Technical report DIT-02-0086, Università di Trento, October 2002

Statistical Learning Theory for Location Fingerprinting in Wireless LANs

Roberto Battiti Mauro Brunato Alessandro Villani
Università di Trento
Dipartimento di Informatica e Telecomunicazioni
via Sommarive 14, I-38050 Pantè di Povo (TN) — ITALY
Email: `battiti|brunato|villani@dit.unitn.it`

***Abstract*— In this paper, techniques and algorithms developed in the framework of statistical learning theory are analyzed and applied to the problem of determining the location of a wireless device by measuring the signal strengths 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 custom 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 the 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[1].

***Index Terms*— Context-aware computing, Location management, Wi-Fi, Mobile computing, Statistical learning theory**

I. 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, and to applications that take advantage of this information. The word *context* refers both to physical world data (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 piece of 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.

[1]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/`)

An important research target for context-aware applications 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 the traditional 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 is supported, and Wi-Fi networks can be extended to create "clouds of connectivity" inside the so-called *hotspots*, i.e. locations with high connection frequency such as office buildings or even the center of a town.

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 the intensity of a radio signal at a given point can only 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 a number of radio points 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 learning machine is proposed and tested for determining the location of a wireless device by using the intensity of the signal received from wireless access point stations in a Wi-Fi network.

This paper is organized as follows. Section II briefly reports previous work on location awareness and currently available implementations. Section III lists the assumptions on available user hardware and user requirements that are at the basis of the present work. Section IV describes the Statistical learning Theory approach and proposes the technique of Support Vector Machines. Section V briefly describes other approaches that have been implemented by the authors, and that are used for comparison in Section VI, where they are tested, benchmarked and discussed. Finally, conclusions are drawn in Section VII.

II. Previous work

Various technologies are being proposed to determine 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 make use of infrared (Active Badge [6], [7]) of ultrasound (Active Bat by AT&T [8], [9], Cricket [10]), or of radio signals that activate transponders attached to the item that must be located (3D-iD by PinPoint Corp. [11]). The SpotON project in Washington University [12] could perform 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) or by signal propagation modeling. Bayesian inference models and other probabilistic approaches are being used both for Wi-Fi products [2], [3] and for GSM telephone networks [13].

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].

III. System requirements and capabilities

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 should be *passive*: measurements should not require the active participation of the fixed infrastructure, but they should be 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.

Moreover, the mobile equipment and the network infrastructure are composed by off-the-shelf hardware, with no additional equipment. This choice allows 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, signal level 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 on a user sketch. However, location estimation can also be considered as a service provided by the network manager, so that this requirement is 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.

IV. Statistical learning theory

While a functional dependency of physical coordinates from received signal strengths does not exist (the same radio measurements could be received in two different locations, while in the same location different values can be detected in two subsequent measurements), a statistical relation can be investigated.

The search for a statistical dependency starts with data collection, where signal strengths from all access points are monitored from a set of known physical positions. Let us call $\mathcal{L}$ the set of location informations, i.e. the space of tuples representing information about physical location. Two cases shall be considered.

a) Regression problem: location data can be expressed as a d-uple of real coordinates where d, the dimension of physical space, may vary from 1 (e.g. position along a corridor) to 5 (position in three-dimensional space and orientation expressed in spherical coordinates); in this case, $\mathcal{L} = \mathbb{R}^d$.

b) Decision or classification problem: location data is a single variable from a two-valued set, usually $\{-1, 1\}$, the values meaning "outside" and "inside" a given area; in this case, $d = 1$ and $\mathcal{L} = \{-1, 1\}$.

In both cases, r independent radio access points are located in the working area and send continuous beaconing signals. In this paper, as can be seen in Section VI-A, both problems are considered; in particular, $d = 2$ in the regression problem (location on a planar map) and $r = 6$ (6 access points are being used).

The collected data can be seen as a sequence of ℓ tuples, each tuple representing the measurement of all radio signals from a single location:

$$(y_1, x_1), \dots, (y_\ell, x_\ell) \in \mathcal{L} \times \mathbb{R}^r,$$

where x represents the radio component, and y the spatial component of the measurement. This set of tuples shall be called the *training set*.

A. Classical learning theory

The purpose of statistical learning theory is to investigate the probabilistic dependency of a random variable Y from a random variable X, by inferring as much information as possible from the training set. In a general way, this amounts to looking for the unknown conditional probability distribution $P(y|x)$. In most practical cases, the target of the investigation is the functional dependency of the expected value for Y from x.

$$E(Y|X = x) = \int_{\mathcal{D}Y} \chi P(\chi|x) d\chi,$$

where $\mathcal{D}Y$ is the domain of the random variable Y, i.e. its value set. Usually, an approximation can be identified within a large family of functions of the form $y = f_\lambda(x)$, where $\lambda \in \Lambda$ is an abstract parameter, in some cases a real vector. The parametric set of function $\mathcal{H} = (f_\lambda)_{\lambda\in\Lambda}$ is usually called the *hypothesis space*.

A measure of how much a function result differs from the experimental value is called a *loss function* $L(y, f_\lambda(x))$. The choice of the loss function influences the metric of the problem, and different results can be expected with different choices of L. In the regression problem, examples for L are the absolute value of the difference, or its square; for classification, a simple expression for L is:

$$L(y, f_\lambda(x)) = \begin{cases} 0 & \text{if } y = f_\lambda(x) \\ 1 & \text{otherwise .} \end{cases}$$

Once the loss function L is given, its expected value is called the *risk functional*:

$$R(\lambda) = \int\limits_{\mathbb{R}^r\times\mathcal{L}} L(y, f_\lambda(x))P(y,x)dxdy, \tag{1}$$

where $P(y, x)$ is the joint probability distribution of x and y. While this probability distribution is unknown, the training set can be used to approximate the true risk functional (1) through its *empirical* version:

$$R_{\text{emp}}(\lambda) = \frac{1}{n}\sum_{i=1}^{n} L(y_i, f_\lambda(x_i)). \tag{2}$$

The *Empirical Risk Minimization Principle* is the foundation of the classical Fisher paradigm, that gives rise, for example, to Least Squares and Maximum Likelihood estimators. These methods suffer from the fact that the distribution shape, or the functional dependence, must be known in advance, so that only a small number of parameters must be determined. Moreover, although consistency results are proved, a large number of training tuples must be provided to have a valid estimate.

However, empirical risk minimization does not always ensure such approximation; an example of this possible divergence is the following “learning” algorithm:

Store all training set elements $(y_i, x_i)_{i=1,\dots,\ell}$. When asked for a given x^*, if $x^* = x_i$ for some i, then answer y_i; otherwise answer 0. (3)

If training tuples are injective, then this machine trivially reduces the empirical risk functional to 0 by learning to respond to training examples; however, it does not even approximate the problem of the risk functional minimization, unless *all* possible tuples are given in the training set (which is obviously impossible in case of infinite sets of values). In other words, this machine does not *generalize*.

Thus, there is no a priori relation between the risk functional value and its empirical counterpart.

B. A general approach: structural risk minimization

To state a relation between the empirical risk functional and the actual risk, upper bounds can be found by generalizing the learning paradigm and introducing the *structural risk minimization* concept.

Consider a chain of hypothesis spaces on a fixed parameter space Λ

$$\mathcal{H}_1 \subset \mathcal{H}_2 \subset \cdots \subset \mathcal{H}_k \subset \dots, \tag{4}$$

where $\mathcal{H}_i = (f^i_\lambda)_{\lambda\in\Lambda}$; a *learning machine* on this space is an algorithm that takes as input the sequence of ℓ training tuples, and outputs an index $\hat{k}$ and a value for the abstract parameter $\hat{\lambda}$; in other words, the learning machine chooses a particular function $f^{\hat{k}}_{\hat{\lambda}}$ based on the training set. This definition includes all classical inference methods, where the chain (4) consists of a single hypothesis space and Λ is the set of parameters of the function family.

Vapnik [14] shows that there is a function $\Phi(\cdot,\cdot)$ such that, given a hypothesis space $\mathcal{H}$ and an arbitrarily small probability η, it is possible to determine a constant h so that, however small the probability η is, the bound

$$R(\lambda) \leq R_{\text{emp}}(\lambda) + \Phi\left(\frac{\ell}{h}, \frac{\ln\eta}{\ell}\right) \tag{5}$$

is valid with probability $1 - \eta$. The rightmost term $\Phi(\ell/h, \ln\eta/\ell)$ is the amplitude of the confidence interval, and for every value of the probability parameter it has the following properties:

$$\lim_{t\to 0^+} \Phi(t,\cdot) = +\infty, \qquad \lim_{t\to+\infty} \Phi(t,\cdot) = 0.$$

The constant h is called the *Vapnik-Chervonenkis dimension* (VC-dimension for short) of the function set. When considering classification problems (so that the hypothesis space is a set of indicator functions), the VC-dimension of a hypothesis space is the largest number h of examples that can be *shattered*, i.e. separated in all 2^h ways, by the functions in the set, and it has the property that if $\mathcal{H} \subset \mathcal{K}$, then their VC-dimensions maintain the same order: $h(\mathcal{H}) \leq h(\mathcal{K})$. Thus, h measures the ability of a function set to arbitrarily discriminate examples. A proper generalization exists for the general regression problem.

The bound (5) reflects the fact that even if the empirical risk can be reduced, assumptions on the value of the actual risk (and, accordingly, on the generalization properties of the machine) can only be made after a careful selection of the hypothesis space. For example, the VC-dimension of the hypothesis space defined by algorithm (3) is $h = +\infty$ (given the learning technique, any finite set of points can be discriminated in all possible ways), so no upper bound holds.

A learning machine is said to implement the *Structural Risk Minimization principle* (SRM) if both summands in the right-hand side of bound (5), i.e. the empirical risk and the confidence interval amplitude, are controlled by appropriate choice of the hypothesis space $\mathcal{H}$ and of the parameter λ.

C. Optimal separating hyperplanes

In the following we shall consider a classification problem, where tuples are in the form $(y, x) \in \{-1, +1\} \times \mathbb{R}^r$. The training set $S = \{(y_i, x_i) | i = 1, \dots, \ell\}$ is *linearly separable* if there is a hyperplane in $\mathbb{R}^r$ separating all x_i such that $y_i = +1$

from all x_i such that $y_i = -1$. In mathematical terms, there exists a vector $w \in \mathbb{R}^r$ and a constant $b \in \mathbb{R}$ such that, for every $i = 1, \dots, \ell$ y_i has the same sign of $w \cdot x_i + b$:

$$\forall i = 1, \dots, \ell \qquad y_i(w \cdot x_i) + b > 0.$$

The *separating hyperplane* has equation $w \cdot x + b = 0$, and in general it is not unique; however, it is said to be *optimal* if it has the largest possible distance from the training set (i.e. from the closest training point).

Every separating hyperplane is determined by the vector w and the constant b; this representation is unique modulo a nonzero multiplicative factor. A unique *canonical* form can be defined by requiring the condition

$$\min_{i=1,\dots,\ell} |w \cdot x_i + b| = 1.$$

Thus, finding the canonical hyperplane can be reduced to the following quadratic programming problem:

$$\begin{array}{ll} \text{Minimize}_{w,b} & \frac{1}{2}\|w\|^2 \\ \text{subject to} & y_i(w \cdot x_i + b) \geq 1, \quad i = 1, \dots, \ell. \end{array} \tag{6}$$

In this case the hypothesis space is the set of indicators discriminated by hyperplanes:

$$\mathcal{H} = \{\text{sign}(w \cdot x + b) | (w, b) \in \Lambda\},$$

and the parameter set is

$$\Lambda = \{(w, b) | w \in \mathbb{R}^r, b \in \mathbb{R}\}.$$

The optimal separating hyperplane problem (6) can be solved by finding the saddle point (minimal with respect to w and b, maximal with respect to α_i) of the Lagrange functional

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{\ell} \alpha_i \left(y_i(x \cdot w + b) - 1\right), \tag{7}$$

where the α_i are the (positive) Lagrange multipliers. Let (w^*, b^*, α^*) the saddle point.

We call *support vectors* those training vectors for which the constraint in (6) holds as an equality. These correspond to the nonzero α_i^* multipliers. Support vectors are the only vectors necessary to define the hyperplane:

$$w^* = \sum_{i:\alpha_i^* \neq 0} y_i \alpha_i^* x_i.$$

Therefore the optimal hyperplane has equation

$$\sum_{i:\alpha_i^* \neq 0} y_i \alpha_i^* x_i \cdot x - b^* = 0. \tag{8}$$

An important theorem, defining a bound on the VC dimension of the canonical hyperplanes hypothesis space, is the following:

Theorem 1: Let R be the radius of a circle containing all points in the training set, and let A be an upper bound on $\|w\|$. Then the VC-dimension h of the hypothesis space is bounded by

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

To take advantage of this result, a new constraint

$$w \cdot w \leq c_r \tag{9}$$

must be added to the optimization problem (6). Moreover, the whole concept can be extended to non-separable training sets if non-negative variables ξ_i are introduced so that hyperplane constraints become

$$y_i(w \cdot x_i + b) \geq 1 - \xi_i$$

and the cost functional takes errors into account. The problem becomes:

$$\begin{array}{ll} \text{Minimize}_{w,b} & \frac{1}{2}\|w\|^2 + C\left(\sum_{i=1}^{\ell} \xi_i\right) \\ \text{subject to} & \begin{cases} y_i(w \cdot x_i + b) \geq 1 - \xi_i & i = 1, \dots, \ell \\ \xi_i \geq 0 & i = 1, \dots, \ell \\ \|w\|^2 \leq c_r, \end{cases} \end{array} \tag{10}$$

where $C > 0$ and $c_r > 0$ are given constants.

D. Support vector machines

Of course, a linear separating function, however generalized with an error minimization term, is far too restrictive for most real-world problems. The idea of the *Support Vector Machine* (SVM) is to map the input vectors into a *feature space* with a higher number of dimensions, and to find an optimal separating hyperplane in the feature space. For example, points in a two-dimensional space $(x_1, x_2) \in \mathbb{R}^2$ may be mapped into the 5-dimensional space $(x_1, x_2, x_1x_2, x_1^2, x_2^2) \in \mathbb{R}^5$; a separating hyperplane in this larger space will correspond to a conic separator in $\mathbb{R}^2$.

The optimal separating hyperplane in the feature space need not be described explicitly [15]. In fact, from equation (8), as well as from its generalization to non-separable data (not shown in this paper for simplicity), all we need to do is to calculate dot products of vector transforms, which amounts to general inner products $K(\cdot, \cdot)$ in $\mathbb{R}^r$.

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

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

However, 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. In 1997 Osuna, Freund and Girosi [16] introduced a decomposition method where the optimization problem is split in an active and an inactive set. Later, Joachims [17] introduced 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.

E. Support vector machines for regression

A modification of optimization problem (10) suitable for regression purposes can be obtained if the optimal separating hyperplane is not used as an indicator function (through the sign function), but directly: in this case the hypothesis space is the set of functions

$$\mathcal{H} = \{w \cdot x + b | (w, b) \in \Lambda\},$$

$$\Lambda = \{(w, b) | w \in \mathbb{R}^r, b \in \mathbb{R}\}.$$

In order to disregard small discrepancies, the empirical risk functional must be calculated for an ε-insensitive loss function:

$$L_\varepsilon(y, f_\lambda(x)) = \max\{\varepsilon, |y - f_\lambda(x)|\},$$

where the loss is equal to ε whenever the discrepancy between the expected and the computed value are less than ε itself.

Last, as in the previous case, reduction of the VC-dimension is attained by forcing constraint (9). Under this constraint we want to minimize the empirical risk functional

$$R_{\text{emp}}(w, b) = \frac{1}{\ell} \sum_{i=1}^{\ell} L_\varepsilon(y_i, x_i \cdot w + b),$$

and this is equivalent to solving the following optimization problem:

$$\begin{array}{ll} \text{Minimize}_{w,b} & \frac{1}{2}\|w\|^2 + C\left(\sum_{i=1}^{\ell} \xi_i^* + \sum_{i=1}^{\ell} \xi_i\right) \\ \text{subject to} & \begin{cases} y_i - w \cdot x_i - b \leq \varepsilon - \xi_i^* & i = 1, \ldots, \ell \\ w \cdot x_i + b - y_i \leq \varepsilon - \xi_i & i = 1, \ldots, \ell \\ \xi_i^* \geq 0 & i = 1, \ldots, \ell \\ \xi_i \geq 0 & i = 1, \ldots, \ell \\ \|w\|^2 \leq c_r, \end{cases} \end{array} \tag{11}$$

where the slack variables ξ_i and ξ_i^* for handling positive and negative discrepancies have been introduced.

Support Vector Machines are being successfully used for pattern recognition [14], [18], classification of text and of web pages [19].

V. Other approaches

To effectively evaluate the statistical learning approach, other techniques have been selected from the literature 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 (y_i, x_i), $i = 1, \ldots, \ell$, x_i being an array of r radio signal intensity values and y_i the location information, either a pair of coordinates (regression problem) or ± 1 (classification).

A. Weighted k Nearest Neighbors

Let $k \leq \ell$ be a fixed positive integer; consider a measured signal strength array 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 $x_{i_1}, \ldots, x_{i_k}$ that are nearest (according to a given radio-space metric) to the given x vector.
2) Calculate the estimated position information y by the following average, weighted with the inverse of the distance between signal strengths:

$$y = \frac{\displaystyle\sum_{j=1}^{k} \frac{1}{d(x_{i_j}, ss) + d_0} \cdot y_{i_j}}{\displaystyle\sum_{j=1}^{k} \frac{1}{d(x_{i_j}, x) + d_0}}, \tag{12}$$

where $d(x_i, x)$ is the radio distance between the two triplets (for example the Euclidean distance) measured in dBm, and d_0 is a small real constant ($d_0 = .01dBm$ in our tests) used to avoid division by zero.

While the above algorithm contains a weighted average, the technique was first proposed in [1] without using distance-dependent weights. It is simple to implement, and results in Section VI 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.

1) Learning phase complexity: The learning phase is rather simple: just store all examples. Complexity is $O(\ell(r + d))$ in time (all tuples must be input) and $O(\ell(r + d))$ in space (all tuples must be stored).

2) Estimation phase complexity: All tuples must be scanned for searching the k nearest ones. We execute a thorough scan of the training set, and every insertion in the set of k nearest indices is performed through a binary search; computing the distance in the radio space has complexity proportional to the number r of radio coordinates, computing the average requires d operations, so the total worst-case time complexity is $O(\ell r \log k + d)$.

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.

B. Probabilistic Analysis

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 [13]. 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 model is necessary.

Once the radio propagation model is determined, this can be used to calculate the conditional probability distribution $p(x|y)$ of an r-uple x of radio signal intensity values, given a d-uple 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, thus implementing a Maximum Likelihood location estimator.

If empirical distributions cannot be detected for a large set of points, then an analytical radio model can be stated in the form of a linear dependence:

$$x = \sum_{i=0}^{c-1} b_i \zeta_i, \tag{13}$$

where c is the number of unknown parameters (coefficients) $b_0, \dots, b_{c-1}$ of the model and ζ_i is some suitable transform of the training set dependent variable vector 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:

$$\zeta_0 = 1, \qquad \zeta_1 = \log d_{\mathrm{AP}}(y), \qquad \zeta_2 = w_{\mathrm{AP}}(y),$$

where $d_{\mathrm{AP}}(y)$ represents the distance of the physical point y from the access point, $w_{\mathrm{AP}}(y)$ 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 equation (13) with these transforms leads to a possible radio propagation model:

$$x = b_0 + b_1 \log d_{\mathrm{AP}}(y) + w_{\mathrm{AP}}(y).$$

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 (13), solution of a system in the form

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

is required, where M is an $r\ell \times c$ matrix, and y is a column vector with $r\ell$ components. The number of rows is given by the number of training samples ℓ, each carrying information about the signal strength of r access points. Straightforward matrix calculation algorithms are used, so that calculating $M^T M$ requires the evaluation of all mutual dot products in the c columns of M, each having $r\ell$ elements, amounting to $O(c^2 r\ell)$ 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 y require $O(c^2 r\ell)$ and $O(r\ell)$ operations respectively. To achieve an acceptable confidence interval, the number $r\ell$ 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 $O(c^2 r\ell)$.
Similar values arise from the second approach, where 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.
Lower exponents can be achieved by using optimized techniques for matrix inversion and multiplication, such as the Strassen algorithm.

2) Estimation phase complexity: This is the most consuming phase. In fact, a likelihood function on the physical space must be 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 r independent probability distributions must be calculated on a 2500-points grid.

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, maximum likelihood estimations) 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.

C. Neural Networks

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

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

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 [20] using second-order derivative information. Usage of neural networks in conjunction with localization problem was proposed in [4].

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 from 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.

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

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 [21] 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, $W = O(rH)$, so the upper bound is $O(r^2H^4)$. 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: $O(r^4H^8)$. While this latter bound can become quite large, both r and H are predetermined and usually low.

VI. Experimental results

In this section details shall be given about the sample collection and data analysis. For concise reference in figures and tables, compared algorithms shall be referred to as SVM (Support Vector Machine), WNN (Weighted k Nearest Neighbors), BAY (Bayesian a posteriori likelihood) and MLP (Multi-Layer Perceptron). The unweighted version of the WNN algorithm shall be referred to as KNN.

A. Setup

1) System and environment: The target system for our experiments is a wireless LAN using the IEEE802.11b (Wi-Fi) standard. The LAN is composed by 6 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 two-slot PCMCIA adapter and an ORiNOCO Silver card by Lucent Technologies. The target environment is the ground floor of a two-storey building, roughly 30m $\times$ 25m large. Its map is shown on Figure 1.

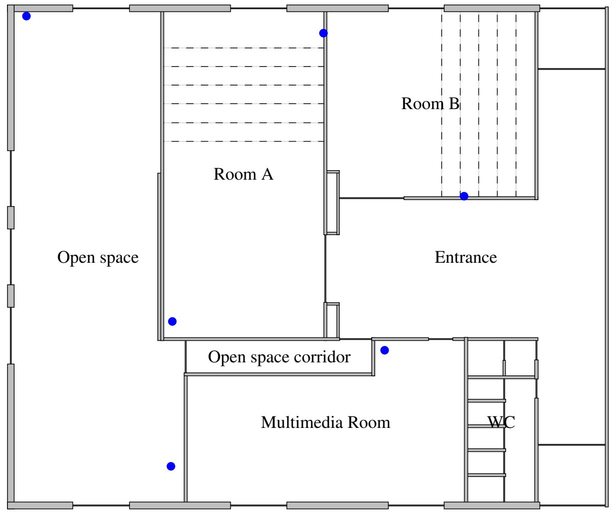

Fig. 1. Map of the testbed environment; small circles represent the positions of the 6 access points, dashed lines are steps.

2) Software tools: To test the Support Vector Machine algorithm, the `mySVM` implementation[22] 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.

3) Sample collection: The sample set used in the following experimental analysis is composed by 257 measurements throughout the floor. The positions where the samples have been measured can be seen in Fig. 5. A fully regular grid could not be followed due to the presence of various obstacles such as tables and pieces of furniture. Every measurement consists of 2 physical coordinates and 6 values representing the quality of the connection reported by the PCMCIA driver as a 1-byte value for each access point. Typical values range from 0dBm (used when signal is lower than noise, or not present) to 60$\div$70dBm in close proximity to the antenna, and their discretization is 1dBm.

B. Parameter tuning

The experimental phase of this work begins with the determination of the best parameters for the algorithms.

1) Support Vector Machine: The kernel function used with all tests is the Radial basis function (see Section IV-D); 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 both in training and in the testing phase. Further tests have been performed to determine the optimal value for the parameter in the RBF function $\gamma = .2\text{m}^{-2}$, the relative weight of errors in the objective function (10) $C = 1\text{m}^{-1}$ and the error tolerance term in the regression machine formulation (11) $\varepsilon = .8$m. Values for the classification problem are $\gamma = .5$ and $C = 1$.

2) Weighted k Nearest Neighbors: The only relevant parameter 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 2% improvement when weights are taken into account.

3) Maximum Likelihood: No numerical parameters have been considered in this heuristic, apart from space discretization: a $40\text{m} \times 35\text{m}$ area, slightly larger than the floor, has been covered by a 120×120-point mesh. The chosen radio propagation model is the linear loss with walls:

$$x = b_0 + b_1 d_{\text{AP}}(y) + b_2 w_{\text{AP}}(y),$$

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 [13] for open-space location estimation in GSM networks, a 47% error reduction has been obtained in our environment by calculating independent sets of coefficients for every access point, and by using the likelihood function as a probability distribution to calculate the average a posteriori position rather than maximizing it.

4) Neural Network: A three-layer perceptron model is used, where the first layer is the input (6 neurons), the second is the hidden layer (8 neurons) and the third is the output (two neurons in the regression problem, 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.

C. Regression

Figure 2 shows the results of the leave-one-out position estimations, where every point in the sample set is removed in turn, the remaining points are used as a training set, and the resulting trained system is tested on the removed point. Every arrow represents the displacement between the true position (tail) and the computed location (head). Note that if position is determined on the basis of other points (such is the case of Support Vector Machines and Weighted k Nearest Neighbors), locations on the border move toward the interior. On the other hand, if a global model is built (neural networks or probabilistic models) then estimations may also move to the outside of the sampled area.

Figure 3 and Table I 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, while the Weighted k Nearest Neighbors method always achieves the best average results, the Support Vector Machine outcome is comparable. The two global models, neural networks and probabilistic models, achieve a 10% performance degradation. While the average estimation precision is in the order of 3m, quantiles reported in Table I 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 WNN technique.

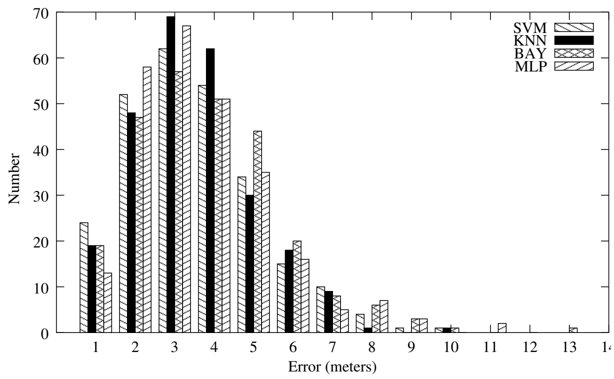

Fig. 3. Leave-one-out estimation error distribution for the regression algorithms

The remaining rows of Table I 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 (12). 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 4 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 probabilistic 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 VI-E, the algorithm is very slow, in that it requires computing the likelihood function 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 knowl-

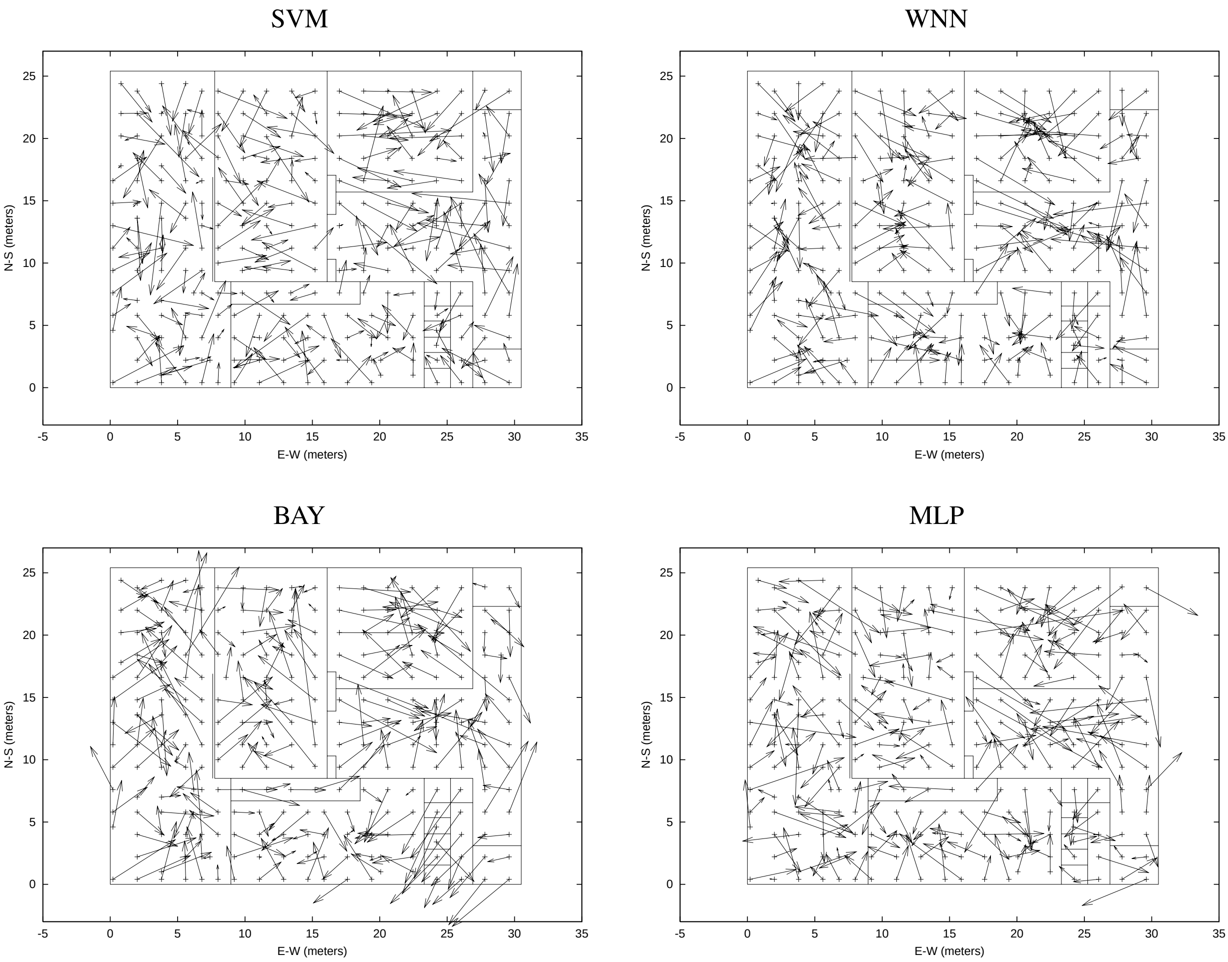

Fig. 2. Position estimation outcome by the four algorithms. Arrows originate on the true position and end on the estimated position.

TABLE I
LEAVE-ONE-OUT ESTIMATION ERROR DISTRIBUTION: AVERAGE AND PERCENTILES

Algorithm	Average	50th percentile	75th percentile	90th percentile	95th percentile
SVM	3.04 ± 0.10	2.75	3.96	5.12	6.09
WNN	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	3.99	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

edge about the environment is required.

D. Classification

The basic classification problem solved by Support Vector Machines is dichotomous, 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 r-uple 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 special case of the multiple networks where corresponding input weights are

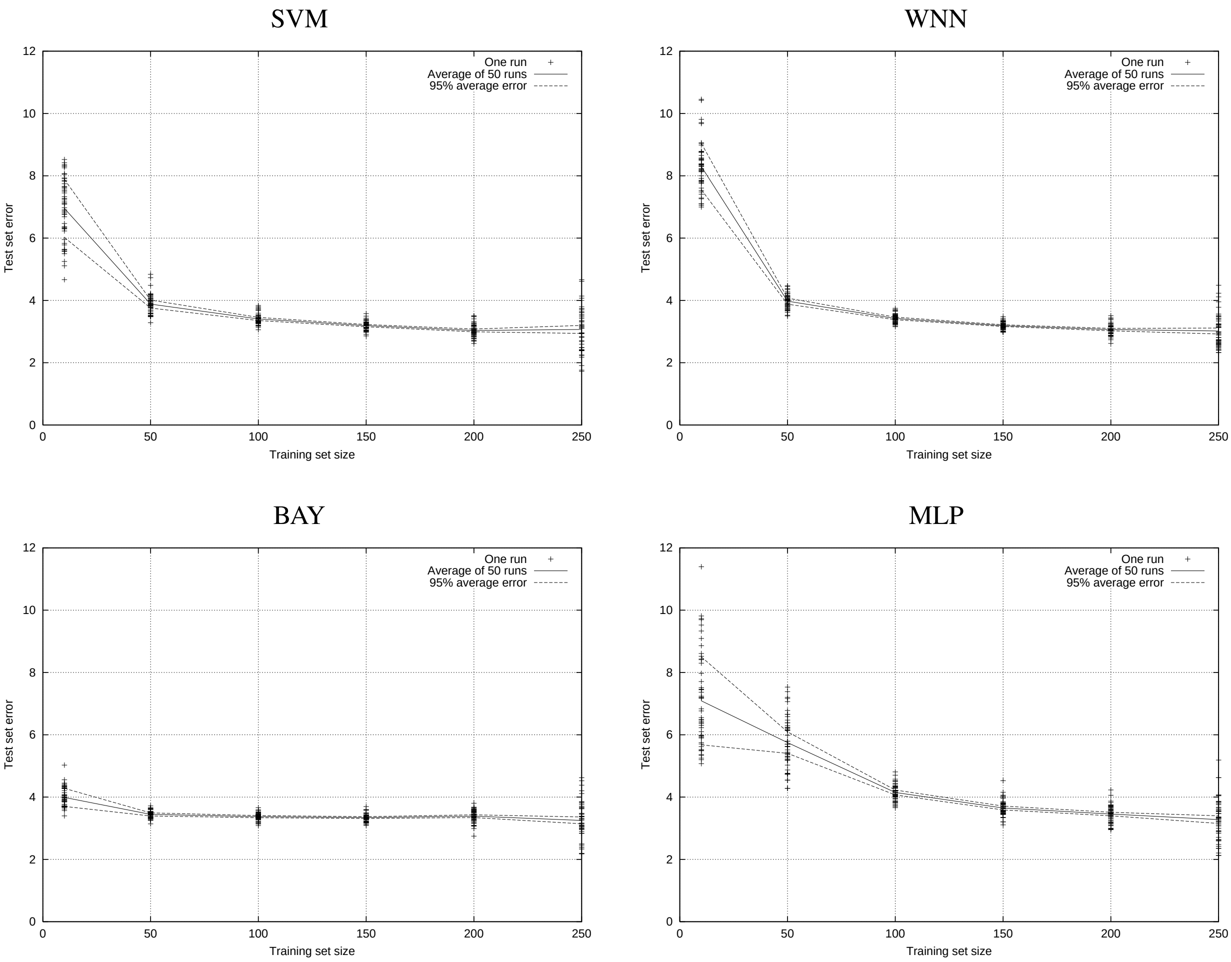

Fig. 4. Test set errors for different training set sizes. For every run, a random training set of the required size is extracted from the sample set, and all other points are used for testing; 50 runs are shown for every size; their average value is plotted with a line. The dashed lines delimit the 95% confidence interval for the average error.

forced to be equal.

A general technique, that can be applied to all regression algorithms, is postprocessing 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.

Figure 5 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 1). 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 strengths are rather low. A better coverage, or a better disposition of the access points, will probably improve the performance of all algorithms.

TABLE II
ESTIMATION PHASE TIMES (SECONDS)

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

E. Execution times

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 II reports the execution times in seconds for every algo-

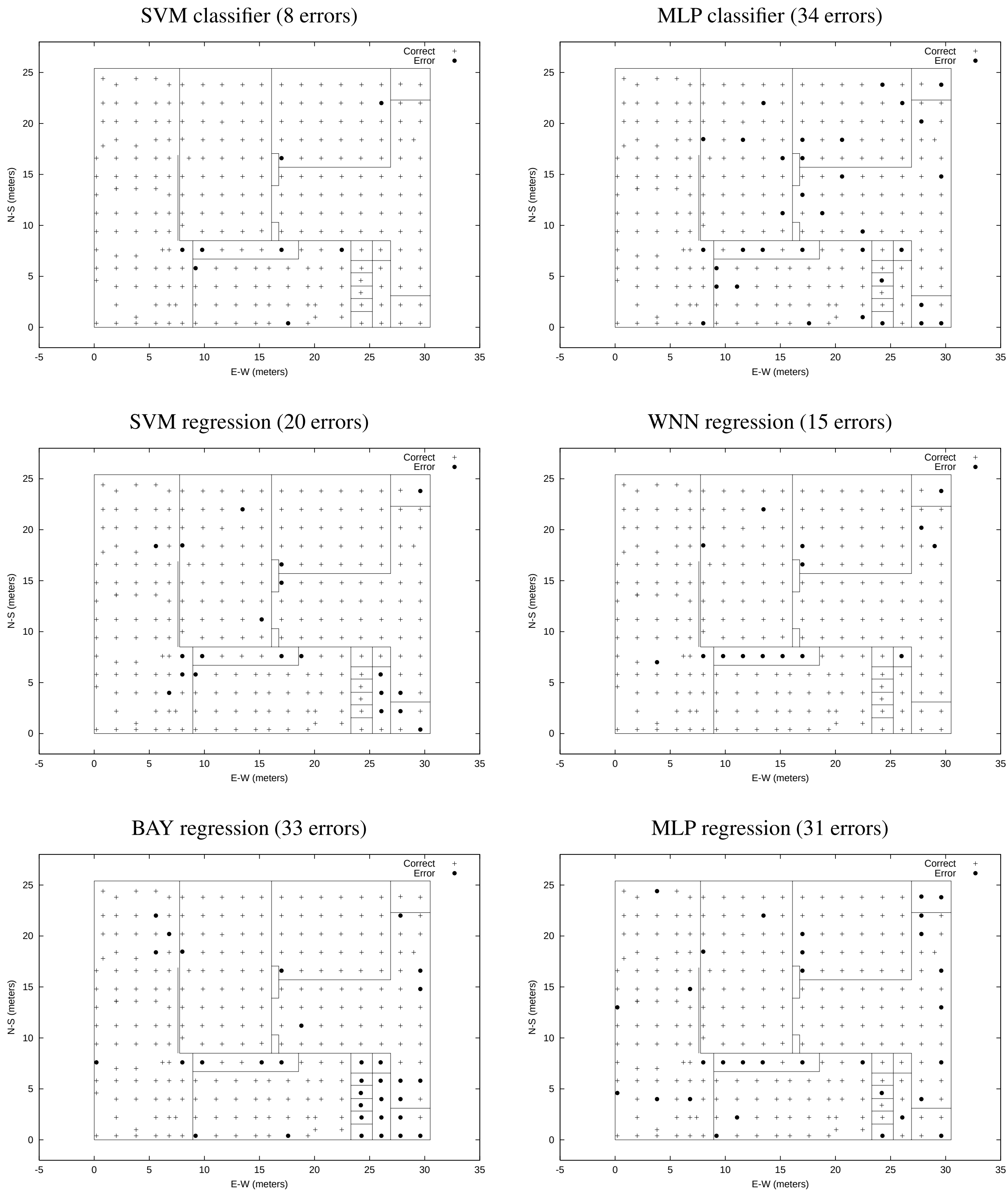

Fig. 5. Classification of samples according to room. In the top row, the classification outcome of SVM (left) and MLP (right) in native classification mode.

rithm and both test sets on the benchmarking machine (see Section VI-A.2). The last column shows the difference between the two testing runs, so that the initialization overhead is removed and the net estimation time for 5000 points is reported. As expected, the probabilistic model, requiring a global likelihood evaluation 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, CPU load and battery life.

VII. 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 the literature.

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. When used for regression (spatial localization), its results closely match those of the most 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.

Acknowledgments

The authors wish to thank Andrea Delai for his work on collecting the test samples and for developing software for the maximum likelihood approach in his thesis for the *laurea* degree [23].

References

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

[2] A. M. Ladd, K. E. Bekris, G. Marceau, A. Rudys, L. E. Kavraki, and 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, and J. Sievänen, "A probabilistic approach to WLAN user location estimation," *International Journal of Wireless Information Networks*, vol. 9, July 2002.

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

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

[6] R. Want, A. Hopper, V. Falcao, and J. Gibbons, "The active badge location system," *ACM Transaction on Information Systems*, vol. 10, pp. 91–102, Jan. 1992.

[7] A. Harter and A. Hopper, "A distributed location system for the active office," *IEEE Network*, vol. 6, pp. 62–70, Jan. 1994.

[8] A. Ward, A. Jones, and A. Hopper, "A new location technique for the active office," *IEEE Personal Communications*, vol. 4, pp. 42–47, Oct. 1997.

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

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

[11] J. Werb and C. Lanzl, "Designing a positioning system for finding things and people indoors," *IEEE Spectrum*, vol. 35, pp. 71–78, Sept. 1998.

[12] J. Hightower, G. Borriello, and 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] T. Roos, P. Myllymäki, and H. Tirri, "A statistical modeling approach to location estimation," *IEEE Transactions on Mobile Computing*, vol. 1, Jan. 2002.

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

[15] B. Boser, I. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *ACM Fifth Annual Workshop on Computational Learning Theory*, (Pittsburgh), pp. 144–152, 1992.

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

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

[18] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 121–167, 1998.

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

[20] R. Battiti, "First-and second-order methods for learning: Between steepest descent and newton's method," *Neural Computation*, vol. 4, pp. 141–166, 1992.

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

[22] S. Rüping, "mySVM — Manual," Tech. Rep. Lehrstuhl Informatik 8, University of Dortmund, 2000. http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/.

[23] A. Delai, "A statistical modeling approach to location estimation in wireless LANs." Laurea degree Thesis, supervisor prof. R. Battiti; Physics Department, University of Trento (Italy), Nov. 2002.