Statistical Learning Theory for Location Fingerprinting in Wireless LANs ⋆

Mauro Brunato *, Roberto Battiti

Università di Trento, Dipartimento di Informatica e Telecomunicazioni
via Sommarive 14, I-38050 Trento — ITALY

Abstract

In this paper, techniques and algorithms developed in the framework of Statistical Learning Theory are applied to the problem of determining the location of a wireless device by measuring the signal strength values from a set of access points (*location fingerprinting*). Statistical Learning Theory provides a rich theoretical basis for the development of models starting from a set of examples. Signal strength measurement is part of the normal operating mode of wireless equipment, in particular Wi-Fi, so that no special-purpose hardware is required.

The proposed techniques, based on the Support Vector Machine paradigm, have been implemented and compared, on the same data set, with other approaches considered in scientific literature. Tests performed in a real-world environment show that results are comparable, with the advantage of a low algorithmic complexity in the normal operating phase. Moreover, the algorithm is particularly suitable for classification, where it outperforms the other techniques.

Key words: Context-aware computing, Location management, Wi-Fi, Mobile computing, Statistical learning theory
1991 MSC: 90B18, 68Q32, 68T05

⋆ This research is partially supported by the Province of Trento (Italy) in the framework of the WILMA (Wireless Internet and Location Management) project (`http://www.wilmaproject.org/`)

* Corresponding author.
Email addresses: `brunato@science.unitn.it` (Mauro Brunato), `battiti@science.unitn.it` (Roberto Battiti).
URLs: `http://dit.unitn.it/~brunato/` (Mauro Brunato), `http://rtm.science.unitn.it/~battiti/` (Roberto Battiti).

1 Introduction

Context-aware computing, also known as *sentient computing*, refers to all techniques by which an electronic device may obtain information about the context in which it operates, so that applications can take advantage of this information. The word *context* refers both to physical world data (e.g., position, time, weather conditions) and to more abstract notions, such as distinction between work and leisure environments.

In mobile computing systems, where wireless networking is used to distribute contents and services to mobile users, a significant and useful context information is location. Knowledge about the user's position has many applications in civil, commercial, military and emergency environments, from helping a tourist through a town to advertising a restaurant to nearby people who are looking for a meal. Additional context information, such as weather or traffic conditions, can be inferred from location.

A wireless technology that is gaining a rapidly growing adoption is the "Wireless Ethernet" standard IEEE802.11b, also known with the more business-friendly name of "Wi-Fi" (Wireless Fidelity). A Wi-Fi network is characterized by a number of base stations, also called *access points*, placed throughout the networked environment and connected to a wired LAN. Each station has a range of roughly 300m in open space, and interference between different stations is dealt with by using different channels and by a CSMA/CA access protocol. Devices are connected by Wi-Fi cards that typically communicate with the access point having the strongest signal. Roaming between access points can be supported, and Wi-Fi networks can be extended to create "clouds of connectivity" inside the so-called *hotspots*, i.e. locations with high demand for connectivity such as office buildings, airports or even town centers.

To detect the position of a device, the intrinsic properties of wireless networks can be used, namely their use of radio signals (1; 2; 3; 4; 5). Propagation of radio signals is complex, and in most real-world cases deriving the signal strength from a model of the environment and of the electromagnetic signal propagation is extremely complex. The intensity of a radio signal at a given point can be obtained by measurements, and it usually varies with time due to many independent causes. For this reason, the functional dependence between the signal strength from an access point (RSSI, received signal strength intensity) and the physical position is not deterministic, but a statistical law connecting signal strength and position can be investigated. In this paper a new application of a powerful learning method is proposed for determining the location of a wireless device by using the intensity of the RSSI from wireless access points in a Wi-Fi network. Moreover, comparison between the proposed technique and other algorithms presented in scientific literature are

performed on the same set of experimental data in order to obtain a coherent set of performance measures.

This paper is organized as follows. Section 2 briefly reports previous work in the area of position estimation. Section 3 lists the assumptions on available hardware and user requirements that are at the basis of the present work. Section 4 describes the Statistical Learning Theory approach. Section 5 proposes the technique of Support Vector Machines and discusses their application to the Location Fingerprinting problem. Section 6 briefly describes other approaches that have been implemented by the authors, and that are used for comparison in Section 7, where they are tested, benchmarked and discussed.

2 Previous work

Various technologies are being proposed to determine the location of users in various contexts. They can be separated into two different branches, depending on whether they are assisted by dedicated hardware or not.

In the first branch, satellite-aided systems like GPS and GLONASS are the most widespread for open-space geolocation. Other techniques targeted at in-building environments use infrared, e.g., Active Badge (6; 7), or ultrasound, e.g., Active Bat by AT&T (8; 9) and Cricket (10), or radio signals that activate transponders attached to the item that must be located, e.g., 3D-iD by PinPoint Corp. (11). The SpotON project at Washington University (12) performs 3D location by using RFIDeas' AIR ID badge recognition system.

In the second branch, the properties of the communications medium are exploited. In particular, systems based on common radio communications technologies such as Wi-Fi and cellular telephony are being actively studied and developed by various research groups. The RADAR location system by Microsoft Research (1) relies on the Wi-Fi technology and calculates the position of a device either by empirical methods based on comparison with previous measurements, such as the nearest neighbors technique that shall be described and compared in this paper, or by signal propagation modeling. Bayesian inference models and other probabilistic approaches are being used both for Wi-Fi products (2; 3; 13) and for GSM telephone networks (14).

Previous work of our research group is mainly focused on the use of neural networks (4), on radio propagation model estimation and on classical statistical models (5).

3 Motivations

In order to allow a widespread use of our system, restrictive assumptions are made on the type of information that the mobile equipment can exchange with the environment.

In particular, all information gathering targeted at location estimation is *passive*: measurements do not require the active participation of the fixed infrastructure, but they are performed during its normal operation, so that the system can work along with any type of firewalling and restrictive policy imposed by system administrators. Another reason for using passive measurements is to avoid burdening the system with additional functions. Last but not least, privacy can be better guaranteed: the user knows his/her precise position, but the system does not (provided that some anonymization technique is supported by the network). It is up to the user to decide about how to use position information.

Moreover, the mobile equipment and the network infrastructure are composed by off-the-shelf standard communication hardware, with no additional equipment. This choice allows a significant cost reduction with respect to dedicated architectures. An important corollary is that all location-specific functions can be implemented by software, if possible at middleware/application level. However, RSSI measures must be read from the hardware through appropriate functions of the dedicated driver, so in some cases low-level or kernel-level software modifications are required.

From the user's point of view, the location detection software needs to be trained as fast as possible: the example collection phase, to be performed when first entering a new environment, must not require a long training phase, and as little knowledge as possible should be required about the environment. For example, the software should be able to operate starting from a user sketch of the environment and without requiring the knowledge of the access point positions. In other contexts, location estimation can be a service provided by the network manager, so that these requirements are not so strict. For instance, the network may offer to the user a digital map and the parameters of the trained location discovery system. In this case, training is done once by the network owner, and an accurate measurement and training process can take place.

For the above stated reasons, only location techniques based on passive signal strength measurements have been chosen for comparisons. These measurements are usually performed by listening to beacon packets whose purpose is to advertise the base station. Common wireless adapters (Wi-Fi, GSM or Bluetooth) provide received signal strength information, either as received

power (dBm) or as an arbitrary "connection quality" integer number. Other potentially relevant data, such as signal timing, or wavefront direction and phase, cannot be obtained without major hardware modifications, which are out of our scope, and will not be considered in this paper.

Methods based on dedicated hardware, such as those described in Section 2, yield precisions up to the millimeter scale, i.e., at least two orders of magnitude better than RSSI-based methods. However, they are rather expensive, and require the user or device to carry some dedicated badge or label, so their comparison with RSSI-based systems is pointless.

We chose to re-implement some of the techniques used in other works — namely, KNN (1), and Bayesian modeling (2; 3; 13) — rather than rely on the results reported by their Authors, because the experimental settings could not be replicated with sufficient accuracy and complete test sets were not available. Our tests have been performed by using IEEE802.11b (Wi-Fi) hardware (see Section 7), but the theoretical discussion applies to any location fingerprinting technique based on received signal strength, and comparisons should remain valid for other infrastructure-based wireless network systems.

4 Statistical Learning Theory

As mentioned, deriving the functional relationship between the position of the mobile terminal and the raw RSSI measurements is a hard task, both because of the difficulty in obtaining detailed knowledge about the building and because of the complexity of radio propagation indoor, characterized as site-specific, severe multipath, and with low probability for a line of sight propagation path between transmitter and receiver (15). The *inverse problem* of deriving the position from the signal strength values is even more demanding.

Tools derived from Statistical Learning Theory (16) can be used to *derive the unknown functional dependency* on the basis of observations. A shift of paradigm occurred in statistics starting from the sixties: in the previous paradigm based on Fisher's research in the 1920–30, in order to derive a functional dependency from observations one had to know the detailed form of the desired dependency and to determine only the values of a finite number of parameters. The new paradigm showed that the detailed knowledge is not required, and that some general properties of the set of functions to which the unknown dependency belongs are sufficient to estimate the dependency from the data.

While we refer to (16) for a detailed presentation of the theory, a brief summary of the main methodological points of Statistical Learning Theory is included to motivate the use of Support Vector Machines as a learning mechanism for

the *location fingerprinting* problem, and to define the methods, terminology and parameters used in the experimental tests, see (16; 17; 18) for more details. The reader who is already familiar with the basic aspects of statistical learning theory may move directly to Section 5.

Let ℓ be the number of observations, labeled examples to be used in supervised learning. In a *classification problem*, each example consists of a pair $(\boldsymbol{x}_i, y_i)$, $i = 1, \ldots, \ell$, where $\boldsymbol{x}_i \in \mathbb{R}^n$ is a vector and $y_i \in \{-1, +1\}$ is the label assigned by the supervisor. In the location fingerprinting application, $\boldsymbol{x}_i$ is the vector containing the signal strength values, thus n is the number of access points and can be seen as the dimension of the signal strength space, and y_i is equal to +1 if the location corresponds to a selected area, -1 otherwise.

In a *regression problem*, the label is a real number, $y_i \in \mathbb{R}$. In the location fingerprinting application, where the physical user coordinates are to be derived, a number of independent regression problems must be solved, one for each physical coordinate (two for the position on a planar map, up to five if three-dimensional position and orientation are required).

Let $P(\boldsymbol{x}, y)$ be the unknown probability distribution from which the examples are drawn. The learning task is to learn the mapping $\boldsymbol{x}_i \rightarrow y_i$ by determining the values of the parameters of a function $f(\boldsymbol{x}, \boldsymbol{\theta})$. The function $f(\boldsymbol{x}, \boldsymbol{\theta})$ is called *hypothesis* and the the set $\{f(\boldsymbol{x}, \boldsymbol{\theta}), \boldsymbol{\theta} \in \Theta\}$ is called the *hypothesis space* and denoted by $\mathcal{H}$. Θ is the set of abstract parameters. A choice of the parameter $\boldsymbol{\theta} \in \Theta$, based on the labeled examples, determines a "trained machine".

The *expected test error* or *expected risk* of a trained machine for the classification case is:

$$R(\boldsymbol{\theta}) = \int \|y - f(\boldsymbol{x}, \boldsymbol{\theta})\| \, \mathrm{d}P(\boldsymbol{x}, y), \tag{1}$$

while the *empirical risk* $R_{\mathrm{emp}}(\boldsymbol{\theta})$ is the mean error rate measured on the training set:

$$R_{\mathrm{emp}}(\boldsymbol{\theta}) = \frac{1}{\ell} \sum_{i=1}^{\ell} \|y_i - f(\boldsymbol{x}_i, \boldsymbol{\theta})\|. \tag{2}$$

The classical learning method is based on the *empirical risk minimization* (ERM) inductive principle: one approximates the function $f(\boldsymbol{x}, \boldsymbol{\theta}^*)$ which minimizes the risk in (1) with the function $f(\boldsymbol{x}, \hat{\boldsymbol{\theta}})$ which minimizes the empirical risk in (2).

The rationale for the ERM principle is that, if R_{emp} converges to R in proba-

bility (as guaranteed by the law of large numbers), the minimum of R_{emp} may converge to the minimum of R. If this does not hold, the ERM principle is said to be *not consistent*.

As shown by Vapnik and Chervonenkis (19), consistency holds if and only if convergence in probability of R_{emp} to R is replaced by *uniform* convergence in probability. Necessary and sufficient conditions for the consistency of the ERM principle is the finiteness of the *Vapnik-Chervonenkis dimension* (VC-dimension) of the hypothesis space $\mathcal{H}$.

The VC-dimension of the hypothesis space $\mathcal{H}$ is, loosely speaking, the largest number of examples that can be separated into two classes in all possible ways by the set of functions $f(\boldsymbol{x}, \boldsymbol{\theta})$. The VC-dimension measures the complexity and descriptive power of the hypothesis space and is often proportional to the number of free parameters of the model $f(\boldsymbol{x}, \boldsymbol{\theta})$. It is usually denoted by letter h, as in the following.

Vapnik and Chervonenkis also provide bounds on the deviation of the empirical risk from the expected risk. A bound that holds with probability $1 - \eta$ is the following one:

$$R(\boldsymbol{\theta}) \leq R_{\text{emp}}(\boldsymbol{\theta}) + \sqrt{\frac{h\left(\ln \frac{2\ell}{h} + 1\right) - \ln \frac{\eta}{4}}{\ell}} \quad \forall \boldsymbol{\theta} \in \Theta$$

By analyzing the bound, if one neglects logarithmic factors, in order to obtain a small expected risk, both the empirical risk *and* the ratio h/ℓ between the VC-dimension of the hypothesis space and the number of examples have to be small. In other words, a valid generalization after training is obtained if the hypothesis space is sufficiently powerful to allow reaching a small empirical risk (i.e., to learn correctly the training examples) but not too powerful to simply memorize the training examples without extracting the structure of the problem.

The choice of an appropriate value of the VC-dimension h is crucial to get good generalization performance, especially when the number of data points is limited.

The method of *structural risk minimization* (SRM) has been proposed by Vapnik based on the above bound, as an attempt to overcome the problem of choosing an appropriate value of h. For the SRM principle one starts from a nested structure of hypothesis spaces

$$\mathcal{H}_1 \subset \mathcal{H}_2 \subset \cdots \subset \mathcal{H}_n \subset \cdots \tag{3}$$

with the property that the VC-dimension $h(n)$ of the set $\mathcal{H}_n$ is such that

$h(n) \leq h(n+1)$. As the subset index n increases, the minima of the empirical risk decrease but the term responsible for the confidence interval increases. The SRM principle chooses the subset $\mathcal{H}_n$ for which minimizing the empirical risk yields the best bound on the actual risk. Disregarding logarithmic factors, the following problem must be solved:

$$\min_{\mathcal{H}_n} \left(R_{\text{emp}}(\boldsymbol{\theta}) + \sqrt{\frac{h(n)}{\ell}} \right) \tag{4}$$

The SVM algorithm described in the following section is based on the SRM principle, by minimizing a bound on the VC-dimension and the number of training errors at the same time.

5 Support Vector Machines for location fingerprinting

In this Section, Statistical Learning Theory, and in particular Support Vector Machines, shall be applied to the location fingerprinting problem. In the following, $\boldsymbol{x}$ will represent a vector of n radio signal measurements, where n is the number of access points. The resulting location information is represented by y, and it is always treated as a scalar quantity; when the outcome is expected to be a vector, for example in the regression problem, d independent problem instances are needed, where d is the dimension of the physical space.

The mathematical derivation of Support vector Machines is summarized first for the case of a linearly separable problem, also to build some intuition about the technique. The notation follows (17).

5.1 Linearly separable problems

Assume that the examples are linearly separable, meaning that there exist a pair $(\boldsymbol{w}, b)$ such that:

$$\begin{aligned} \boldsymbol{w} \cdot \boldsymbol{x} + b &\geq \quad 1 \quad \forall \boldsymbol{x} \in \text{Class}_1 \\ \boldsymbol{w} \cdot \boldsymbol{x} + b &\leq -1 \quad \forall \boldsymbol{x} \in \text{Class}_2 \end{aligned}$$

The hypothesis space contains the functions:

$$f_{\boldsymbol{w},b} = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b).$$

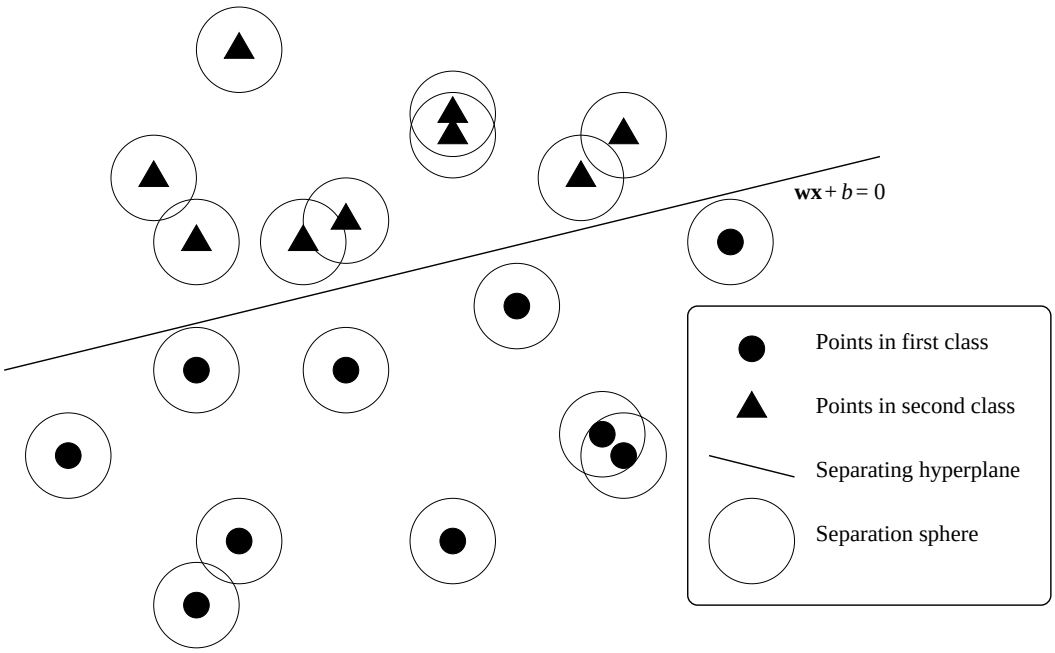

Fig. 1. Hypothesis space constraint.

Because scaling the parameters $(\boldsymbol{w}, b)$ by a constant value does not change the decision surface, the following constraint is used to identify a unique pair:

$$\min_{i=1,\ldots,\ell} |\boldsymbol{w} \cdot \boldsymbol{x}_i + b| = 1$$

A structure on the hypothesis space can be introduced by limiting the norm of the vector $\boldsymbol{w}$. It has been demonstrated by Vapnik that if all examples lie in an n-dimensional sphere with radius R then the set of functions $f_{\boldsymbol{w},b} = \text{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + b)$ with the bound $\|\boldsymbol{w}\| \leq A$ has a VC-dimension h that satisfies

$$h \leq \min\{\lceil R^2 A^2 \rceil, n\} + 1.$$

The geometrical explanation of why bounding the norm of $\boldsymbol{w}$ constrains the hypothesis space is as follows (see Figure 1): if $\|\boldsymbol{w}\| \leq A$, then the distance from the hyperplane $(\boldsymbol{w}, b)$ to the closest data point has to be larger than $1/A$, because we consider only hyperplanes that do not intersect spheres of radius $1/A$ placed around each data point. In the case of linear separability, minimizing $\|\boldsymbol{w}\|$ amounts to determining a separating hyperplane with the maximum *margin* (distance between the convex hulls of the two training classes measured along a line perpendicular to the hyperplane).

The problem can be formulated as:

$$\begin{aligned} &\text{Minimize}_{\boldsymbol{w},b}\ \tfrac{1}{2}\|\boldsymbol{w}\|^2 \\ &\text{subject to} \quad y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geq 1\ i = 1,\ldots,\ell. \end{aligned}$$

The problem can be solved by using standard quadratic programming (QP) optimization tools.

The dual quadratic program, after introducing a vector $\boldsymbol{\Lambda} = (\lambda_1, \ldots, \lambda_\ell)$ of non-negative Lagrange multipliers corresponding to the constraints is as follows:

$$\begin{aligned} &\text{Maximize}_{\boldsymbol{\Lambda}}\ \boldsymbol{\Lambda}\cdot\mathbf{1} - \tfrac{1}{2}\boldsymbol{\Lambda}\cdot D\cdot\boldsymbol{\Lambda} \\ &\text{subject to}\ \begin{cases} \boldsymbol{\Lambda}\cdot\boldsymbol{y} = 0 \\ \boldsymbol{\Lambda} \geq 0 \end{cases} \end{aligned} \tag{5}$$

where $\boldsymbol{y}$ is the vector containing the example classification, and D is a symmetric $\ell \times \ell$ matrix with elements $D_{ij} = y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j$.

The vectors $\boldsymbol{x}_i$ for which $\lambda_i > 0$ are called *support vectors*. In other words, support vectors are the ones for which the constraints in (5) are active. If $\boldsymbol{w}^*$ is the optimal value of $\boldsymbol{w}$, the value of b at the optimal solution can be computed as $b^* = y_i - \boldsymbol{w}^* \cdot \boldsymbol{x}_i$ for any support vector $\boldsymbol{x}_i$, and the classification function can be written as

$$f(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{\ell} y_i \lambda_i^* \boldsymbol{x}\cdot\boldsymbol{x}_i + b^*\right).$$

Note that the summation index can as well be restricted to support vectors, while all other vectors have null λ_i^* coefficients. The classification is determined by a linear combination of the classifications obtained on the examples y_i weighted according to the scalar product between input pattern and example pattern (a measure of the "similarity" between the current pattern and example $\boldsymbol{x}_i$) and by parameter λ_i^*.

5.2 Non-separable problems

If the hypothesis set is unchanged but the examples are not linearly separable one can introduce a penalty proportional to the constraint violation ξ_i

(collected in vector Ξ), and solve the following problem:

$$\begin{aligned} &\text{Minimize}_{\boldsymbol{w},b,\Xi}\ \frac{1}{2}\|\boldsymbol{w}\|^2 + C\left(\sum_{i=1}^{\ell}\xi_i\right)^k \\ &\text{subject to}\quad \begin{cases} y_i(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geq 1-\xi_i & i=1,\ldots,\ell \\ \xi_i \geq 0 & i=1,\ldots,\ell \\ \|\boldsymbol{w}\|^2 \leq c_r, \end{cases} \end{aligned} \tag{6}$$

where the parameters C and k determine the cost caused by constraint violation, while c_r limits the norm of the coefficient vector. In fact, the first term to be minimized is related to the VC-dimension, while the second is related to the empirical risk (see the above described SRM principle). In our case, k is set to 1.

5.3 Nonlinear hypotheses

Extending the above techniques to nonlinear classifiers is based on mapping the input data $\boldsymbol{x}$ into a higher-dimensional vector of *features* $\varphi(\boldsymbol{x})$ and using *linear* classification in the transformed space, called the *feature space*. The SVM classifier becomes:

$$f(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{\ell} y_i\lambda_i^*\varphi(\boldsymbol{x})\cdot\varphi(\boldsymbol{x}_i) + b^*\right)$$

After introducing the *kernel function* $K(\boldsymbol{x},\boldsymbol{y}) \equiv \varphi(\boldsymbol{x})\cdot\varphi(\boldsymbol{y})$, the SVM classifier becomes

$$f(\boldsymbol{x}) = \text{sign}\left(\sum_{i=1}^{\ell} y_i\lambda_i^* K(\boldsymbol{x},\boldsymbol{x}_i) + b^*\right),$$

and the quadratic optimization problem becomes

$$\begin{aligned} &\text{Maximize}_{\boldsymbol{\Lambda}}\ \boldsymbol{\Lambda}\cdot\mathbf{1} - \tfrac{1}{2}\boldsymbol{\Lambda}\cdot D\cdot\boldsymbol{\Lambda} \\ &\text{subject to}\quad \begin{cases} \boldsymbol{\Lambda}\cdot\boldsymbol{y} = 0 \\ 0 \leq \boldsymbol{\Lambda} \leq C\mathbf{1}, \end{cases} \end{aligned}$$

where D is a symmetric $\ell\times\ell$ matrix with elements $D_{ij} = y_i y_j K(\boldsymbol{x}_i,\boldsymbol{x}_j)$

An extension of the SVM method is obtained by weighting in a different way the errors in one class with respect to the error in the other class, for example when the number of samples in the two classes is not equal, or when an error for a pattern of a class is more expensive than an error on the other class.

This can be obtained by setting two different penalties for the two classes: C^+ and C^- so that the function to minimize becomes

$$\frac{1}{2}\|\boldsymbol{w}\|^2 + C^+(\sum_{i:y_i=+1}^{\ell}\xi_i)^k + C^-(\sum_{i:y_i=-1}^{\ell}\xi_i)^k.$$

If the feature functions $\varphi(\boldsymbol{x})$ are chosen with care one can calculate the scalar products without actually computing all features, therefore greatly reducing the computational complexity.

For example, in a one-dimensional space one can consider monomials in the variable x multiplied by appropriate coefficients a_n:

$$\varphi(x) = (a_0 1, a_1 x, a_2 x^2, \ldots, a_d x^d),$$

so that $\varphi(x) \cdot \varphi(y) = (1 + xy)^d$. In more dimensions, it can be shown that if the features are monomials of degree $\leq d$ then one can always determine coefficients a_n so that:

$$K(\boldsymbol{x}, \boldsymbol{y}) = (1 + \boldsymbol{x} \cdot \boldsymbol{y})^d.$$

The kernel function $K(\cdot, \cdot)$ is a convolution of the canonical inner product in the feature space. Common kernels for use in a SVM are the following.

(1) Dot product: $K(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x} \cdot \boldsymbol{y}$; in this case no mapping is performed, and only the optimal separating hyperplane is calculated.
(2) Polynomial functions: $K(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} \cdot \boldsymbol{y} + 1)^d$, where the *degree* d is given.
(3) Radial basis functions (RBF): $K(\boldsymbol{x}, \boldsymbol{y}) = \mathrm{e}^{-\gamma\|\boldsymbol{x}-\boldsymbol{y}\|^2}$ with parameter γ.
(4) Sigmoid (or neural) kernel: $K(\boldsymbol{x}, \boldsymbol{y}) = \tanh(a\boldsymbol{x} \cdot \boldsymbol{y} + b)$ with parameters a and b.
(5) ANOVA kernel: $K(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{n} \mathrm{e}^{-\gamma(x_i - y_i)}\right)^d$, with parameters γ and d.

When ℓ becomes large the quadratic optimization problem requires a $\ell \times \ell$ matrix for its formulation, so it rapidly becomes an unpractical approach as the training set size grows. Osuna, Freund and Girosi (17) introduce a decomposition method where the optimization problem is split into an active and an inactive set. Later, Joachims (20) introduces efficient methods to select the working set and to reduce the problem by taking advantage of the small number of support vectors with respect to the total number of training points.

5.4 Support Vectors for regression

Support vector methods can be applied also for regression, i.e., to estimate a function $f(\boldsymbol{x})$ from a set of training data $\{(\boldsymbol{x}_i, y_i)\}$. As it was the case for

classification, we start from the case of linear functions and then use preprocessing of the input data $\boldsymbol{x}_i$ into an appropriate feature space to make the algorithm nonlinear.

In order to fix the terminology, the linear case for a function $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b$ can be summarized. The convex optimization problem to be solved becomes:

$$\begin{aligned} &\text{Minimize}_{\boldsymbol{w}} \ \tfrac{1}{2}\|\boldsymbol{w}\|^2 \\ &\text{subject to } \begin{cases} y_i - (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \leq \varepsilon \\ (\boldsymbol{w} \cdot \boldsymbol{x}_i + b) - y_i \leq \varepsilon, \end{cases} \end{aligned}$$

assuming the existence of a function that approximates all pairs with ε precision.

If the problem is not feasible, a *soft margin* loss function with slack variables ξ_i, ξ_i^*, collected in vector Ξ, is introduced in order to cope with the infeasible constraints, obtaining the following formulation (16):

$$\begin{aligned} &\text{Minimize}_{\boldsymbol{w},b,\Xi} \ \frac{1}{2}\|\boldsymbol{w}\|^2 + C\left(\sum_{i=1}^{\ell} \xi_i^* + \sum_{i=1}^{\ell} \xi_i\right) \\ &\text{subject to } \begin{cases} y_i - \boldsymbol{w} \cdot \boldsymbol{x}_i - b \leq \varepsilon - \xi_i^* & i = 1, \ldots, \ell \\ \boldsymbol{w} \cdot \boldsymbol{x}_i + b - y_i \leq \varepsilon - \xi_i & i = 1, \ldots, \ell \\ \xi_i^* \geq 0 & i = 1, \ldots, \ell \\ \xi_i \geq 0 & i = 1, \ldots, \ell \\ \|\boldsymbol{w}\|^2 \leq c_r, \end{cases} \end{aligned} \tag{7}$$

As in the classification case, C determines the tradeoff between the flatness of the function and the tolerance for deviations larger than ε. Detailed informations about support vector regression can be found also in (21). Support Vector Machines are being successfully used for pattern recognition (16; 18), classification of text and of web pages (22).

6 Other approaches for location fingerprinting

To effectively evaluate the statistical learning approach for location fingerprinting, other techniques have been selected from scientific papers and implemented.

In the following section, based on the previously introduced notation, a training set of ℓ tuples shall be considered, where each tuple is of the form $(\boldsymbol{y}_i, \boldsymbol{x}_i)$, $i = 1, \ldots, \ell$, $\boldsymbol{x}_i$ being an array of n radio signal intensity values and $\boldsymbol{y}_i$ the location information. In particular, in the regression problem $\boldsymbol{y}_i$ is actually a

d-component vector (where $d = 2, \ldots, 5$ depending on the required location information), while in the classification problem $\boldsymbol{y}_i = \pm 1$.

6.1 Weighted k Nearest Neighbors (WKNN)

Let $k \leq \ell$ be a fixed positive integer; consider a measured signal strength array $\boldsymbol{x}$. A simple algorithm to estimate its corresponding location information y is the following:

(1) Find within the training set the k indices $i_1, \ldots, i_k$ whose radio strength arrays $\boldsymbol{x}_{i_1}, \ldots, \boldsymbol{x}_{i_k}$ that are nearest (according to a given radio-space metric) to the given $\boldsymbol{x}$ vector.
(2) Calculate the estimated position information $\boldsymbol{y}$ by the following average, weighted with the inverse of the distance between signal strength tuples:

$$\boldsymbol{y} = \frac{\displaystyle\sum_{j=1}^{k} \frac{\boldsymbol{y}_{i_j}}{d(\boldsymbol{x}_{i_j}, \boldsymbol{x}) + d_0}}{\displaystyle\sum_{j=1}^{k} \frac{1}{d(\boldsymbol{x}_{i_j}, \boldsymbol{x}) + d_0}}, \tag{8}$$

where $d(\boldsymbol{x}_i, \boldsymbol{x})$ is the radio distance between the two n-tuples (for example the Euclidean distance) measured in dBm, and d_0 is a small real constant ($d_0 = .01 dBm$ in our tests) used to avoid division by zero.

The WKNN algorithm is simple to implement, and results in Section 7 show that it achieves low estimation errors. Its main drawbacks, from the theoretical point of view, are the algorithmic complexity of the testing phase and the high VC dimension. A simpler technique, not using distance-dependent weights, was first proposed in (1). It shall be called KNN, for (unweighted) k Nearest Neighbors.

6.1.1 Learning phase complexity

The learning phase is rather simple: just store all examples. Recalling that n is the number of access points and d is the number of physical dimensions to determine, complexity is $O(\ell(n + d))$ in time (all tuples must be input) and $O(\ell(n + d))$ in space (all tuples must be stored).

6.1.2 Estimation phase complexity

All tuples must be scanned for searching the k nearest ones. If a thorough scan of the training set is performed for finding the nearest neighbors, and

every insertion in the set of k nearest indices is performed through a binary search, then computing the distance in the radio space has complexity proportional to the number n of radio coordinates, computing the average requires d operations, so the total worst-case time complexity is $O(\ell n \log k + d)$.

6.1.3 VC dimension

When $k = 1$, if ℓ is fixed then up to ℓ different points can be arbitrarily classified by building the appropriate training set. The same is true for generic k if we put points at large mutual distances in order to reduce the weight of all points but one. Thus, for a fixed training set cardinality ℓ, the VC dimension is $h \geq \ell$, so the ratio ℓ/h, the fundamental parameter for computing the risk confidence interval, can never be higher than 1; if the training set cardinality is not bound, $h = +\infty$. As a direct consequence, no acceptable confidence interval can be expressed via statistical learning theory techniques.

6.2 Bayesian modeling (BAY)

Methods based on conditional probability estimation require the knowledge of the signal propagation model, either in the form of an empirical distribution from repeated observations on each physical point in a given set (2), or by selecting a suitable radio propagation model and by estimating its parameters on the basis of empirical observations (14). In the first case, a large number of observations is required in order to have a precise distribution estimate at each sample point; in the second case, a proper radio propagation model is necessary.

Once the radio propagation model is determined, it can be used to calculate the conditional probability distribution $p(\boldsymbol{x}|\boldsymbol{y})$ of an n-tuple $\boldsymbol{x}$ of radio signal intensity values, given a d-tuple $\boldsymbol{y}$ of space coordinates. The Bayes law of conditional probability can be used to calculate the inverse dependency, in the form of a probability distribution of the physical coordinates depending on signal strength information. This probability distribution can be used in turn to calculate the position in many ways. In this paper, two possible position estimators shall be used to determine the user's location estimate $\hat{\boldsymbol{y}}$, namely the *average position*

$$\hat{\boldsymbol{y}} = \int \boldsymbol{y} \, \mathrm{d}P(\boldsymbol{y}|\boldsymbol{x}), \tag{9}$$

and the *maximum likelihood* estimator

$$\hat{\boldsymbol{y}} = \underset{\boldsymbol{y}}{\operatorname{maxarg}} \, P(\boldsymbol{y}|\boldsymbol{x}). \tag{10}$$

Detecting empirical distributions on many sample locations can take a long time, and it can be impractical. In this case an analytical radio model can be stated in the form of a linear dependence:

$$\boldsymbol{x} = \sum_{i=0}^{c-1} b_i \boldsymbol{\zeta}_i, \tag{11}$$

where c is the number of unknown parameters (coefficients) $b_0, \ldots, b_{c-1}$ of the model and $\boldsymbol{\zeta}_i$ is some suitable transform of the training set dependent variable vector $\boldsymbol{y}$.

For example, by using a logarithmic loss model where the signal decay depends on the logarithm of distance and on the number of walls crossed by line of sight, then $c = 3$ and the following transforms can be applied:

$$\boldsymbol{\zeta}_0 = \mathbf{1}, \qquad \boldsymbol{\zeta}_1 = \log \boldsymbol{d}_{\mathrm{AP}}(\boldsymbol{y}), \qquad \boldsymbol{\zeta}_2 = \boldsymbol{w}_{\mathrm{AP}}(\boldsymbol{y}),$$

where $\boldsymbol{d}_{\mathrm{AP}}(\boldsymbol{y})$ represents the vector of distances of the physical point $\boldsymbol{y}$ from the access points (logarithms are computed componentwise), $\boldsymbol{w}_{\mathrm{AP}}(\boldsymbol{y})$ represents the number of walls and b_0 becomes a constant term (a similar model is used in (1), where walls exceeding a maximum number are not counted in w_{AP}). Rewriting (11) with these transforms leads to a possible radio propagation model:

$$\boldsymbol{x} = b_0 \mathbf{1} + b_1 \log \boldsymbol{d}_{\mathrm{AP}}(\boldsymbol{y}) + b_2 \boldsymbol{w}_{\mathrm{AP}}(\boldsymbol{y}).$$

Let us note that now the complete knowledge about the AP and walls position in the building is required. This approach will be discussed and extended in Section 7.2.

6.2.1 Learning phase complexity

Depending on the chosen approach to model the radio propagation phenomenon, different complexities arise from the corresponding algorithms. If a model fitting approach is chosen, for instance a linear fit on simple computable functions (logarithms, polynomials) of physical values, then the solution of a linear model is required. In case of a linear model with c unknown coefficients such as (11), solution of a system in the form

$$\hat{\boldsymbol{b}} = (M^T M)^{-1} M^T \boldsymbol{y},$$

is required, where M is an $n\ell \times c$ matrix, and $\boldsymbol{y}$ is a column vector with $n\ell$ components. The number of rows is given by the number of training samples ℓ, each carrying information about the signal strength of n access points. In the following complexity evaluation, straightforward matrix calculation algorithms shall be considered: lower exponents can be achieved by using optimized techniques for matrix inversion and multiplication, such as the Strassen

algorithm (23). Calculation of $M^T M$ requires the evaluation of all mutual dot products in the c columns of M, each having $n\ell$ elements, amounting to $O(c^2 n\ell)$ operations (of course, presence of constant columns and symmetry can be exploited to reduce the dominating term constant). Inversion of the resulting $c \times c$ matrix requires $O(c^3)$ time, and subsequent multiplications by M^T and by $\boldsymbol{y}$ require $O(c^2 n\ell)$ and $O(n\ell)$ operations respectively. To achieve an acceptable confidence interval, the number $n\ell$ of samples must be much larger than the number c of parameters in the model, thus the total time complexity to calculate the model coefficients amounts to $O(c^2 n\ell)$.
Similar values arise when measurements are used to estimate position-dependent parameters of the signal strength distribution. However, the system may just store the empirical signal strength distributions as histograms, one per sample position, thus reducing the preprocessing time complexity, but increasing space requirements.

6.2.2 Estimation phase complexity

This is the most consuming phase. In fact, a conditional probability distribution on the physical space must be evaluated on the physical space in order to be averaged or maximized. Computationally, this amounts to discretizing space by placing a grid, and calculating a probability function for each point. The discretization step has a direct influence on the final positioning error; by taking a 50cm $\times$ 50cm discretization step on a 25m $\times$ 25m square area, the product of n independent probability distributions must be calculated on a 2500-points grid.

6.2.3 VC dimension

Like all classical paradigms, where functional dependency is known up to a finite number of parameters, the VC dimension of the family of functions (in our case, radio propagation models) in the model is low, being approximately proportional to the number of parameters. As a consequence, strict estimations can be done on the loss functional confidence interval. On the other hand, the dimension cannot be tuned, so a reasonably low value for the empirical loss functional can be achieved with high probability only through a high number of measures.

6.3 Multi-Layer Perceptrons (MLP)

A multi-layer perceptron neural network is composed of a large number of highly interconnected units (*neurons*) working in parallel to solve a specific problem.

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 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

$$f(x) = \frac{1}{1 + \mathrm{e}^{-x}}.$$

Other transfer functions can be used for the output layer; for example, the identity function can be used for unlimited output, while the sigmoidal function is more suitable for "yes/no" classification problems.

The training technique adopted in this paper is the one-step-secant (OSS) method with fast line searches (24) using second-order derivative information. Usage of neural networks in conjunction with the localization problem was proposed in (4).

6.3.1 Learning phase complexity

The time complexity of the learning phase is usually high; for example, in our case it takes a few minutes, although it varies greatly according to the heuristic adopted for calculating the weights (usually iterative) and to the halting condition. Usually a few thousand passes are required, each involving computation of the outcome for every training set point, followed by modification of weights.

6.3.2 Estimation phase complexity

Calculating the outcome of a neuron with n_{IN} inputs requires the evaluation of a linear function on input data, which is performed in $O(n_{\mathrm{IN}})$ time, while the sigmoid function is evaluated in constant time. Calculation of the outcomes of the H hidden-layer neurons, having n inputs each, takes $O(nH)$ time, while $O(Hd)$ time is required for evaluating all d output neurons. Thus, the total time complexity is $O(H(n + d))$.

6.3.3 VC dimension

While determining the VC dimension of a neural network is usually complex, an estimate for single-output feed-forward neural networks is proved in (25) leading to a $O(H^2W^2)$ upper bound, where H is the number of hidden neurons and W is the number of weights. In our classification problem, where we

consider a complete feed-forward network, $W = O(nH)$, so the upper bound is $O(n^2H^4)$. For the regression problem, an upper bound can be simply calculated by considering two distinct single-output neural networks, so that the VC-dimension of the whole system is bounded by the product of the two: $O(n^4H^8)$. While this latter bound can become quite large, both n and H are predetermined and usually low.

7 Experimental results

For concise reference in figures and tables, the algorithms being compared shall be referred to as SVM (Support Vector Machine), WKNN (Weighted k Nearest Neighbors), BAY (Bayesian approach) and MLP (Multi-Layer Perceptron). The unweighted version of the WKNN algorithm shall be referred to as KNN.

7.1 Setup

The target system for our experiments is a wireless LAN using the IEEE802.11b (Wi-Fi) standard. The LAN is composed by six AVAYA WP-II E access points by Lucent Technologies, equipped with external omnidirectional antennas. The mobile equipment is a Compaq iPAQ H-3870 palmtop computer with Familiar Linux 0.52 operating system, equipped with a PCMCIA adapter and an ORiNOCO Silver card by Lucent Technologies. The target environment is roughly 30m $\times$ 25m large. Its map is shown on Figure 2.

To test the Support Vector Machine algorithm, the `mySVM` implementation(26) has been chosen. All other algorithms are implemented in C and C++ language by the authors. All tests are performed on Linux-based machines, or on Windows-based machines using the Cygwin emulation library. The machine used for time benchmarks is a 1.7GHz PIII Linux desktop computer with 256MB RAM.

The sample set used in the following experimental analysis consists of 257 measurements throughout the floor. A text file containing the outcome of the measurement session can be found at

`http://ardent.unitn.it/software/data/`.

The positions where the samples have been measured can be seen in Fig. 8. A fully regular grid could not be followed due to the presence of various obstacles such as tables and pieces of furniture. At each sample point one complete measurement was taken by a person standing with the PDA in his hand and always oriented towards the same direction (north). The outcome of

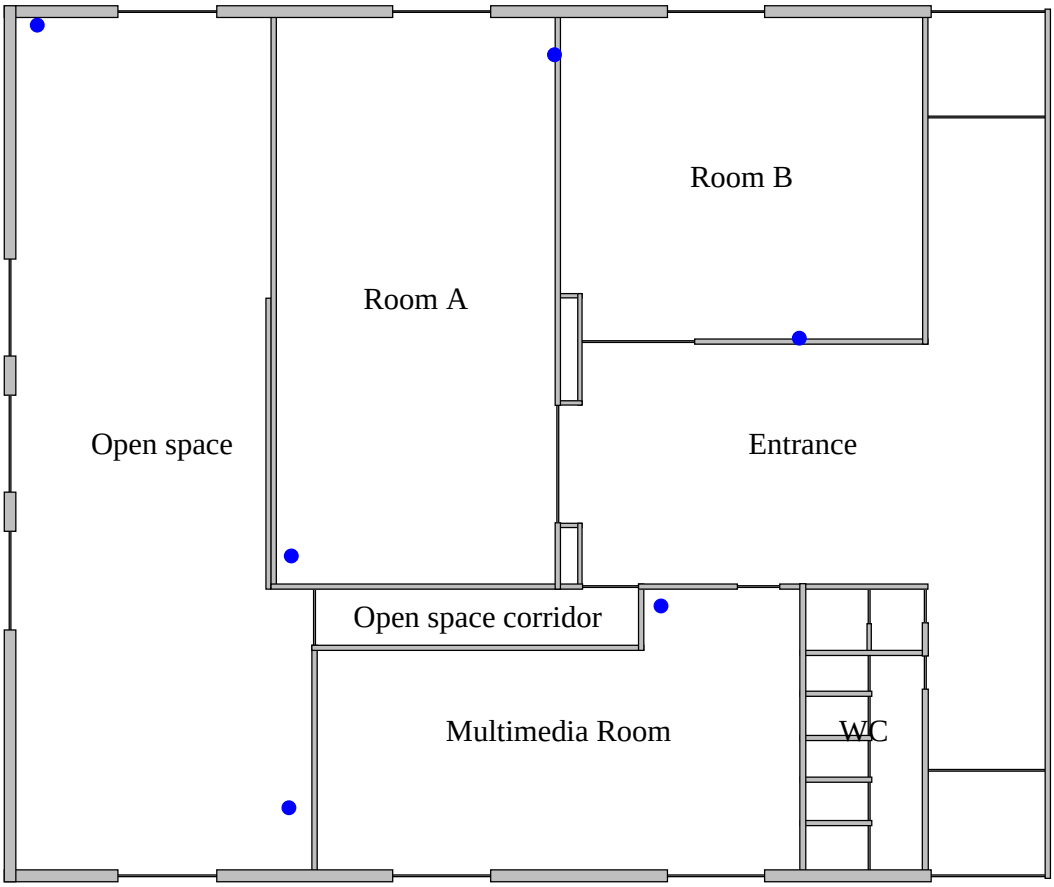

Fig. 2. Map of the testbed environment; small circles represent the positions of the six access points, dashed lines are steps. Refer to Fig. 8 to see where samples were measured.

every measurement consists of two physical coordinates followed by six integer values representing the RSSI reported by the PCMCIA driver for each Access Point as a 1-byte value. Typical values range from -102dBm (used when signal is lower than noise, or not present) to about -30dBm in close proximity to the antenna, and their discretization, due to hardware limitations, is 1dBm. Sample points were easily identifiable by the floor tiling, so accuracy of the spatial coordinates can be estimated in the order of a few centimeters (the experimenter needed to evaluate whether the PDA was on the vertical of the grid point).

7.2 Setup and parameter tuning

The experimental phase of this work begins with the determination of the best parameters for each algorithm. All tests for parameter tuning have been performed on the complete data set by leave-one-out position estimations. Recalling (beginning of Section 6) that ℓ is the number of samples, $\boldsymbol{x}_i$ is the

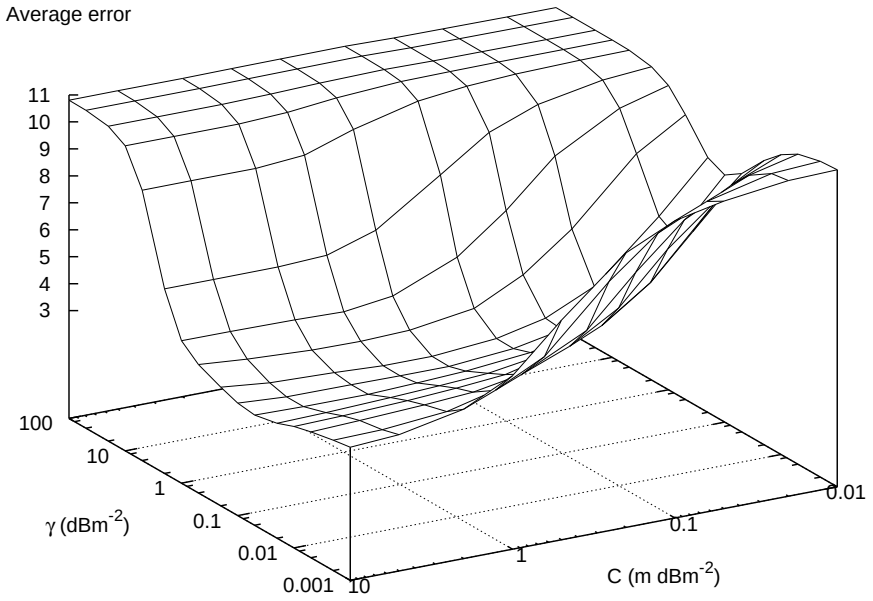

Fig. 3. Average position error for different values of parameters γ and C for the regression SVM.

i-th signal strength tuple and $\boldsymbol{y}_i$ is the corresponding desired outcome, the physical coordinates at which $\boldsymbol{x}_i$ was detected, the leave-one-out procedure works as follows:

- For $i = 1, \ldots, \ell$:

(1) Let $L = (\boldsymbol{y}_i, \boldsymbol{x}_i)$ be the i-th sample tuple;
(2) Let the test set be $\{L\}$ (a singleton);
(3) Put all other tuples in the training set (which is therefore composed of $\ell - 1$ tuples);
(4) Train the system with the training set;
(5) Test the trained system on tuple L and store the outcome (the estimated value for $\boldsymbol{y}_i$, which we shall refer to as $\bar{\boldsymbol{y}}_i$).

At the end of the procedure, every tuple has been used once for testing, and all other times for training. A set of location estimation outcomes $(\bar{\boldsymbol{y}}_i)_{i=1,\ldots,\ell}$ has been collected and it can be used for comparison with the true values $(\boldsymbol{y}_i)_{i=1,\ldots,\ell}$.

7.2.1 Support Vector Machine

As described in Section 5, for the regression (i.e., position estimation) problem the training and test cycles have been applied to two independent instances of

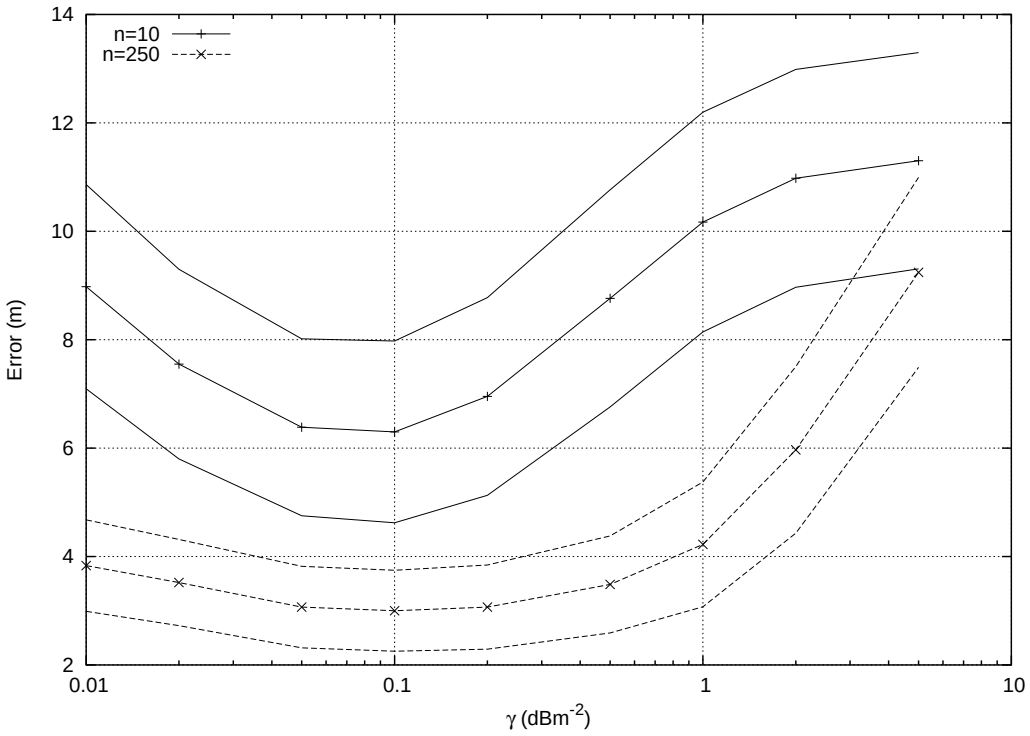

Fig. 4. Optimization of the γ parameter for different training set sizes for the regression SVM.

the learning machine, one per physical coordinate (common SVM implementations only admit scalar output). In other words, the training data contain six input columns (the received signal strength values) and one desired outcome (one of the two coordinates for regression, ± 1 for classification). The kernel function used with all tests is the Radial Basis Function (RBF, see Section 5). The structure of the problem suggests in fact that it is not linearly separable. Tests with polynomial kernels of degree up to $d = 4$ have shown a long training phase, up to tens of minutes, while the RBF kernel is much faster. Further tests have been performed to determine the optimal value for the parameter γ that is regulating the width of the Gaussians in the RBF function, the relative weight C of errors in the objective function (6) and the error tolerance term ε in the regression machine formulation (7).

Figure 3 shows the dependency of the average positioning error on parameters γ and C. The area around the minimum is rather flat, so that parameters are allowed to vary without a significant increase of the error, guaranteeing a high degree of robustness with respect to the detailed parameter values. Substantial error increases are verified when $C \to 0$, which amounts to removing the constraint violation penalty from the objective function in (7), and when γ is too large, so that radial basis functions are too narrow. When $C > 10$ the corresponding quadratic optimization tends to be very slow, and the error increases slightly. The values used for the regression tests are $\gamma = .1\mathrm{dBm}^{-2}$, $C = 1\mathrm{m}\,\mathrm{dBm}^{-2}$ and $\varepsilon = .8\mathrm{m}$.

The dependency of parameter γ on different training set sizes has been also tested, and some results are shown in Figure 4, where the average error is plotted for two different training set sizes: every point in the graph represents 50 tests where n random samples ($n = 10$ and $n = 250$) are extracted and used as training set; the resulting trained system is then tested on the remaining samples. The 95% confidence interval for the average error is shown. While the average error depends on the training sample size, the optimal value for the parameter γ remains approximately the same, with an experimental minimum at $\gamma = .1\text{dBm}^{-2}$ in both cases. This facts confirms the robustness of the system also with respect to variation of the training set size.
Experimental optimal values for the classification problem are $\gamma = .5\text{dBm}^{-2}$ and $C = 1\text{dBm}^{-2}$.

7.2.2 Weighted k Nearest Neighbors

For each training-test cycle, the algorithm was provided with the training set, whose $\ell - 1$ samples are composed of six radio strength values for the radio space and the corresponding two-coordinate outcome (or a yes/no outcome in the classification case) in the physical space. The only relevant parameter to tune is the number of nearest neighbors in the radio signal space to take into account for the average calculation. In our case, the number has been fixed experimentally to $k = 8$. However, for a large range of admissible values of k (5 to 15 approximately) the estimation error does only change by 1%. Comparison with the approach in (1), where the average is not weighted with distance in radio space, shows a 3% improvement when weights are taken into account.

7.2.3 Bayesian approach

In order to approximate the likelihood function and the integral calculations, physical space (a 40m $\times$ 35m area, slightly larger than the floor) has been covered by a 120 $\times$ 120-point mesh. The chosen radio propagation model is the linear loss with walls:

$$\boldsymbol{x} = b_0\mathbf{1} + b_1\boldsymbol{d}_{\text{AP}}(\boldsymbol{y}) + b_2\boldsymbol{w}_{\text{AP}}(\boldsymbol{y}),$$

where w_{AP} is the sum of the widths of all walls crossed by the line of sight between the access point and the user. While a single model for all base stations has been proposed by (14) for open-space location estimation in GSM networks, a 47% error reduction has been obtained in our environment by calculating independent sets of coefficients $(b_0^{(i)}, b_1^{(i)}, b_2^{(i)})$ for every access point $i = 1, \ldots, n$, so that the signal strength x_i from access point i is given by

$$x_i = b_0^{(i)} + b_1^{(i)} d_{\text{AP}i}(\boldsymbol{y}) + b_2^{(i)} w_{\text{AP}i}(\boldsymbol{y}).$$

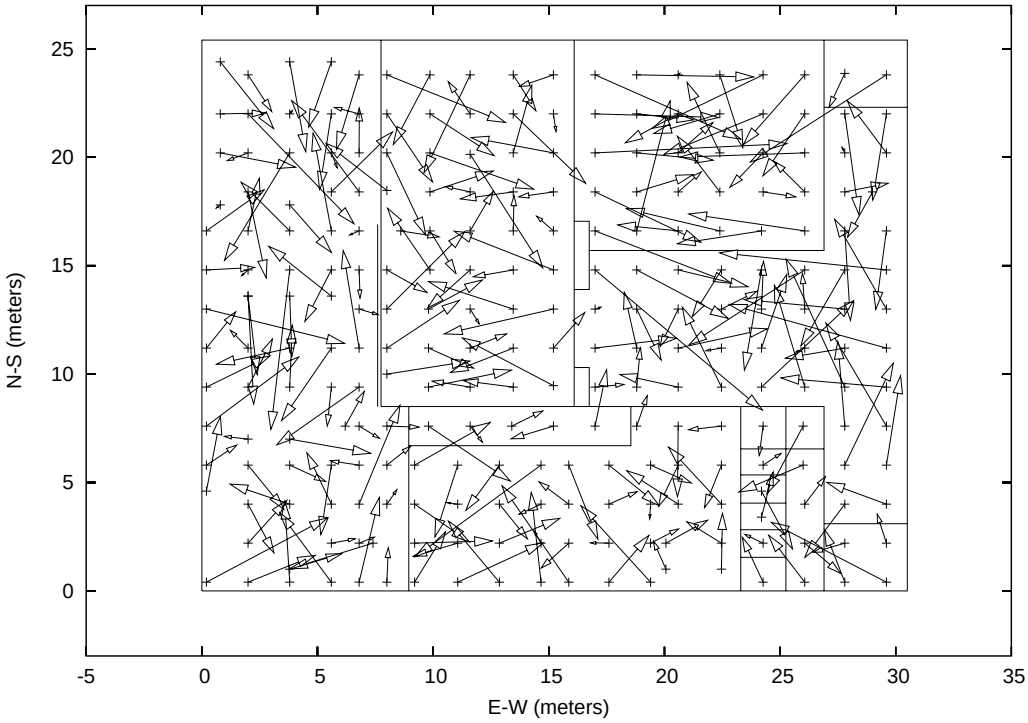

Fig. 5. Position estimation outcome by the SVM algorithm. Arrows originate on the true position and end on the estimated position.

and by using the probability distribution in order to calculate the average a posteriori position by equation (9), rather than maximizing it by treating it as a likelihood function, as in equation (10).

7.2.4 Multi-Layer Perceptron

A three-layer perceptron model is used, where the first layer (input) has six neurons, the second (hidden) layer has 8 neurons and the third (output) has two neurons in the regression problem and one in the classification. The transfer function of the hidden neurons is a sigmoid with $(0, 1)$ output. The transfer function of the output layer is linear in the regression case, a sigmoid with $(-0.5, 0.5)$ output in the classification case. Input values have been rescaled in the $[0, 1]$ range by dividing them by 100. To match the output sigmoid, the classification network is trained with outcomes ± 0.5.

7.3 The regression problem

Figure 5 shows the results of the leave-one-out position estimations, where every point in the data set is removed in turn, the remaining points are used as a training set, and the resulting trained system is tested on the removed

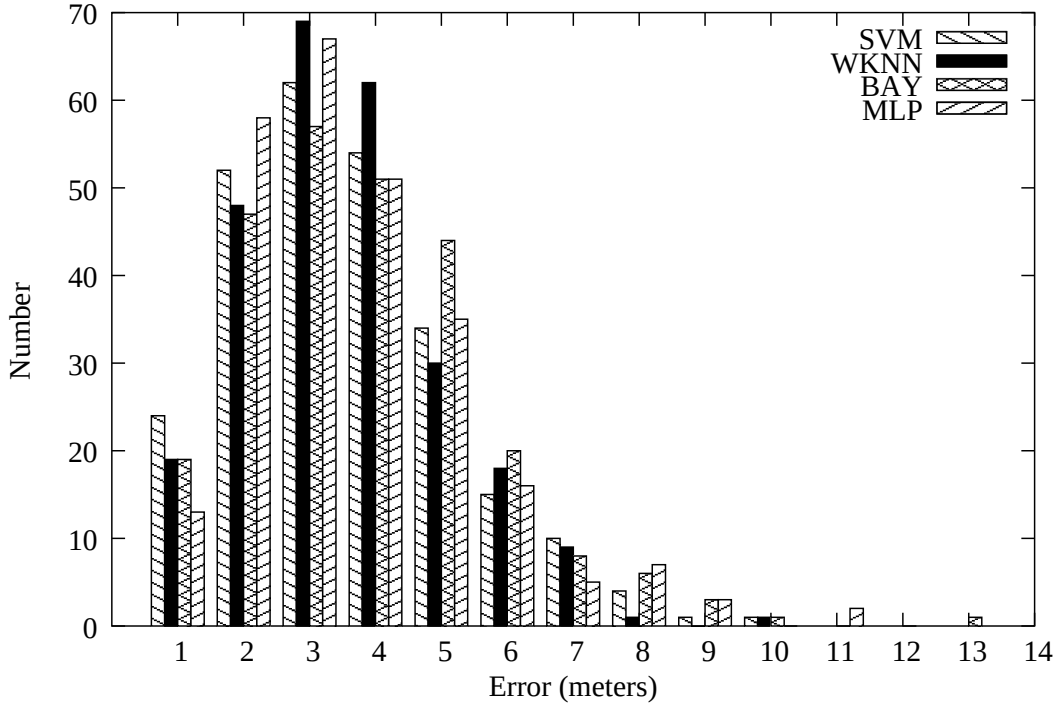

Fig. 6. Leave-one-out estimation error distribution for the regression algorithms

Table 1
Leave-one-out estimation error distribution: average and percentiles

Algorithm	Average	50th percentile	75th percentile	90th percentile	95th percentile
SVM	$\pm$	2.75	3.96	5.12	6.09
WKNN	3.06 ± 0.10	2.84	3.93	5.16	5.79
BAY	3.35 ± 0.11	3.04	4.39	5.61	6.61
MLP	3.18 ± 0.11	2.82	4.01	5.40	6.73
KNN (unweighted)	3.12 ± 0.10	2.91	3.98	5.21	6.10
BAY (no walls)	3.55 ± 0.12	3.30	4.56	5.87	6.82
BAY (single model)	4.97 ± 0.18	4.43	6.54	8.68	10.88
BAY (max. likelihood)	3.83 ± 0.15	3.42	5.14	6.83	8.42

point, for the SVM algorithm. The other algorithms have similar behavior. Every arrow represents the displacement between the true position (tail) and the computed location (head).

Figure 6 and Table 1 provide information about the error distribution of the four techniques. The four top data rows report experimental results about the four considered techniques. An interesting fact is that the Weighted k Nearest Neighbors and the Support Vector Machine outcomes are comparable, and

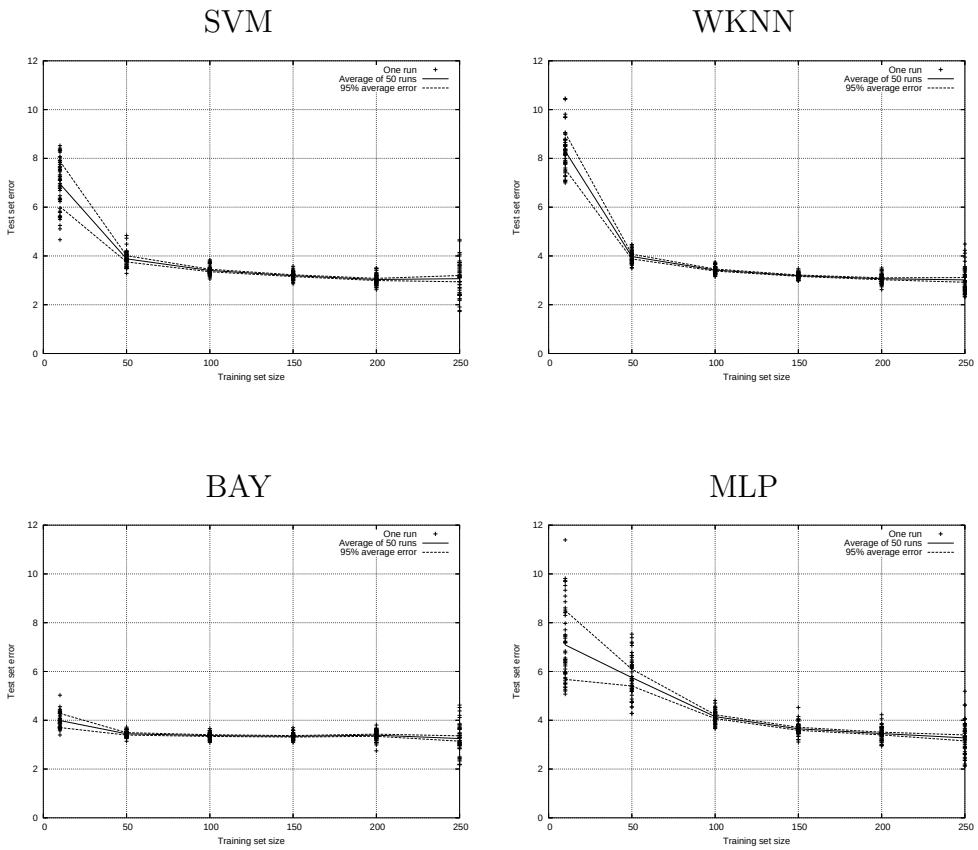

Fig. 7. Test set errors for different training set sizes. For every run, a random training set of the required size is extracted from the sample set, and all other points are used for testing; 50 runs are shown for every size; their average value is plotted with a line. The dashed lines delimit the 95% confidence interval for the average error.

they both outperform the two global models: neural networks and Bayesian inference models achieve a 10% performance degradation. While the average estimation precision is in the order of 3m, quantiles reported in Table 1 show that three measures out of four have an error below 4m, and only one in 20 has an error higher than 6m with the SVM and the WKNN technique.

The remaining rows of Table 1 show performance degradation when different choices are made for two of the algorithms. In particular, the KNN row shows the outcome of the k Nearest Neighbors algorithm if weights are not considered in the average calculation (8). The three bottom rows show the outcome of the BAY algorithm when some features are switched off. In particular, if walls are not considered in the linear model (open space assumption), then a 7% degradation is observed. If a single propagation model is used for all access points, then a 49% increase in error is detected. By using the conditional probability distribution as a likelihood function to be maximized (instead of a distribution to calculate a weighted position average) the error is increased by 15%.

Figure 7 shows the precision of the methods for different training set sizes. Sizes from 10 to 250 points are considered. For every size, 50 runs are executed. Every run consists in selecting a random training set of the given size from the measurement set; the systems are trained with it, then they are tested on the remaining points; the resulting error is plotted as a cross in the diagrams. The average of the 50 runs is plotted with a continuous line, while the 95% confidence interval for the true value of the error is represented by the two dashed lines.

Note that, while the error decreases as expected with sample size, the Bayesian model seems less sensitive to sample size, and an average error of 4m can be obtained with as little as 10 training points. This is due to usage of a simple linear model, computed via an LSQ technique, to estimate radio propagation, so that inaccuracies in radio propagation make it useless to improve the accuracy of the estimation by adding points to the model. On the contrary, the neural network shows a more gradual slope, and achieving the same precision requires more than 100 training points.

This consideration would make BAY the algorithm of choice when the user does not want to spend a long time in training the algorithm, so that a small number of points are available for training. However, as shown in Section 7.5, the algorithm is very slow, in that it requires computing the conditional probability distribution in a grid of points. Moreover, the positions of access points and the topography of the environment must be known in advance.

On the opposite, even though they require more training points, the other algorithms soon achieve better precisions and are computationally more affordable. In addition, no knowledge about the environment is required.

7.4 The classification problem

The basic classification problem solved by Support Vector Machines has two possible outcomes, for instance being inside or outside a room. The more general *labeling* problem consists of attributing a label (number) to each room, and to tag every measurement with these values. To solve it, for each room a different SVM classifier must be trained. The training set outcome is $+1$ for points inside the room, -1 for points that are outside. To label an unknown n-tuple of radio measurements, it is submitted to all SVMs, and the room whose SVM shows the highest outcome is selected. In most cases, only one of the outcomes is positive, but many uncertain cases may arise.

The same technique can be used for neural networks, by training one neural network per room with one output neuron. Actually, a single neural network with as many output neurons as rooms could be used. However, this is just a

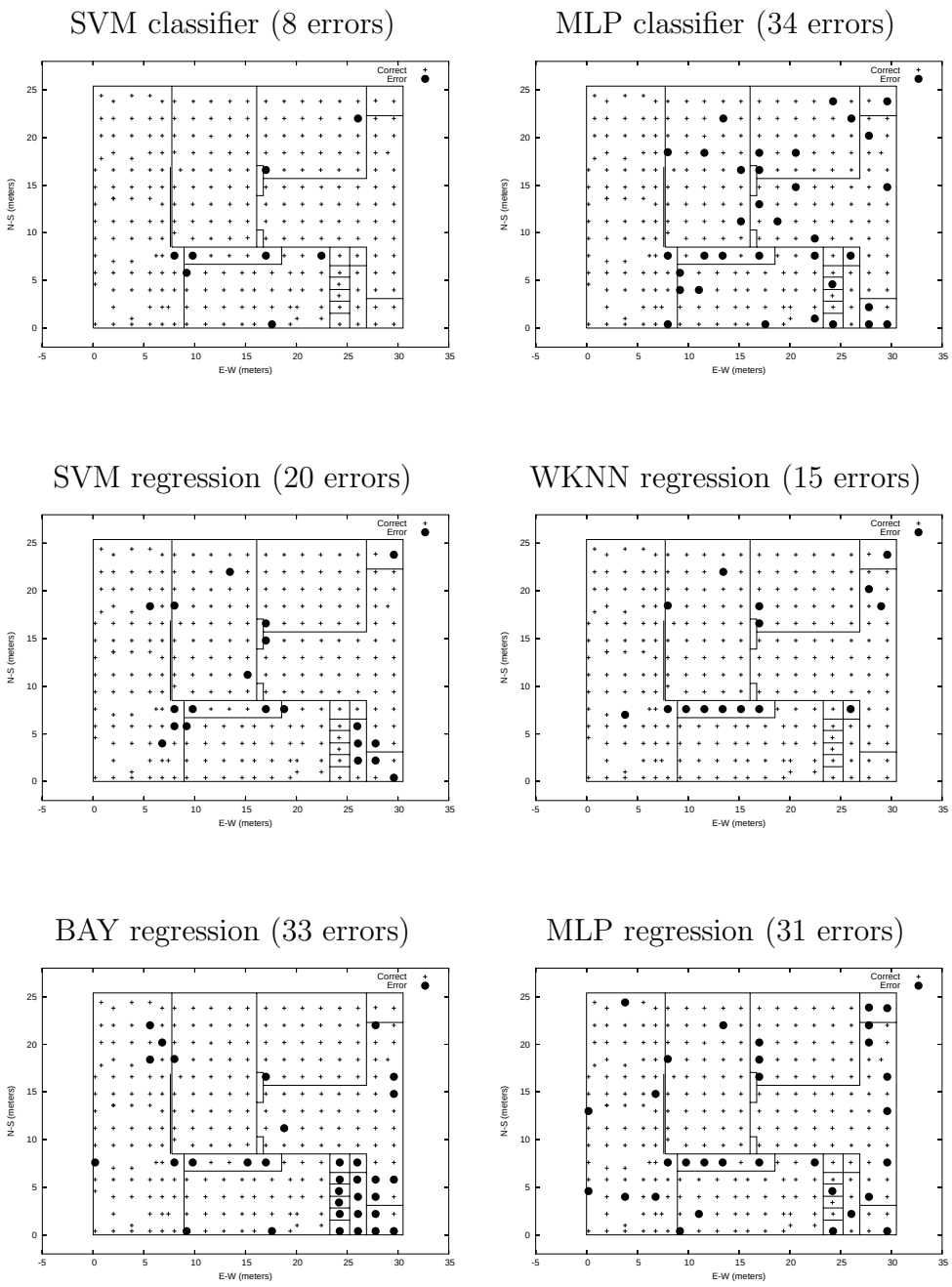

Fig. 8. Classification of samples according to room. In the top row, the classification outcome of SVM (left) and MLP (right) in native classification mode.

special case of the multiple networks where corresponding input weights are forced to be equal.

A general technique, that can be applied to all regression algorithms, is post-processing of the regression outcome, so that every point is classified according to the room containing its estimated coordinates. This method can be applied to all four techniques.

Table 2
Estimation phase times (seconds)

Algorithm	50 tests	5050 tests	Difference (5000 tests)
SVM	0.02	1.40	1.38
WKNN	0.01	0.86	0.85
BAY	1.97	131.03	129.06
MLP	0.00	0.14	0.14

Figure 8 shows the outcome of the classification algorithms on a leave-one-out test, where correctly classified sample points are shown as crosses and errors are reported as black dots. In the four regression algorithms, black dots correspond precisely to the tails of wall-crossing arrows of Figure vectors. The Support Vector Machine as native classification engine significantly outperforms all other classification algorithms.

In this test, a total of 7 rooms are identified, corresponding to the area labeled in the map (Figure 2). The WC area is considered a single room. Note that for all algorithms errors are usually found along the room borders, or in places where signal strength values are rather low. A better coverage, or a better disposition of the access points, will probably improve the performance of all algorithms.

7.5 Benchmarks

Three test sets have been performed. The first set measures the execution time, and is aimed at comparing the four heuristics. The second and third sets, monitoring power consumption estimation and bandwidth respectively, inspect the actual impact of passive AP scans on battery use and on the normal network activity.

To have a complete comparison of the four heuristics, their execution times have been compared on a large test set. Out of the complete 257-point sample set, 207 have been randomly extracted and used as a training set. The remaining 50 points have been used for testing. To obtain a large test set, they have been replicated 101 times, so that a large 5050-point test file as been obtained. To remove the time overhead due to configuration loading and preprocessing, two tests have been performed. The first on the original 50-point set, the other on the large set.

Table 2 reports the execution times in seconds for every algorithm and both test sets on the benchmarking machine (see Section 7.1). The last column shows the difference between the two testing runs, so that the initialization

Table 3
Absorbed power for various operating modes

Action	Power (battery% per second)	Network lifetime
Idle		
1. Wi-Fi interface off	0.0062	3h48'
2. Wi-Fi interface on and associated	0.0101	2h20'
AP scan enabled		
3. Scanning for a non existing ESSID	0.0137	1h43'
4. Localization (WKNN) once per second	0.0135	1h45'
5. Same as 4, with moderate network activity	0.0135	1h45'
6. Localization (WKNN) once per minute	0.0134	1h46'
7. Full network activity	0.0145	1h38'

overhead is removed and the net estimation time for 5000 points is reported. As expected, the Bayesian inference model, requiring the evaluation of the conditional probability distribution on a grid of points, has proved much slower than the other methods. The neural network, on the contrary, is much faster, because it only requires straightforward calculations. The execution time of the Support Vector Machine and the Weighted k Nearest Neighbors algorithms are at a larger order of magnitude, but their higher precision justifies them.

While all reported times are acceptable for normal operations, usage with a mobile device discourages a heavy algorithm such as BAY. In fact, lower processor speeds may render this approach impractical and too consuming in terms of memory and CPU load.

Experiments on power consumption were performed on an iPAQ h5450 PDA with inbuilt Wi-Fi interface, running the Microsoft PocketPC 2003 operating system. All tests have been run with the internal battery alone, backlight off and default power settings for at least 1500 seconds. Power consumption has been evaluated by using the appropriate system calls returning the battery status. These estimates are used by the system to decide when the Wi-Fi hardware must be disabled, so they actually depict the expected lifetime of the networking activity.

Table 3 reports battery power absorption corresponding to different system activities. Power measures are given in terms of battery percentage per second; the nominal capacity of the battery is not declared by the manufacturer, so conversion to a more suitable unit would be inaccurate and has been avoided. The first column describes the activity of the PDA, the second the correspond-

ing average consumption during a 1500-seconds run, the third the estimated networking session lifetime in the hypothesis that Wi-Fi hardware shall be disabled by the system at about 15% remaining capacity. Line 1 of the table refers to the idle PDA with network card switched off, in line 2 the network interface is on and associated to an AP, but no packet is transmitted or received, so the card is held in idle mode. Lines 3–6 report battery use with different network activities: card driver performing a continuous scan in search of a non existing ESSID (line 3), RSSI scan and WKNN localization once per second (line 4), the same with moderate network activity (a 10KB web page reloaded every 15 seconds, line 5), RSSI scan and WKNN localization once per minute (line 6). Note that power consumption is very similar for all sorts of network activity. Line 7, finally, shows power consumption when the wireless link is used at full capacity by a TCP downstream. The results suggest that the Wi-Fi hardware used for the experiments, when active, does not suffer from periodic scanning activity. The additional power consumption caused by the WKNN localization algorithm is negligible.

Last, the full-bandwidth TCP downstream has been monitored with and without scanning activity. For this purpose a small Wi-Fi network has been set up in a zone with no interference from other APs. A server accepting incoming TCP connections has been connected to the wired LAN and a client program that receives and discards packets has been run on the PDA. Without the AP scanning program, an average throughput of 2560kbps has been recorded. When the AP scanning was set at one scan per second, the throughput decreased to about 2400kbps, amounting to a 7% difference, which is reasonably small to justify the employment of RSSI-based localization techniques.

8 Conclusions

A new location discovery technique based on Support Vector Machines has been introduced along with the underlying statistical learning theory concepts. This technique can be used in its regression version to estimate the location of a mobile user, and as a classification engine to decide the area, for example the room, the user is currently in.

An experimental testbed setup has been described, and the proposed technique has been compared with three other algorithms presented in scientific literature. All comparisons have been performed on the same data set.

The Support Vector Machine algorithm displays a very low error rate when used as classifier, and it outperforms all other techniques in the described experiments, although when used for regression (spatial localization), its results closely match those of another effective technique, the Weighted k Nearest

Neighbors.

This paper is focused on a Wi-Fi system. However, the same techniques can be applied in principle to every wireless mobile transceiver, such as a cellular phone or a Bluetooth device, provided that data about the signals received from multiple fixed stations can be accessed.

Acknowledgements

The authors wish to thank Andrea Delai for his work on collecting the test samples and for developing software for the Bayesian approach in his thesis for the *laurea* degree.

References

[1] P. Bahl, V. N. Padmanabhan, RADAR: An in-building RF-based user location and tracking system, in: Proceedings of IEEE INFOCOM 2000, 2000, pp. 775–784.

[2] A. M. Ladd, K. E. Bekris, G. Marceau, A. Rudys, L. E. Kavraki, D. S. Wallach, Robotics-based location sensing using wireless ethernet, Tech. Rep. TR02-393, Department of Computer Science, Rice University (2002).

[3] T. Roos, P. Myllymäki, H. Tirri, P. Misikangas, J. Sievänen, A probabilistic approach to WLAN user location estimation, International Journal of Wireless Information Networks 9 (3).

[4] R. Battiti, A. Villani, T. Le Nhat, Neural network models for intelligent networks: deriving the location from signal patterns, in: Proceedings of AINS2002, UCLA, 2002.

[5] M. Brunato, Csaba Kiss Kalló, Transparent location fingerprinting for wireless services, in: Proceedings of Med-Hoc-Net 2002, Cagliari, Italy, 2002.

[6] R. Want, A. Hopper, V. Falcao, J. Gibbons, The active badge location system, ACM Transaction on Information Systems 10 (1) (1992) 91–102.

[7] A. Harter, A. Hopper, A distributed location system for the active office, IEEE Network 6 (1) (1994) 62–70.

[8] A. Ward, A. Jones, A. Hopper, A new location technique for the active office, IEEE Personal Communications 4 (5) (1997) 42–47.

[9] A. Harter, A. Hopper, P. Steggles, A. Ward, P. Webster, The anatomy of a context-aware application, in: Proceedings of MOBICOM 1999, 1999, pp. 59–68.

[10] N. B. Priyantha, A. Chakraborty, H. Balakrishnan, The cricket location-support system, in: MOBICOM 2000, 2000, pp. 32–43.

[11] J. Werb, C. Lanzl, Designing a positioning system for finding things and people indoors, IEEE Spectrum 35 (9) (1998) 71–78.

[12] J. Hightower, G. Borriello, R. Want, SpotON: An Indoor 3D Location Sensing Technology Based on RF Signal Strength, The University of Washington, Technical Report: UW-CSE 2000-02-02 (Feb. 2000).

[13] M. A. Youssef, A. Agrawala, A. U. Shankar, S. H. Noh, A probabilistic clustering-based indoor location determination system, Tech. Rep. CS-TR-4350, University of Maryland Computer Science Department (Mar. 2002).
URL `http://www.cs.umd.edu/Library/TRs/CS-TR-4350/CS-TR-4350.ps.Z`

[14] T. Roos, P. Myllymäki, H. Tirri, A statistical modeling approach to location estimation, IEEE Transactions on Mobile Computing 1 (1) (2002) 59–69.

[15] K. Pahlavan, A. Levesque, Wireless Information Networks, John Wiley & Sons, 1995.

[16] V. N. Vapnik, The Nature of Statistical Learning Theory, Springer Verlag, 1995.

[17] E. Osuna, R. Freund, F. Girosi, Support vector machines: Training and applications, Tech. Rep. AIM-1602, MIT Artificial Intelligence Laboratory and Center for Biological and Computational Learning (1997).

[18] C. J. C. Burges, A tutorial on support vector machines for pattern recognition, Data Mining and Knowledge Discovery 2 (2) (1998) 121–167.

[19] V. Vapnik, A. J. Chervonenkis, On the uniform convergence of relative frequencies of events to their probabilities, Theory Probab. Apl. 16 (1971) 264–280.

[20] T. Joachims, Making large-scale SVM learning practical, in: B. Schölkopf, C. J. C. Burges, A. J. Smola (Eds.), Advances in Kernel Methods - Support Vector Learning, MIT-Press, Cambridge, Mass., 1999, Ch. 11.

[21] A. J. Smola, B. Schvlkopf, A tutorial on support vector regression, Tech. Rep. NeuroCOLT NC-TR-98-030, Royal Holloway College, University of London, UK (1998).

[22] T. Joachims, Text categorization with support vector machines: Learning with many relevant features, in: Proceedings of the European Conference on Machine Learning, Springer, 1998.

[23] V. Strassen, Gaussian elimination is not optimal, Numerische Mathematik 13 (1969) 354–356.

[24] R. Battiti, First-and second-order methods for learning: Between steepest descent and newton's method, Neural Computation 4 (1992) 141–166.

[25] M. Karpinski, A. Macintyre, Polynomial bounds for VC dimension of sigmoidal neural networks, in: Proceedings of 27th ACM Symposium on Theory of Computing, 1995, pp. 200–208.

[26] S. Rüping, mySVM — Manual, Tech. Rep. Lehrstuhl In-

formatik 8, University of Dortmund, http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/ (2000).