Neural network models for intelligent networks: deriving the location from signal patterns

Roberto Battiti, Alessandro Villani, and Thang Le Nhat

Università di Trento, Dipartimento di Informatica e Telecomunicazioni
via Sommarive 14, I-38050 Povo (TN), Italy
battiti|avillani|thang@science.unitn.it

Abstract. The knowledge of the location of a mobile terminal can be used to reduce the cognitive burden on the users in context–aware systems and it is a crucial information for routing in ad–hoc networks and for optimizing the topology in order to minimize the energy consumption of the nodes. The strengths of the RF signals arriving from more access points are related to the position (*location fingerprinting*), but the complexity of the inverse problem to derive the position from the signals and the lack of complete information, motivate to consider flexible models based on a network of functions (*neural networks*), developed through a supervised learning strategy.
The advantage of the method is that it does not require ad-hoc infrastructure in addition to the wireless LAN, while the flexible modelling and learning capabilities of neural networks achieve small errors in determining the position, are amenable to incremental improvements, and do not require the detailed knowledge of the access point locations and of the building characteristics. The system does not participate in an active manner to determine the position, therefore guaranteeing a complete privacy.
Experimental results and comparisons with alternative techniques are presented and discussed.

Keywords: location- and context-aware computing, wireless LAN, IEEE 802.11b, neural networks, machine learning.

1 Introduction

The research in this paper proposes a new method to determine the location of a mobile terminal. Knowledge of the location and suitable models are important in order to reduce the cognitive burden on the users in context- and location-aware systems [7, 1, 14]. Location awareness is considered for example in the infostation-based hoarding work of [12], and in the websign system of [15]. In addition, the location of mobile terminals is required by techniques for routing in ad–hoc networks and for optimizing the topology in order to minimize the energy consumption of the nodes, see for example [13].

Therefore, many different system and technologies to determine the location of users for mobile computing applications have been proposed. Besides Global Positioning System (GPS), used very successfully in open areas but ineffective indoor, there are

several indoor location techniques such as Active Badge[19] using infrared signals, Active Bat[20, 9], Cricket[16], using a combination of RF and ultrasound to estimate the distance, PinPoint 3D-iD[21], RADAR[2], Nibble[6] and SpotON[10], using RF-based location determination.

The current paper considers a high-speed wireless LAN environment using the IEEE 802.11b standard. The method is based on neural network models and automated learning techniques. As it is the case for the RADAR system, no special-purpose equipment is needed in addition to the wireless LAN, while the flexible modelling and learning capabilities of neural networks achieve lower errors in determining the position, are amenable to incremental improvements, and do not require the detailed knowledge of the access point locations and of the building characteristics in addition to a map of the working space. Preliminary results of this work are present in [5].

The following part of this paper is organized as follows. Section 2 describes the methodology for modelling the input-output relationship through multi-layer perceptron neural networks, Section 3 describes the system and the collection of data points for the experiments, Section 4 analyzes the distance error results obtained from the neural network and Section 5 discusses the results obtained by using the *k-nearest-neighbors* method. The effects of the signal variability on the position error are assessed in Section 6.

2 Methodology: Models based on neural networks

In our system the signal strengths received at a mobile terminal from different access points (at least three) are used to determine the position of the terminal inside a working area. The starting point of the method is the relationship between distance and signal strength from a given access point. Detailed radio propagation models for indoor environments are considered for example in [2]. If one knows distances d_i from the mobile terminal to at least three different APs, one can calculate the position of the mobile terminal in the system.

In a variegated and heterogeneous environment, e.g. inside a building or in a complex urban geometry, the received power is a very complex function of the distance, the geometry of walls, the infrastructures contained in the building. Even if a detailed model of the building is available, solving the direct problem of deriving the signal strength given the location requires a lengthy simulation. The inverse problem, of deriving the location from the signal strengths is more complicated and very difficult to solve in realistic situations.

Neural network models and automated learning techniques are an effective solution to estimate the location and to reduce the distance error. The non-linear transformation of each unit and a sufficiently large number of free parameters guarantee that a neural network is capable of representing the relationship between inputs (signal strengths) and outputs (position). Let us note that the distance from the access points, and therefore the detailed knowledge of their position, is not required by the system: a user may train and use the method without asking for this information.

The specification of the free parameters of the model (also called ”weights” of the network) requires a learning strategy that starts from a set of labelled examples to con-

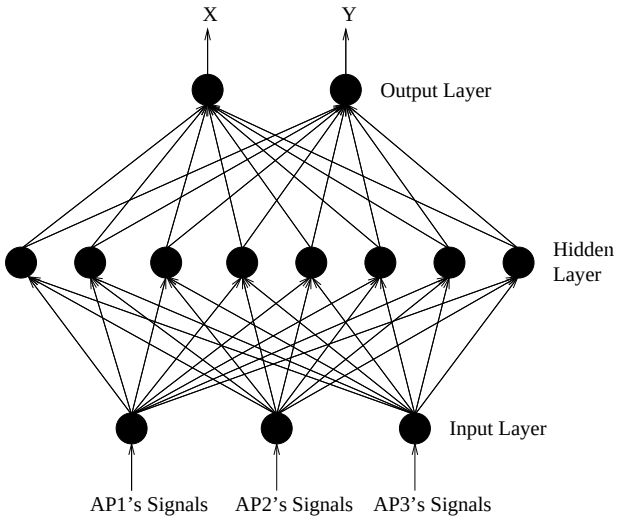

Fig. 1. An example of multi-layer perceptron configuration

struct a model that will then generalize in an appropriate manner when confronted with new data, not present in the training set.

We consider the "standard" multi-layer perceptron (MLP) architecture, see for example Fig. 1, with weights connecting only nearby layers and the sum-of-squared-differences *energy* function defined as:

$$E(w) \;=\; \frac{1}{2}\sum_{p=1}^{P} E_p \;=\; \frac{1}{2}\sum_{p=1}^{P}(t_p \;-\; o_p(w))^2 \tag{1}$$

where t_p and o_p are the target and the current output values for pattern p, respectively, as a function of the network weights w.

The architecture of the multi-layer perceptron is organized as follows: the signals flow sequentially through the different layers from the input to the output layer. For each layer, each unit ("neuron") first calculates a scalar product between a vector of weights and the vector given by the outputs of the previous layer. A transfer function is then applied to the result to produce the input for the next layer. The transfer function for the hidden layers is the sigmoidal function: $f(x) = 1/(1+e^{-x})$, while for the output layer it is the identity function, so that the output signal is not bounded.

It has been demonstrated that a network with a single hidden layer is sufficient to approximate any continuous function to a desired accuracy, provided that the number of hidden neurons is sufficiently large [11]. In this work we consider a single-hidden-layer MLP and a training technique that uses second-derivatives information: the one-step-secant method with fast line searches OSS introduced in [3, 4]. The one-step-secant method *OSS* is a variation of what is called *one-step (memory-less) Broyden-Fletcher-*

Goldfarb-Shanno method, see [17]. The OSS method is described in detail and is used for multi-layer perceptrons in [3] and [4].

3 System and experimental setup

Our system consists of a wireless Local Area Network based on the IEEE 802.11b standard. It is located on the first floor of a 3-storeyed building. The floor has dimensions of 25.5 m x 24.5 m, for a total area of 624.75 m^2 and includes more than eleven rooms (offices and classrooms).

The three access points are AVAYA WP-II E model by Lucent Technologies, two with external antennas. The wireless stations are Pentium-based laptop computers running Linux version 7.2. Each laptop is equipped with the ORiNOCO PC card - a wireless network interface card by Lucent Technologies.

The network operates in the 2.4 GHz license-free ISM band and supports data rates of 1, 2, 5.5, and 11Mbps. The 2.4 GHz ISM band is divided into 13 channels (IEEE & ETSI Wireless LAN Standard) and only three channels are used: channel 1, 7, 13 at 2412, 2442 and 2472 MHz respectively, in order to minimize the interference.

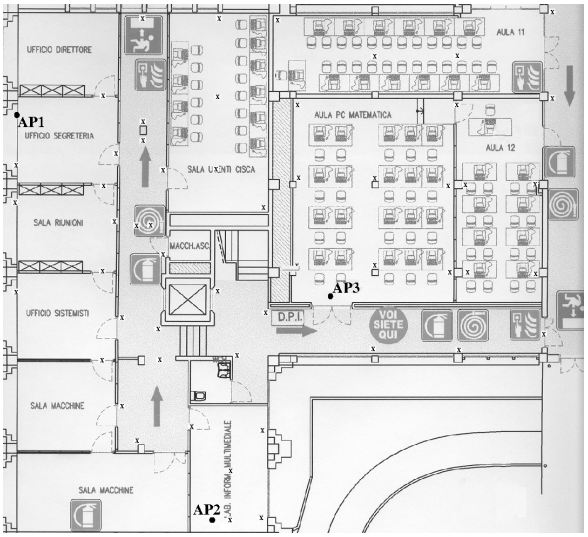

Fig. 2. The floor layout of the experiment, with access points locations.

To facilitate the collection of labelled example patterns, the map of the area is stored on a laptop and a user interface has been designed based on a single click on the displayed map. The origin of the coordinate system (0,0) is placed at the left bottom corner of the map. The (x, y) coordinates of the access points are as follows: AP1 $= (0.66m, 19.36m)$, AP2 $= (10m, 0.5m)$, AP3 $= (15.3m, 8.6m)$.

When the user is at an identifiable position in the experimental area (e.g., at the entrance of a room, close to a corner, close to a column, etc.) he clicks on the displayed map in a point corresponding to the current position. Immediately after the click, the three received radio signal strengths from the APs are automatically measured and they are saved together with the point's coordinates in a file, to prepare the examples for training and testing the neural network.

A total of 194 measurement points are collected during different periods of the day.

4 Selection of the multi-layer perceptron architecture

A labelled training set (given by inputs signals and corresponding output locations) is used by the OSS learning algorithm to determine the free parameters of the flexible MLP architecture. The measure of the error on the training set given by eq. 1 is minimized by OSS during the learning procedure.

It is essential to note that the objective of the training algorithm is to build a model with good *generalization* capabilities when confronted with new input values, values not present in the training set. The generalization is related both to the number of parameters and to the length of the training phase. In general, an excessive number of free parameters and an excessively long training phase (*over-training*) reduce the training error of eq. 1 to small values but prejudice the generalization: the system *memorizes* the training patterns and does not extract the regularities in the task that make generalization possible.

The theoretical basis for appropriate generalization is described by the theory of the Vapnik - Chervonenkis (VC) dimension [18]. Unfortunately, the VC dimension is not easily calculated for a specific problem and experimentation is often the only way to derive an appropriate architecture and length of the training phase for a given task.

The purpose of the experiments in this section is to determine the architecture, in our case the number of hidden units, and the length of the training phase leading to the best generalization results. Fig. 3 describes a significant summary of the results obtained in the experiments.

The three architectures considered are given by 4, 8, and 16 hidden units. A set of labelled examples (signal strengths and correct location) has been collected as described in Sec. 3. Among all examples, 140 are extracted randomly and are used for the training phase, the remaining ones (54) are used to test the generalization, at different steps during the training process. The plotted value is the average absolute distance error DE over all patterns:

$$DE = \frac{1}{P} \sum_{p=1}^{P} \sqrt{(tx_p - ox_p(w))^2 + (ty_p - oy_p(w))^2} \tag{2}$$

where $ox_p(w)$ and $oy_p(w)$ are the x and y coordinates obtained by the network and tx_p and ty_p the correct "target" values. P is the number of test or training patterns, depending on the specific plot.

Both the training error and the generalization error are shown in each figure. As expected, the training error decreases during training, while the generalization error first

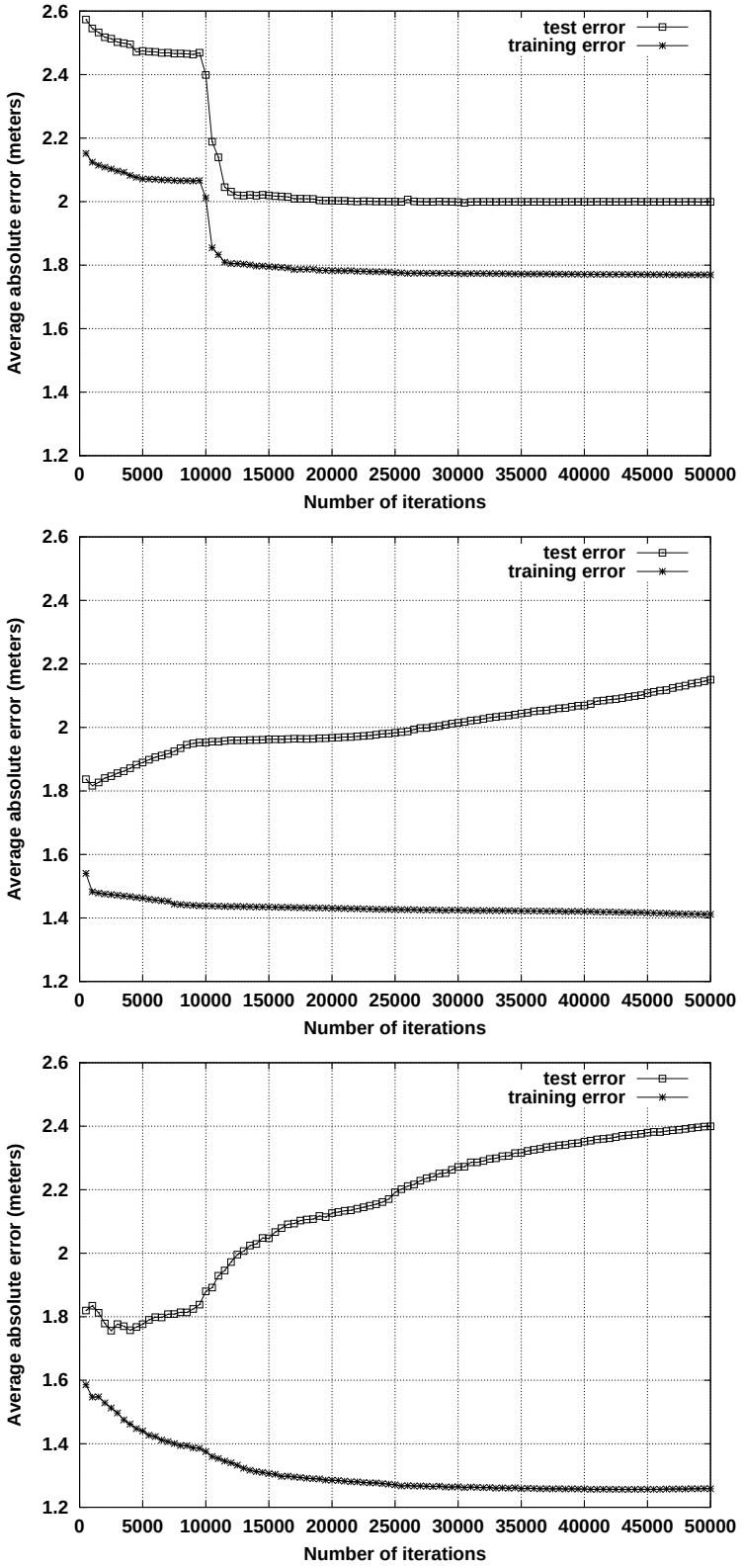

Fig. 3. Training and test error for architecture $3 \rightarrow 4 \rightarrow 2$ (top), $3 \rightarrow 8 \rightarrow 2$ (middle), $3 \rightarrow 16 \rightarrow 2$ (bottom).

decreases, then reaches a minimum value and finally tends to increase (*over-training* effect). The *over-training* effect is particularly strong for the architecture with 16 hidden units. The best generalization values (of about 1.80 meters) are reached after approximately 3,000 iterations for both the 8 and 16 hidden units architectures.

The robustness of the MLP model for different architectures and for different lengths of the training phase is to be noted. When the architecture changes from 4 to 8 to 16 hidden units, the optimal generalization value changes only by less than 14% (from 1.99 meters to about 1.75 meters). When the number of iterations increases from 2,000 to 50,000 the generalization error remains approximately stable for the more compact architecture (4 hidden units), and increases slowly for the architectures with more hidden units, as expected because of the larger flexibility of the architecture.

After this series of tests, the architecture $3 \rightarrow 16 \rightarrow 2$ achieves the best result (in the examples of Fig. 3 we obtain a result on the test set of 1.75 meters). The structure of the neural network used in the subsequent tests consists of three layers: 3 input units, 16 hidden layer units and 2 outputs. The CPU time for a single training session (3,000 iterations with 194 examples) on the architecture $3 \rightarrow 16 \rightarrow 2$ is of 21 seconds on a 1400 MHz Pentium IV.

To study the effect of the number of training examples on the generalization error a series of tests has been executed. For each test, a specified number of examples are extracted randomly from the entire set and used for training, while the remaining ones are used to test generalization after training is completed. For a given number of examples, 100 tests have been executed.

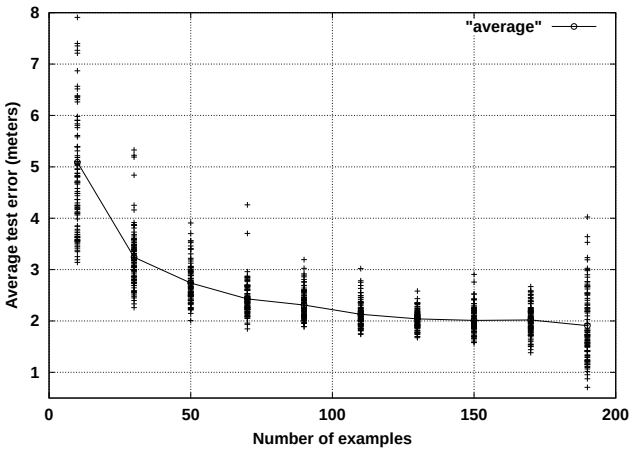

Fig. 4. MLP: reduction of average distance error (test) as a function of number of examples in the training set (with 16 neurons in the hidden layer): average and individual results.

The reduction of the average distance error (test) as a function of number of examples in the training set (with 16 neurons in the hidden layer) is illustrated in Fig. 4. The

average error decreases rapidly as a function of the number of examples, to reach a final average value of 1.91. The standard deviation of the individual results is larger when either the number of training examples or the number of test examples is very small.

In order to use the largest number of examples for training, the leave-one-out technique has been adopted for a final test. In this case, all examples apart from one are used for training, while the generalization is tested on the single left-out example. The experiment is repeated by considering each pattern in turn as the left-out example. When using an architecture with 16 neurons in the hidden layer the average result of the leave-one-out test is of 1.82 meters.

5 Comparison with K-Nearest-Neighbors

The k-nearest-neighbors rule is a well known non-parametric technique. For the case of classification, the nearest-neighbor rule classifies a pattern with the same label as the one of the closest labelled sample: the procedure is suboptimal but, with unlimited number of samples the error rate is never worse than twice the minimum (Bayes) error rate [8]. In the context of regression to obtain the location, the k-nearest-neighbors rule (a generalization that considers k instead of the single nearest neighbor) has been used in [2].

Because it is difficult to compare the results obtained in different experimental environments, the *k-nearest-neighbors* rule in the signal space has been used to classify the *same* examples used for the previous tests. This method computes the distances between the vector of the received signals at a certain point and the vectors of signals of a set of points with known locations. The vectors having the k smallest distances are considered and the estimated position is computed as an average of the positions associated with the smallest-distance vectors.

The same data set used for MLP is divided into two disjointed sets of examples: one for training and one for testing.

In the training set we consider both the signal strengths and the position:

$$(AP_1^i, AP_2^i, AP_3^i, x^i, y^i)$$

If the training set consists of N examples, in order to compute the position corresponding to a triplet of signal strengths (AP_1, AP_2, AP_3) taken from the test set, one proceeds as follow:

- For each $i \in 1 \ldots N$, compute the distance d_i between (AP_1^i, AP_2^i, AP_3^i) and (AP_1, AP_2, AP_3)
- Find the k smallest distances (with k fixed): $d_{i_1} \ldots d_{i_k}$
- Compute the estimated position as the average of the positions recorded for the k examples selected at the previous step: $x = \frac{\sum_{j=1}^{k} x^{i_j}}{k}$, $y = \frac{\sum_{j=1}^{k} y^{i_j}}{k}$

The standard Euclidean distance d_{i_i} is used:

$$d_{i_i}^2 = (AP_1^i - AP_1)^2 + (AP_2^i - AP_2)^2 + (AP_3^i - AP_3)^2$$

Instead of the simple average, a *weighted average* can be adopted. In this case the positions are weighted according to the inverse of the distance (unless the distance is zero, in which case the weight is 1,000). The estimated position is now computed as:

$$x = \frac{\sum_{j=1}^{k} x^{i_j} \frac{1}{d_{i_j}}}{\sum_{j=1}^{k} \frac{1}{d_{i_j}}}, \quad y = \frac{\sum_{j=1}^{k} y^{i_j} \frac{1}{d_{i_j}}}{\sum_{j=1}^{k} \frac{1}{d_{i_j}}}$$

To analyze the effect of the number of neighbors k on the error, the leave-one-out technique is used on the whole data set. Fig. 5 shows the average error results for the weighted and standard methods. In both cases, the best result is obtained for $k = 6$. The average distance error is 1.81 meters when using the normal average and 1.78 meters for the weighted average case. The weighted method achieves better results and is more stable when the parameter k increases: the effect of far away neighbors on the average becomes less dangerous, as expected.

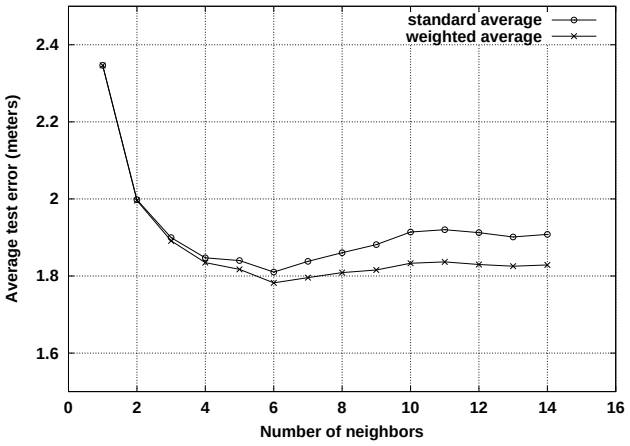

Fig. 5. K-NN: reduction of average distance error (test) as a function of number of neighbors: standard and weighted average.

A series of tests has been executed to study the average distance error when the number of the examples used for the training set is changed, in the same manner as in Sec. 4. As it is shown in Fig. 6 the error decreases from 4.5 meters by using 10 training examples, to less than 1.90 meters by using more than 170 examples. With only 50 examples the average distance error is already of 2.38 meters.

6 Effect of Signal Variability on Location Errors

The signal strength variability at a fixed position causes a location error that cannot be eliminated if a single measure is executed. This error can be considered as a lower

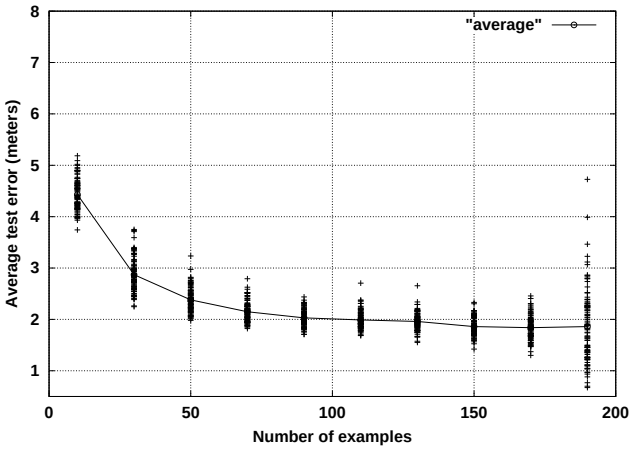

Fig. 6. K-NN: reduction of average distance error (test) as a function of number of examples in the training set (using 6 neighbors).

bound, unless more than one signal measure is executed (for example by averaging or by considering a moving average).

In order to verify the variability during the day of the signal strength and the corresponding variability of the position estimation, a mobile terminal was left at a fixed position to acquire the signals received from the access points during the day. At the end a total of 11,625 values of the signals for the same position were collected, although, because of quantization, only 465 different triplets of signals exist in the data set. The values were acquired during a working day when many people moved in the test-bed area.

After using the same training set as in Sec. 4–Sec. 5, the position of the mobile terminal is evaluated through both MLP and k-nearest-neighbors. Fig. 7–8 show the obtained positions and their distribution, by counting how many positions fall in each bin. Fig. 7 shows the result obtained with MLP. The error in the position estimation caused by signal fluctuation is rather limited, with 0.359 meters standard deviation. The Fig. 8 describes the result for k-nearest-neighbors case. The standard deviation for the distance is now 0.392 meters, a little higher than the standard deviation for MLP.

Both results indicate a significant gap between the error obtained with the presented techniques and the estimate of the location variability caused only by signal fluctuations. Additional work should therefore consider different and more powerful modelling techniques aiming at a possible additional reduction in the distance error.

7 Conclusion

A new model based on MLP neural networks to determine the position of a mobile terminal in a wireless LAN context has been presented. The technique has been evaluated in a real-world test-bed consisting of access points and mobile terminals using the

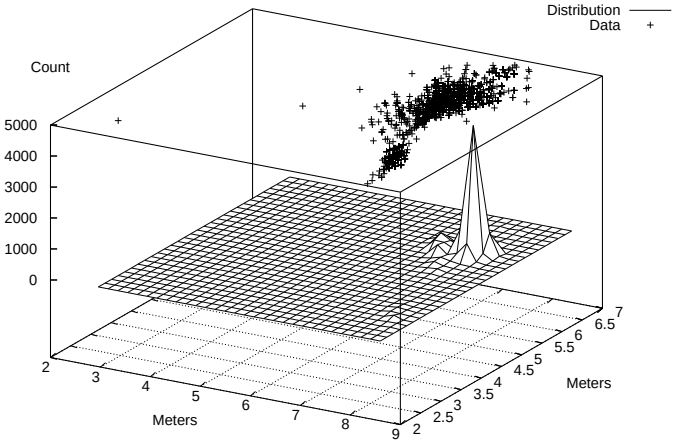

Fig. 7. Scattering of positioning using MLP with 16 neurons in the hidden layer.

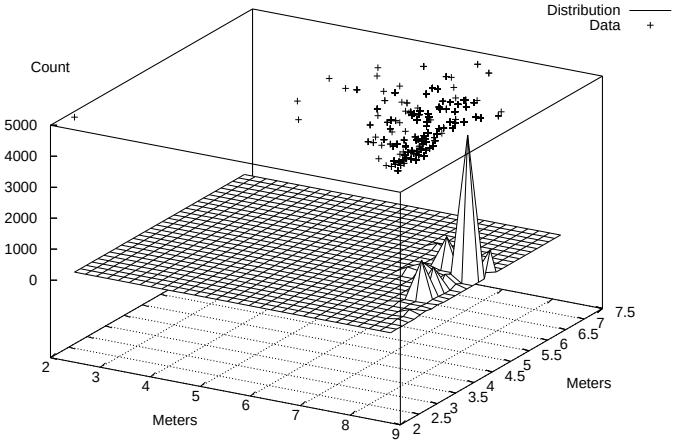

Fig. 8. Scattering of positioning using 6 nearest neighbors.

IEEE 802.11b standard. In addition, a comparison with an alternative method based on k-nearest-neighbors has been executed in the same experimental context. The results obtained with the MLP neural network (1.82 meters) are comparable to the best results obtained with the k-nearest-neighbors when the parameter k is optimized for the current context (1.81 meters for the standard average, 1.78 meters for the weighted average).

The fact that a basic method like the k-nearest-neighbors achieves similar results is an indication that the complex dependencies between signals and positions are not easily modelled by an MLP neural network.

The presented methods are not associated to the specific considered technology. We plan to continue this work by considering different technologies and contexts (e.g., the Bluetooth protocol) and different non-parametric modelling techniques that may lower the position errors to values closer to the errors caused by the signal fluctuations.

Acknowledgments

We acknowledge the financial support of the Wireless Internet and Location Management Architecture (WILMA) project, University of Trento and ICT-irst, and the help of Mauro Brunato.

References

1. J. Anhalt, A. Smailagic, D. P. Siewiorek, F. Gemperle, D. Salber, S. Weber, J. Beck, and J. Jennings. Toward context-aware computing: Experiences and lessons. *IEE Intelligent System*, 3(16):38–46, May-June 2001.
2. Paramvir Bahl and Venkata N. Padmanabhan. RADAR: An in-building RF-based user location and tracking system. In *IEEE INFOCOM 2000*, pages 775–784, March 2000.
3. R. Battiti. Accelerated back-propagation learning: Two optimization methods. *Complex Systems*, 3(4):331–342, 1989.
4. R. Battiti. First-and second-order methods for learning: Between steepest descent and newton's method. *Neural Computation*, 4:141–166, 1992.
5. R. Battiti, T. Le Nhat, and A. Villani. Location–aware computing: a neural network model for determining location in wireless LANs. Technical Report 5, Dipartimento di Informatica e Telecomunicazioni, Unversita' di Trento, Feb 2002.
6. P. Castro, P. Chiu, T. Kremenek, and R. Muntz. A probabilistic room location service for wireless networked environments. In *Ubicomp 2001: Ubiquitous Computing*, pages 18–34. Springer, 2001.
7. G.M. Djuknic and R.E. Richton. Geolocation and assisted GPS. *IEEE Computer*, pages 123–125, February 2001.
8. R.O. Duda and P. E. Hart. *Pattern Classification and Schene Analysis*. John Wiley and Sons, 1973.
9. Andy Harter, Andy Hopper, Pete Steggles, Andy Ward, and Paul Webster. The anatomy of a context-aware application. In *MOBICOM 1999*, pages 59–68, August 1999.
10. Jeffrey Hightower, Gaetano Borriello, and Roy Want. *SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength*. The University of Washington, Technical Report: UW-CSE 2000-02-02, February 2000.
11. Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4:251–257, 1991.

12. U. Kubach and K. Rothermel. Exploiting location information for infostation-based hoarding. In *Seventh Annual International Conference on Mobile Compuiting and Networking, Rome*, pages 15–27, 2001.
13. M. A. Marsan, C-F. Chiasserini, A. Nucci, G. Carello, and L. D. Giovanni. Optimizing the topology of bluetooth wireless personal area networks. In *Proceedings of IEEE Infocom 2002, in press*, 2002.
14. A. Misra, S. Das, A. McAuley, and S. K. Das. Autoconfiguration, registration and mobility management for pervasive computing. *IEEE Personal Communications (Special Issue on Pervasive Computing)*, 8(4):24–31, Aug 2001.
15. S. Pradhan, C. Brignone, J.-H. Cui, A. McReynolds, and M.T. Smith. Websigns: hyperlinking physical locations to the web. *IEEE Computer*, 34(8):42–48, August 2001.
16. Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. The cricket location-support system. In *MOBICOM 2000*, pages 32–43, August 2000.
17. D. F. Shanno. Conjugate gradient methods with inexact searches. *Mathematics of Operations Research*, 3(3):244–256, 1978.
18. V.N. Vapnik and A. Ja. Chervonenkis. On the uniform convergence of relative frequencies of events to their probabilities. *Theory Probab. Apl.*, 16:264–280, 1971.
19. Roy Want, Andy Hopper, Veronica Falcao, and Jonathan Gibbons. The active badge location system. *ACM Transaction on Information Systems*, 10(1):91–102, January 1992.
20. Andy Ward, Alan Jones, and Andy Hopper. A new location technique for the active office. *IEEE Personal Communications*, 4(5):42–47, October 1997.
21. Jay Werb and Colin Lanzl. Designing a positioning system for finding things and people indoors. *IEEE Spectrum*, 35(9):71–78, September 1998.