ANN is then trained and evaluate its accuracy on the remaining data afterwards, as is the common practice. Such approach, known as *pre-training* or *static training* [6], although appropriate for stationary systems, will not work very well in the non-stationary case. Parameters acquired through learning will – with gradually changing statistical properties – quickly become outdated.

If, on the other hand, the parameters of the ANN are continually being adapted using new observations, such approaches are known as *dynamic training*. There is a number of ways to actually perform dynamic learning, instances of which are to be presented hereinafter. In any case, it should now be obvious that there is a necessity to employ dynamic learning, should it be implemented in one way or another.

In a previous section we have mentioned some of the basic training methods, all of which are conventionally used to implement static learning. Let us now provide an instance of a dynamic learning method. One such method is employed by Vintan et al. [6]. Incidentally, their work focuses on the very same problem this paper discusses. Their dynamic learning mechanism differs from our approach though. Once they've acquired the prediction by running the ANN forward, they determine whether the prediction is correct. If that is the case, a backward step is made. If, on the other hand, the prediction is incorrect, the backward step is applied until the prediction is correct and one more time after that [6].

7.2 Fixed-size training data set

The approach we propose is to assemble a data set of a predefined size. The data set is then to be used to pre-train the ANN. After that, the ANN is ready to make predictions. To account for nonstationary time series, we keep the initial data set and fix its size. Then every time a new observation is made, we present it to the forecaster which uses it to compute a new input-output pair, which is then inserted into the training data set we keep. It is obvious that in order to keep the size of the data set fixed, we have to delete one existing input-output pair for every new entry.

This approach is inspired by the so-called *On-the-fly training* method proposed by Pomerleau in [23]. Pomerleau uses the principle to train a controller for autonomous control of a vehicle – in his work the new data replaces such entries in the training data set, for which the ANN has the smallest error, that is, such entries that the ANN has already learned. After that, a single forward-backward run is performed. That is to say, a single epoch of training is performed in which every input-output pair is presented to the network exactly once.

Our own work, on the other hand, seems to suggest that such approach does not apply very well to forecasting. Our results show that the accuracy of the forecaster decreases drastically when such mechanism is in place. We have therefore, after conducting several experiments, decided to choose the entry that is to be replaced randomly. As shown in the following section, the behaviour is much improved in such case.

It is also possible to set the fixed size parameter to a value different from the size of the initial training data set. In such case some elements are erased from the initial data set after the initial training if the fixed size parameter is smaller than the size of the initial training data set. If, on the other hand, the fixed size parameter is greater, new input-output pairs are added to the data set without replacing the old ones until this fixed size is reached. In the experimental results, unless explicitly specified, the fixed size of the data set is the same as the size of the initial training data set.

There is another notion that requires discussion – how often the network should be trained and how many training epochs should be performed. In relation to this, let us now introduce the concept that we term the *observation batch number*. It has the form of $s:e$, where s is the size of the observation batch – the training is only performed every s observations with the maximum number of training epochs set to e. Several settings of the observation batch number are explored in the following section and the results are presented there as well.

8. EXPERIMENTAL RESULTS

Let us now present experimental results. Unless explicitly stated otherwise, these are all based on 2 observation sequences (denoted as *case1* and *case2*) generated by the simulation model described in section 4. For case 2, several parameters have been modified and thus both sequences exhibit significantly different behaviour. These cases will be used to evaluate the algorithm's ability to adapt in nonstationary prediction tasks using dynamic training with fixed-size training data set. As mentioned, these experiments are carried out using an environment with the number of rooms equal to 14 – therefore the total of 4 bits is sufficient to represent all rooms. The order of the forecaster is 6 (we present the network with six previous values and the current one).

8.1 Accuracy over time

As to evaluation of accuracy – the forecaster is presented with an observation sequence in every time step and predicts the following observation. The time steps are divided into windows of predefined size. The accuracy measure shown in the chart is then calculated for each window as follows:

$$a = n_c / n_w, \tag{2}$$

where n_w is the size of the window of time steps over which the accuracy is computed and n_c is the number of correct predictions. Thus the accuracy is the number of correct predictions over the number of time steps, that is, over the total number of predictions in that window. The number of windows is fixed to 75 so that the resulting charts can be compared with ease. Thus, size of window n_w may differ from one observation sequence to another.

Fig. 4 shows charts of accuracy over time with static training only for case1 and case2. Note that the forecaster is only trained statically for case1 and then tested on both case1 and case2. This data provides further evidence that dynamic training is vital when forecasting nonstationary time series. Accuracy evaluated over the entire run is 0.771862 for case1 and 0.364431 for case2.

When using both static training and dynamic training on a fixed-size training data set, the resulting accuracy is as shown in Fig. 5 and Fig. 6. Each of these figures is evaluated for a different observation batch number.

As we can see, the difference between observation batch number 5:3 (Fig. 5) and 10:3 (Fig. 6) is not that noticeable. There is of course the fact that it may take longer to react when adaptation is only done every 10 observations, but, on the other hand, this seems to help to decrease the effect of various anomalies such as that just before the end of case1 – as we can see, accuracy of 5:3 drops drastically at this point – it actually goes below 0.5, while in 10:3 this effect is less noticeable.

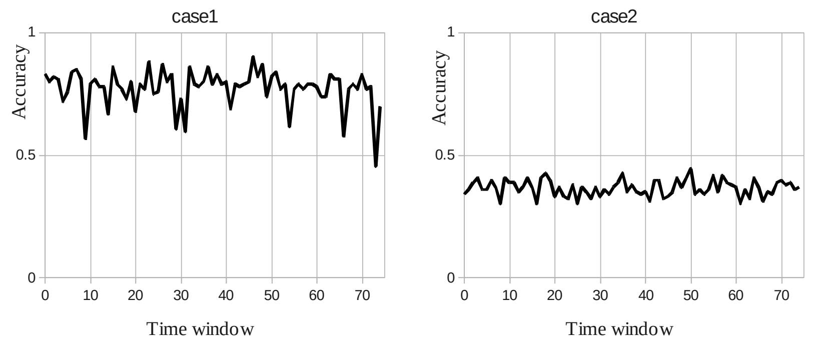

Fig. 4. Accuracy over time when using static training only

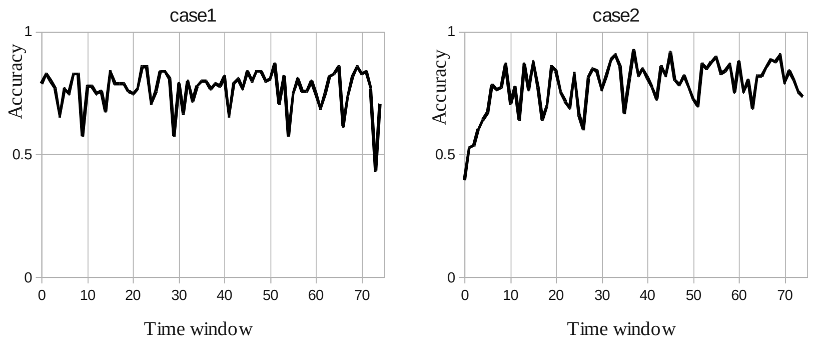

Fig. 5. Accuracy over time with static and dynamic training; 5:3

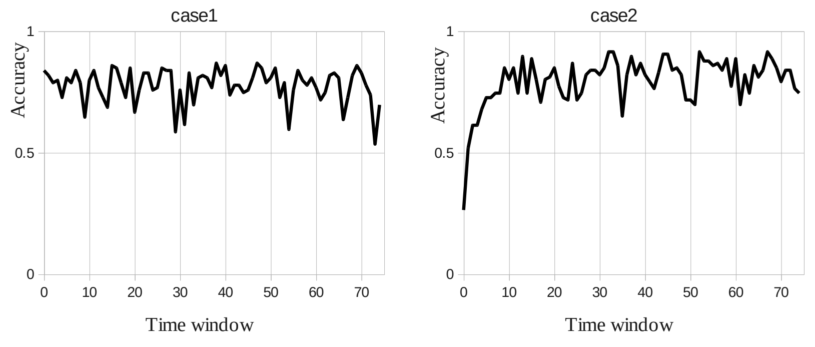

Fig. 6. Accuracy over time with static and dynamic training; 10:3

8.2 Accuracy over the entire run

Let us also present a table of accuracy computed over the entire run (Table 1). It also shows the margin between the results achieved by our method and those achieved by selecting a random room while respecting the physical possibilities set out in Fig. 1. The accuracy shown is the average over 50 runs of the algorithm.

Table 1. Accuracy over the entire run

Method / Case	Random selection	On-the-fly learning	Static training	Dynamic training, 5:3	Dynamic training, 10:3
Case1	0.51	0.549301	0.725239	0.764755	0.762407
Case 2		0.370096	0.38286	0.790061	0.784424

As we can see, the results are slightly better for case2, not for case1 on portion of which re-training was done. This is due to the statistical character of the series and similar behaviour would occur if we pre-trained using portion of case2. However, this shows that the online adaptation based on the proposed method of dynamic training is effective.

9. CONCLUSION

A dynamic learning method for artificial neural networks based on the so-called on-the-fly training has been presented and its features and merits have been discussed in some detail. The experimental data clearly show that the approach is effective and that it can be used to provide an artificial neural network with dynamic learning thus providing for online adaptation capabilities. It has been shown how such approach can be applied to the problem of person movement prediction. Both accuracy and dynamic properties of the solution have been investigated and the corresponding experimental results provided.

Concerning future research a more detailed model with more zones should be created so as to investigate scalability of the solution, preferably in comparison to other related methods. In addition, it would also seem appropriate to extend the model so as to accommodate time in a more precise and expressive manner.

Application of this algorithm to an extended model with multiple zones per room (with every zone being linked to a given function of the room – e.g. kitchen could be divided into 4 zones: surroundings of the fridge, kitchen sideboard, middle zone and entering/leaving zone) should also provide interesting results.

It would also be necessary to provide further testing of the algorithm on other data, preferably on data collected in a real environment such as the Augsburg Indoor Location Tracking Benchmarks mentioned in [6].

Acknowledgements

This paper has been worked out in the framework of project No. ITMS 26220220100, supported by ASFEU - Agency for Structural Funds of European Union.

References

[1] WEISER, M.: *The Computer for the 21st Century*. In: Scientific American: vol. 265, no. 3. 1991

[2] COOK, D., DAS, S.: *Smart Environments: Technology, Protocols and Applications*. Wiley-Interscience. Hoboken, New Jersey, 2005. ISBN 0-471-54448-5

[3] ROY, A., DAS, S.K., BASU, K.: *A Predictive Framework for Location-Aware Resource Management in Smart Homes*. In: IEEE Transactions on Mobile Computing, Vol. 6. 2007. ISSN: 1536-1233

[4] BHATTACHARYA, A., DAS, S.K.: *LeZi-update: An information-theoretic framework for personal mobility tracking in PCS networks*. In: Wireless Networks, vol. 8, no. 2. Kluwer Academic Publishers, 2002

[5] MOZER, M.: *The Neural Network House: An Environment that Adapts to its Inhabitants*. In: Proceedings of the American Association for Artificial Intelligence Spring Symposium on Intelligent Environments. Menlo Park, CA: AAAI Press. 1998

[6] VINTAN, L., GELLERT, A., PETZOLD, J., UNGERER, T.: *Person Movement Prediction Using Neural Networks*. 2004. Universität Augsburg, Fakultät für Angewandte Informatik: Technical Report

[7] PRIYANTHA, N.B.: *The Cricket Indoor location system*. Massachusetts Institute of Technology, Dissertation Thesis. 2005

[8] KRUMM., J., ET AL.: *Multi-camera multi-person tracking for Easy Living*. In: Proceedings of the Third IEEE International Workshop on Visual Surveillance. 2000. ISBN: 0-7695-0698-4

[9] ORR, R.J., ABOWD, G.D.: *The Smart Floor: A Mechanism for Natural User Identification and Tracking*. In: CHI'00 extended abstracts on Human factors in computing systems. Georgia Institute of Technology. 2000

[10] TAPIA, E., INTILLE, S., LARSON, K.: *Activity recognition in the home using simple and ubiquitous sensors*. In: Pervasive Computing, Second International Conference, PERVASIVE. 2004

[11] PARK S., KAUTZ H.: *Hierarchical Recognition of Activities of Daily Living using Multi-Scale, Multi-Perspective Vision and RFID*. In: Intelligent Environments, IET 4th International Conference. 2008. ISBN 978-0-86341-894-5

[12] PETZOLD, J., ET AL.: *Global State Context Prediction Techniques Applied to a Smart Office Building*. In: Communication Networks and Distributed Systems Modeling and Simulation Conference. San Diego, California. 2004

[13] ROY, N.: *A Context-Aware Learning, Prediction and Mediation Framework for Resource Management in Smart Pervasive Environments*. University of Texas at Arlington, Dissertation Thesis. 2008

[14] VAN DER SMAGT, P., KRÖSE, B.: *An Introduction to Neural Networks*. 1996. http://www.avaye.com/files/articles/nnintro/nn_intro.pdf

[15] WILSON, D.R., MARTINEZ, T.R.: *The General Inefficiency of Batch Training for Gradient Descent Learning*. In: Neural Networks. Elsevier, 2003

[16] RIEDMILLER, M., BRAUN, H.: *A Direct Adaptive Method for Faster Backpropagation Learning: the Rprop Algorithm*. In: IEEE International Conference on Neural Networks. 1993

[17] IGEL, C., HÜSKEN, M.: *Improving the Rprop Learning Algorithm*. In: Proceedings of the Second International Symposium on Neural Computation, NC'2000. 2000

[18] YAO, X.: *Evolving Artificial Neural Networks*. In: Proceedings of the IEEE, vol. 87: 9. 1999. ISSN 0018-9219 http://www.gpa.etsmtl.ca/cours/sys843/pdf/Ref2.pdf

[19] HSIEH, T.J., HSIAO, H.F., YEH, W.C.: *Forecasting Stock Markets Using Wavelet Transforms and Recurrent Neural Networks: An Integrated System Based on Artificial Bee Colony Algorithm*. In: Applied Soft Computing: Vol. 11, Issue 2. 2010

[20] RAMIREZ-ROSADO, I.J., FERNANDEZ-JIMENEZ , L.A.: *Short-term wind power forecasting using simple recurrent genetically optimized neural networks*. In: MIC '08 Proceedings of the 27th IASTED International Conference on Modelling, Identification and Control. 2008

[21] WONG, W.K., XIA, M., CHU, W.C.: *Adaptive neural network model for time-series forecasting*. In: European Journal of Operational Research 207. Elsevier, 2010

[22] PLUMMER, E.A.: *Time Series Forecasting with Feed-forward Neural Networks: Guidelines an Limitations*. Department of Computer Scienceand The Graduate School of The University of Wyoming, MSc Thesis. 2000

[23] POMERLEAU, D.A.: *Efficient Training of Artificial Neural Networks for Autonomous Navigation*. In: Neural Computation: 3. 1991