Extracting Places and Activities from GPS Traces Using Hierarchical Conditional Random Fields

Lin Liao Dieter Fox Henry Kautz
Department of Computer Science & Engineering
University of Washington
Seattle, WA 98195
Email:{liaolin,fox,kautz}@cs.washington.edu

Abstract

Learning patterns of human behavior from sensor data is extremely important for high-level activity inference. We show how to extract a person's activities and significant places from traces of GPS data. Our system uses hierarchically structured conditional random fields to generate a consistent model of a person's activities and places. In contrast to existing techniques, our approach takes high-level context into account in order to detect the significant places of a person. Our experiments show significant improvements over existing techniques. Furthermore, they indicate that our system is able to robustly estimate a person's activities using a model that is trained from data collected by other persons.

1 Introduction

The problem of learning patterns of human behavior from sensor data arises in many applications, including intelligent environments [6], surveillance [7, 9], human robot interaction [4], and assistive technology for the disabled [28]. A focus of recent interest is the use of data from wearable sensors, and in particular, GPS (global positioning system) location data, to learn to recognize the high-level activities in which a person is engaged over a period of many weeks, and to further determine the relationship between activities and places that are important to the user [3, 20, 23, 11, 21]. The goal of this research is to segment a user's day into everyday activities such as "working," "visiting," "travel," and to recognize and label significant places such as "workplace," "friend's house," "bus stop." However, previous approaches to automated activity and place labeling suffer from design decisions that limit their accuracy and flexibility:

Restricted activity models: Previous approaches to location-based activity recognition have rather limited models of a person's activities. Ashbrook and colleagues [3] only reason about moving between significant places, without considering different types of places or different routes between places. In the context of indoor mobile robotics, Bennewitz *et al.* [4] showed how to learn different motion paths between places. However, their approach does not model different types of places and does not estimate the user's activities when moving between places. In our previous work [20, 29] we developed a hierarchical dynamic Bayesian network model that can reason about different transportation routines between significant places. Gogate and colleagues [11] showed how to use deterministic constraints to more efficiently reason in such models. In separate work, we developed an approach that can learn to distinguish between different types of significant places, such as work place, home, or restaurant [23]. However, this model is limited in that it is not able to consider information about motion *between* places and about activities occurring at each point in time.

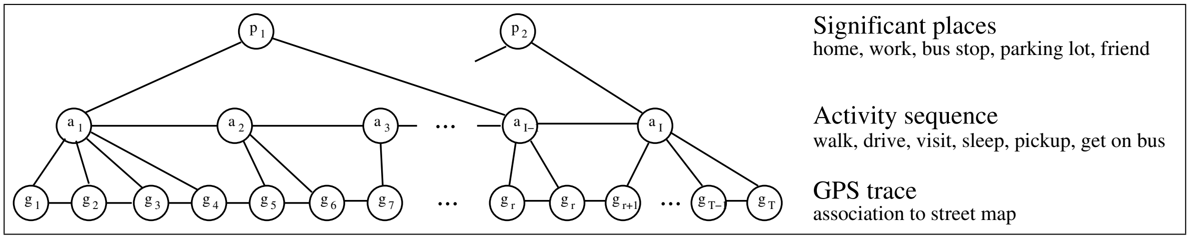
p1
p2
Significant places
home, work, bus stop, parking lot, friend
a1
a2
a3
...
aI-
aI
Activity sequence
walk, drive, visit, sleep, pickup, get on bus
g1
g2
g3
g4
g5
g6
g7
...
gr
gr
gr+1
...
gT-
gT
GPS trace
association to street map

Figure 1: The concept hierarchy for location-based activity recognition. For each day of data collection, the lowest level typically consists of several thousand GPS measurements, the next level contains around one thousand discrete activity cells, and the place level contains around five places.

Inaccurate place detection: Virtually all previous approaches address the problem of determining a person's significant places by assuming that a geographic location is significant if and only if the user spends at least θ minutes there, for some fixed threshold θ [3, 20, 23, 11, 4]. In practice, unfortunately, there is no threshold that leads to a satisfying detection of all significant places. For instance, locations such as the place where the user drops off his children at school may be visited only briefly, and so would be excluded when using a high threshold θ. A low threshold, on the other hand, would include too many insignificant locations, for example, a street intersection where the user waited at a traffic light. Such detection errors can only be resolved by taking additional context information into account, such as the user's current activity.

In this paper we present a novel, unified approach to automated activity and place labeling which overcomes these limitations. Key features of our system are:

- It achieves *high accuracy in detecting significant places* by taking a user's context into account when determining which places are significant. This is done by simultaneously estimating a person's activities over time, identifying places that correspond to significant activities, and labeling these places by their type. This estimation is performed in a unified, discriminatively trained graphical model (conditional random field). As a result, our approach does not rely on arbitrary thresholds regarding the time spent at a location or on a pre-specified number of significant places.

- It creates a *rich interpretation of a user's data*, including transportation activities as well as activities performed at particular significant places. It allows different kinds of activities to be performed at the same location, and vice-versa.

- This complex estimation task requires *efficient, approximate inference and learning algorithms*. Our system performs inference using loopy belief propagation, and parameter learning is done using pseudo-likelihood. In order to efficiently reason about aggregations, such as how many different places are labeled as a person's home, we apply Fast Fourier Transforms to compute aggregation messages within belief propagation.

This paper is organized as follows. We begin with an explanation of the basic concepts in our activity model. We then provide background on conditional random fields (CRFs) as well as the inference and parameter learning algorithms in CRFs. In Section 4, we show how to apply CRFs to the problem of location-based activity recognition. Finally, we present experimental results on real-world data that demonstrate significant improvement in coverage and accuracy over previous work.

2 Hierarchical Activity Model

The basic concept underlying our activity model is shown in Figure 1. Each circle in the model indicates an object such as a GPS reading, an activity, or a significant place. The edges illustrate probabilistic dependencies between these objects.

GPS readings are the input to our model — a typical trace consists of approximately one GPS reading per second; each reading is a point in 2D space. We segment a GPS trace in order to generate a discrete sequence of activity nodes at the next level of the model. This segmentation is done *spatially*, that is, each activity node represents a set of consecutive GPS readings that are within a certain area. If a street map is available, then we perform the segmentation by associating the GPS readings to a discretized version of the streets in the map (in our experiments we used 10m for discretization). This spatial segmentation is very compact and convenient for estimating high-level activities. For instance, our model represents a 12 hour stay at a location by a single node. Our model can also reason explicitly about the duration of a stay, for which dynamic models such as standard dynamic Bayesian networks or hidden Markov models have only limited support [12, 32].

Activities are estimated for each node in the spatially segmented GPS trace, as illustrated in Figure 1. In other words, our model labels a person's activity whenever she passes through or stays at a 10m patch of the environment. We distinguish two main groups of activities, *navigation activities* and *significant activities*. Activities related to navigation include walking, driving car, and riding bus. Significant activities are typically performed while a user stays at a location, such as work, leisure, sleep, visit, drop off / pickup, or when the user switches transportation modes, such as getting on/off a bus, or getting in/out of a car. To determine activities, our model relies heavily on temporal features, such as duration or time of day, and geographic information, such as locations of restaurants, stores, and bus stops.

Significant places are those locations that play a significant role in the activities of a person. Such places include a person's home and work place, the bus stops and parking lots the person typically uses, the homes of friends, stores the person frequently shops in, and so on. Note that our model allows different activities to occur at the same significant place. Furthermore, due to signal loss and noise in the GPS readings, the same significant place can comprise multiple, different locations.

Our activity model poses two key problems for probabilistic inference. First, the model can become rather complex, including thousands of probabilistic nodes with non-trivial probabilistic constraints between them. Second, a person's significant places depend on his activities and it is therefore not clear how to construct the model deterministically from a GPS trace. We solve the first problem by building on and extending approximate inference algorithms for conditional random fields. The second problem is solved by constructing the model as part of this inference. We do this by generating the highest level of the activity model (significant places) based on the outcome of inference in the lower level (activity sequence). Inference is then repeated using both levels connected appropriately (see Section 4.3). Before we explain our approach in detail, we will first give an overview of conditional random fields.

3 Preliminaries: Conditional Random Fields

In this section, we will provide the background of conditional random fields and the algorithms for inference and learning.

3.1 Overview

Our goal is to develop a probabilistic temporal model that can extract high-level activities from sequences of GPS readings. One possible approach is to use generative models such as hidden Markov models (HMM) [34, 7] or dynamic Bayesian networks [14, 25, 20]. However, discriminative models such as conditional random fields (CRF), have recently been shown to outperform generative techniques in areas such as natural language processing [18, 37], information extraction [31, 15], web page classification [38], and computer vision [17, 33]. We therefore decided to investigate the applicability of such models for activity recognition.

CRFs are undirected graphical models that were developed for labeling sequence data [18]. Instead of relying on Bayes rule to estimate the distribution over hidden states from observations, CRFs *directly* represent the conditional distribution over hidden states given the observations. Unlike HMMs, which assume that observations are independent given the hidden state, CRFs make no assumptions about the dependency structure between observations. CRFs are thus especially suitable for classification tasks with *complex* and *overlapped* attributes or observations.

Similar to HMMs and Markov random fields, the nodes in CRFs represent a sequence of observations (*e.g.*, GPS readings), denoted as $\mathbf{x} = \langle x_1, x_2, \ldots, x_T \rangle$, and corresponding hidden states (*e.g.*, activities), denoted as $\mathbf{y} = \langle y_1, y_2, \ldots, y_T \rangle$. These nodes, along with the connectivity structure imposed by undirected edges between them, define the conditional distribution $p(\mathbf{y}|\mathbf{x})$ over the hidden states $\mathbf{y}$. The fully connected sub-graphs of a CRF, called *cliques*, play a key role in the definition of the conditional distribution represented by a CRF. Let $\mathcal{C}$ be the set of all cliques in a given CRF. Then, a CRF factorizes the conditional distribution into a product of *clique potentials* $\phi_c(\mathbf{x}_c, \mathbf{y}_c)$, where every $c \in \mathcal{C}$ is a clique of the graph and $\mathbf{x}_c$ and $\mathbf{y}_c$ are the observed and hidden nodes in such a clique. Clique potentials are functions that map variable configurations to non-negative numbers. Intuitively, a potential captures the "compatibility" among the variables in the clique: the larger the potential value, the more likely the configuration. Using clique potentials, the conditional distribution over the hidden state is written as

$$p(\mathbf{y} \mid \mathbf{x}) \;=\; \frac{1}{Z(\mathbf{x})} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c, \mathbf{y}_c), \tag{1}$$

where $Z(\mathbf{x}) = \sum_{\mathbf{y}} \prod_{c \in \mathcal{C}} \phi_c(\mathbf{x}_c, \mathbf{y}_c)$ is the normalizing partition function. The computation of this partition function is exponential in the size of $\mathbf{y}$ since it requires summation over all possible configurations of hidden states $\mathbf{y}$. Hence, exact inference is possible for a limited class of CRF models only.

Without loss of generality, potentials $\phi_c(\mathbf{x}_c, \mathbf{y}_c)$ are described by log-linear combinations of *feature functions* $\mathbf{f}_c()$, *i.e.*,

$$\phi_c(\mathbf{x}_c, \mathbf{y}_c) \;=\; \exp\left\{\mathbf{w}_c^T \cdot \mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c)\right\}, \tag{2}$$

where $\mathbf{w}_c^T$ is the transpose of a weight vector $\mathbf{w}_c$, and $\mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c)$ is a function that extracts a vector of features from the variable values. The feature functions, which are often binary or real valued, are typically designed by the user (combinations of such functions can be learned from data [24]). As we will show in Section 3.2, the weights are learned from labeled training data. Intuitively, the weights represent the importance of different features for correctly identifying the hidden states. The log-linear feature representation (2) is very compact and guarantees the non-negativeness of potential values. We can now rewrite the conditional distribution (1) as

$$\begin{aligned} p(\mathbf{y} \mid \mathbf{x}) &= \frac{1}{Z(\mathbf{x})} \prod_{c \in \mathcal{C}} \exp\left\{\mathbf{w}_c^T \cdot \mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c)\right\} && (3) \\ &= \frac{1}{Z(\mathbf{x})} \exp\left\{\sum_{c \in \mathcal{C}} \mathbf{w}_c^T \cdot \mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c)\right\}. && (4) \end{aligned}$$

(4) follows by moving the products into the exponent.

3.2 Inference

In this section we will explain the inference techniques for CRFs. We will use $\mathbf{x}$ to denote observations and $\mathbf{y}$ to denote hidden states. Given a set of observations, inference in a CRF can have two tasks: to estimate the marginal distribution of each hidden variable, or to estimate the most likely configuration of the hidden variables (*i.e.*, the maximum a posteriori, or MAP, estimation). Both tasks can be solved under a framework called *belief propagation* (BP), which works by sending local messages through the graph structure of the model. The BP algorithm was originally proposed in the context of Bayesian networks [30], and was formulated equivalently in models such as factor graphs [16] and Markov networks (including CRFs) [40]. BP generates provably correct results if the graph has no loops, such as trees or polytrees [30]. If the graph contains loops, in which case BP is called loopy BP, then the algorithm is only approximate and might not converge to the correct probability distribution [26].

Without loss of generality, we only describe the BP algorithm for pairwise CRFs, which are CRFs that only contain cliques of size two. We will briefly discuss how to use BP in non-pairwise CRFs in the last paragraph of this section. Before running the inference algorithm in a pair-wise CRF, it is possible to remove all observed nodes $\mathbf{x}$ by merging their values into the corresponding potentials; that is, a potential $\phi(\mathbf{x}, \mathbf{y})$ can be written as $\phi(\mathbf{y})$ because $\mathbf{x}$ is fixed to one value. Therefore, the only potentials in a pair-wise CRF are local potentials, $\phi(y_i)$, and pair-wise potentials, $\phi(y_i, y_j)$. Corresponding to the two types of inference problems, there are two types of BP algorithms: *sum-product* for marginal estimation and *max-product* for MAP estimation.

3.2.1 Sum-product for marginal estimation

In the BP algorithm, we introduce a “message” $m_{ij}(y_j)$ for each pair of neighbors y_i and y_j, which is a distribution (not necessarily normalized) sent from node i to its neighbor j about which state variable y_j should be in. The messages propagate through the CRF graph until they (possibly) converge, and the marginal distributions can be estimated from the stable messages. A complete BP algorithm defines how to initialize messages, how to update messages, how to schedule the message updates, and when to stop passing messages.

- **Message initialization:** All messages $m_{ij}(y_j)$ are initialized as uniform distributions over y_j.

- **Message update rule:** The message $m_{ij}(y_j)$ sent from node i to its neighbor j is updated based on local potentials $\phi(y_i)$, the pair-wise potential $\phi(y_i, y_j)$, and all the messages to i received from i’s neighbors other than j (denoted as $n(i) \setminus j$). More specifically, for sum-product, we have

$$m_{ij}(y_j) \quad = \quad \sum_{y_i} \phi(y_i)\phi(y_i, y_j) \prod_{k \in n(i) \setminus j} m_{ki}(y_i) \tag{5}$$

- **Message update order:** The algorithm iterates the message update rule until it (possibly) converges. Usually, at each iteration, it updates each message once, and the specific order is not important (although it might affect the convergence speed).

- **Convergence conditions:** To test whether the algorithm converged, BP measures the difference between the previous messages and the updated ones. The convergence condition is met when all the differences are below a given threshold ϵ. More formally, the condition is

$$||m_{ij}(y_j)^{(k)} - m_{ij}(y_j)^{(k-1)}|| < \epsilon, \forall i, \text{ and } \forall j \in n(i) \tag{6}$$

where $m_{ij}(y_j)^{(k)}$ and $m_{ij}(y_j)^{(k-1)}$ are the messages after and before iteration k, respectively.

In the sum-product algorithm, after all messages are converged, it is easy to calculate the marginals of each node and each pair of neighboring nodes as

$$\begin{aligned} b(y_i) &\propto \phi(y_i) \prod_{j \in n(i)} m_{ji}(y_i) && (7) \\ b(y_i, y_j) &\propto \phi(y_i)\phi(y_j)\phi(y_i, y_j) \prod_{k \in n(i) \setminus j} m_{ki}(y_i) \prod_{l \in n(j) \setminus i} m_{lj}(y_j) && (8) \end{aligned}$$

The above algorithm can be applied to any topology of pair-wise CRFs. When the network structure does not have a loop (for example, when it is a tree), the obtained marginals are guaranteed to be exact. When the structure has loops, the BP algorithm usually cannot obtain exact marginals, or it may even not converge. Fortunately, empirical experiments show that loopy belief propagation often converges to a good approximation of the correct posterior.

In practice, when the network structure does not have any loops, we do not have to initialize all the messages as uniforms. Instead, we start BP with the nodes at the edge of the graph (*i.e.*, nodes with only one neighbor), and compute a message $m_{ij}(y_j)$ only when all the messages on the right side of (5) are available. By doing this, the algorithm converges by only computing each message once, and the results are guaranteed to be exact.

3.2.2 Max-product for MAP estimation

We denote the messages sent in the max-product algorithm as $m_{ij}^{max}(y_j)$. The whole algorithm of max-product is very similar to the sum-product, except that in the message update rule summation is replaced by maximization. The new rule becomes

$$m_{ij}^{max}(y_j) = \max_{y_i} \phi(y_i)\phi(y_i, y_j) \prod_{k \in n(i) \setminus j} m_{ki}^{max}(y_i) \qquad (9)$$

We run the max-product algorithm in the same way as sum-product. After the algorithm converges, we calculate the MAP belief at each node y_i as

$$b(y_i) \propto \phi(y_i) \prod_{j \in n(i)} m_{ji}^{max}(y_i) \qquad (10)$$

Suppose there is a unique MAP configuration $\mathbf{y}^*$. Then each component of $\mathbf{y}^*$ is simply the most likely value according to the MAP belief:

$$\mathbf{y}_i^* = \operatorname*{argmax}_{y_i} b(y_i) \qquad (11)$$

So far, we explained the two BP algorithms in the context of pairwise CRFs. For non-pairwise CRFs, there is a standard way to convert them to pairwise ones [40]. Intuitively, this conversion generates a new node for each clique of size greater than two. The state space of the new node consists of the joint state of the nodes it was generated from. Thus, the complexity of belief propagation is exponential in the number of nodes in the largest clique of the CRF.

3.3 Parameter Learning

The goal of parameter learning is to determine the weights of the feature functions used in the conditional likelihood (4). CRFs learn these weights *discriminatively*, that is, the weights are determined so as to maximize the conditional likelihood $p(\mathbf{y}|\mathbf{x})$ of labeled training data. This is in contrast to generative learning, which aims to learn a model of the joint probability $p(\mathbf{y}, \mathbf{x})$. Ng and Jordan [27] present a discussion and comparison of these two learning regimes, concluding that discriminative learning asymptotically reaches superior performance but might require more training examples until its performance converges. The reader may notice that discriminative learning can also be performed for generative models such as hidden Markov models [8], Kalman filters [1], or Markov random fields [2].

3.3.1 Maximum Likelihood (ML) Estimation

As can be seen in (4), given labeled training data $(\mathbf{x}, \mathbf{y})$, the conditional likelihood $p(\mathbf{y}|\mathbf{x})$ only depends on the feature weights $\mathbf{w}_c$. In the derivation of the learning algorithm it will be convenient to re-write (4) as

$$\begin{aligned} p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) &= \frac{1}{Z(\mathbf{x})} \exp\left\{\sum_{c \in \mathcal{C}} \mathbf{w}_c^T \cdot \mathbf{f}_c(\mathbf{x}_c, \mathbf{y}_c)\right\} && (12)\\ &= \frac{1}{Z(\mathbf{x}, \mathbf{w})} \exp\left\{\mathbf{w}^T \cdot \mathbf{f}(\mathbf{x}, \mathbf{y})\right\}, && (13)\end{aligned}$$

where $\mathbf{w}$ and $\mathbf{f}$ are the vectors resulting from "stacking" the weights and the feature functions for all cliques in the CRF, respectively. In order to make the dependency on $\mathbf{w}$ more explicit, we will write the conditional likelihood from now on as $p(\mathbf{y}|\mathbf{x}, \mathbf{w})$. A common parameter estimation method is to search for the $\mathbf{w}$ that maximizes this likelihood, or equivalently, that minimizes the *negative log-likelihood*, $-\log p(\mathbf{y}|\mathbf{x}, \mathbf{w})$ [18, 38, 23]. To avoid overfitting, one typically imposes a so-called shrinkage prior on the weights to keep them from getting too large. More specifically, we define the objective function to minimize as follows:

$$\begin{aligned} L(\mathbf{w}) &\equiv -\log p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) + \frac{\mathbf{w}^T\mathbf{w}}{2\sigma^2} && (14)\\ &= -\mathbf{w}^T \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}) + \log Z(\mathbf{x}, \mathbf{w}) + \frac{\mathbf{w}^T\mathbf{w}}{2\sigma^2} && (15)\end{aligned}$$

The rightmost term in (14) serves as a zero-mean, Gaussian prior with variance σ^2 on each component of the weight vector. (15) follows directly from (14) and (13). While there is no closed-form solution for maximizing (15), it can be shown that (15) is convex relative to $\mathbf{w}$. Thus, L has a global optimum which can be found using numerical gradient algorithms. The gradient of the objective function $L(\mathbf{w})$ is given by

$$\begin{aligned} \nabla L(\mathbf{w}) &= -\mathbf{f}(\mathbf{x}, \mathbf{y}) + \frac{\nabla Z(\mathbf{x}, \mathbf{w})}{Z(\mathbf{x}, \mathbf{w})} + \frac{\mathbf{w}}{\sigma^2} && (16)\\ &= -\mathbf{f}(\mathbf{x}, \mathbf{y}) + \frac{\sum_{\mathbf{y}'} \exp\{\mathbf{w}^T \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}')\}\mathbf{f}(\mathbf{x}, \mathbf{y}')}{Z(\mathbf{x}, \mathbf{w})} + \frac{\mathbf{w}}{\sigma^2} && (17)\\ &= -\mathbf{f}(\mathbf{x}, \mathbf{y}) + \sum_{\mathbf{y}'} P(\mathbf{y}' \mid \mathbf{x}, \mathbf{w})\mathbf{f}(\mathbf{x}, \mathbf{y}') + \frac{\mathbf{w}}{\sigma^2} && (18)\\ &= -\mathbf{f}(\mathbf{x}, \mathbf{y}) + E_{P(\mathbf{y}'|\mathbf{x},\mathbf{w})}[\mathbf{f}(\mathbf{x}, \mathbf{y}')] + \frac{\mathbf{w}}{\sigma^2} && (19)\end{aligned}$$

where (17) follows from the definition of the partition function, $Z(\mathbf{x}, \mathbf{w}) = \sum_{\mathbf{y}'} \exp\{\mathbf{w}^T \cdot \mathbf{f}(\mathbf{x}, \mathbf{y}')\}$, (18) follows from the definition of the conditional likelihood by putting $Z(\mathbf{x}, \mathbf{w})$ inside the summation, and in (19), the second term is expressed as an expectation over the distribution $P(\mathbf{y}' \mid \mathbf{x}, \mathbf{w})$. Therefore, the gradient is just the difference between the *empirical feature values* $\mathbf{f}(\mathbf{x}, \mathbf{y})$ and the *expected feature values* $E_{P(\mathbf{y}'|\mathbf{x},\mathbf{w})}[\mathbf{f}(\mathbf{x}, \mathbf{y}')]$, plus a prior term. To compute the expectation over the feature values it is necessary to run inference in the CRF using the current weights $\mathbf{w}$. This can be done via belief propagation as discussed in the previous section. Sha and Pereira [37] showed that straightforward gradient descent often converges slowly during learning, but that modern numerical optimization algorithms, such as conjugate gradient or quasi-Newton techniques (as used in our experiments), can be much faster.

3.3.2 Maximum Pseudo-Likelihood (MPL) Estimation

Maximizing the likelihood requires running an inference procedure at each iteration of the optimization, which can be very expensive. An alternative is to maximize the *pseudo-likelihood* of the training data [5], which is the following approximation to the conditional likelihood:

$$p(\mathbf{y} \mid \mathbf{x}, \mathbf{w}) \quad \approx \quad \prod_{i=1}^{n} p(\mathbf{y}_i \mid \mathrm{MB}(\mathbf{y}_i), \mathbf{w}) \tag{20}$$

Here, $\mathrm{MB}(\mathbf{y}_i)$ is the Markov blanket of variable $\mathbf{y}_i$, which contains the immediate neighbors of $\mathbf{y}_i$ in the CRF graph (note that the value of each node is known during learning). Thus, the pseudo-likelihood is the product of all the *local likelihoods*, $p(\mathbf{y}_i \mid \mathrm{MB}(\mathbf{y}_i)$. The pseudo-likelihood can be re-written as

$$\prod_{i=1}^{n} p(\mathbf{y}_i \mid \mathrm{MB}(\mathbf{y}_i), \mathbf{w}) \quad = \quad \prod_{i=1}^{n} \frac{1}{Z(\mathrm{MB}(\mathbf{y}_i), \mathbf{w})} \exp\{\mathbf{w}^T \cdot \mathbf{f}(\mathbf{y}_i, \mathrm{MB}(\mathbf{y}_i))\}, \tag{21}$$

where $\mathbf{f}(y_i, \mathrm{MB}(\mathbf{y}_i))$ is the local feature counts involving variable $\mathbf{y}_i$, and $Z(\mathrm{MB}(\mathbf{y}_i), \mathbf{w}) = \sum_{\mathbf{y}'_i} \exp\{\mathbf{w}^T \cdot \mathbf{f}(\mathbf{y}'_i, \mathrm{MB}(\mathbf{y}'_i))\}$ is the *local* normalizing function. Therefore, computing pseudo-likelihood is much more efficient than computing likelihood $p(\mathbf{y}|\mathbf{x}, \mathbf{w})$, because pseudo-likelihood only requires computing local normalizing functions and avoids computing the global partition function $Z(\mathbf{x}, \mathbf{w})$.

As with ML, in practice we minimize the negative log-pseudo-likelihood and a shrinkage prior, and the objective function becomes

$$PL(\mathbf{w}) \quad \equiv \quad -\sum_{i=1}^{n} \log p(\mathbf{y}_i \mid \mathrm{MB}(\mathbf{y}_i), \mathbf{w}) + \frac{\mathbf{w}^T\mathbf{w}}{2\sigma^2} \tag{22}$$

$$\quad = \quad \sum_{i=1}^{n} \big(-\mathbf{w}^T \cdot \mathbf{f}(\mathbf{y}_i, \mathrm{MB}(\mathbf{y}_i)) + \log Z(\mathrm{MB}(\mathbf{y}_i), \mathbf{w})\big) + \frac{\mathbf{w}^T\mathbf{w}}{2\sigma^2} \tag{23}$$

Again, $PL(\mathbf{w})$ is a convex function and it is possible to use gradient-based algorithms to find the $\mathbf{w}$ that minimizes $PL(\mathbf{w})$. The gradient can be computed as

$$\nabla PL(\mathbf{w}) \quad = \quad \sum_{i=1}^{n} \Big(-\mathbf{f}(\mathbf{y}_i, \mathrm{MB}(\mathbf{y}_i)) + E_{P(\mathbf{y}'_i|\mathrm{MB}(\mathbf{y}_i), \mathbf{w})}[\mathbf{f}(\mathbf{y}'_i, \mathrm{MB}(\mathbf{y}_i))]\Big) + \frac{\mathbf{w}}{\sigma^2}. \tag{24}$$

As we can see, (24) can be expressed as the difference between empirical feature values and expected feature values, similar to (19). However, the key difference is that (24) can be evaluated very efficiently without running a complete inference procedure. Learning by maximizing pseudo-likelihood has been shown to perform very well in several domains [17, 36]. In our experiments we found that this type of learning is extremely efficient and consistently achieves good results.

3.3.3 Parameter Sharing

The definition of the weight vector and its gradient described above does not support *parameter sharing*, which requires the learning algorithm to learn the same parameter values (weights) for different cliques

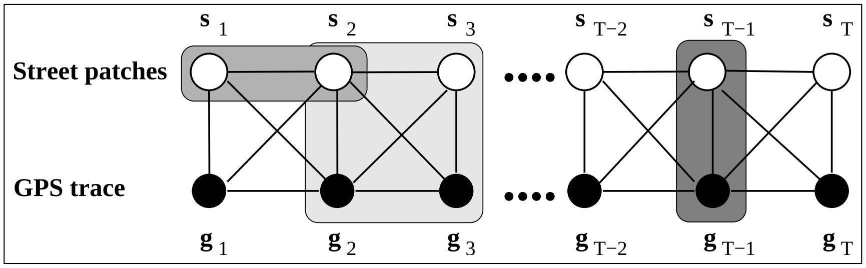

Figure 2: CRF for associating GPS measurements to street patches. The shaded areas indicate different types of cliques.

in the CRF. For instance, as we will describe, the CRF in Section 4.1 only contains three different weights, one for each *type* of feature (see Eq.(26)). Thus, the same weight $\mathbf{w}_{\mathrm{m}}$ is used for each clique containing a street patch node s_t and a GPS reading node g_t. Without parameter sharing, a CRF would learn for each location / GPS combination the weights that are best in classifying that specific location. However, we want to learn a *generic model* that can take any GPS trace and classify the locations in that trace, even if the trace is collected by a different person visiting very different locations. Parameter sharing enforces learning such kind of models since the parameters are independent of the individual locations. Parameter sharing can be achieved by making sure that all the weights belonging to a certain type of feature are identical. As it turns out, the gradients with respect to these shared weights are almost identical to the gradients (19) and (24). The only difference lies in the fact that the gradient for a shared weight is given by the sum of all the gradients computed for the individual cliques in which this weight occurs [38, 23].

Parameter sharing can be modeled conveniently using probabilistic relational models such as relational Markov networks [38, 23]. These techniques allow the automatic specification and construction of CRF models using so-called clique templates, which enable the specification of parameter sharing for inference and learning.

4 Conditional Random Fields for Activity Recognition

In this section, we will describe the application of CRFs to activity recognition from location data. Specifically, we will explain how we construct from raw GPS traces the CRFs that encodes the hierarchical activity model, and how we simultaneously label all the types of activities and significant places.

4.1 GPS to street map association

As mentioned above, we segment GPS traces by grouping consecutive GPS readings based on their spatial relationship. Without a street map, this segmentation can be performed by simply combining all consecutive readings that are within a certain distance from each other (10m in our implementation). However, it might be desirable to associate GPS traces to a street map, for example, in order to relate locations to addresses in the map. Street maps are represented by directed graphs, where a vertex typically is an intersection between streets, and an edge usually represents a city block section of a street and is associated with a direction either up or down the street [20].

To jointly estimate the GPS to street association and the trace segmentation, we associate each GPS measurement to a 10m patch on a street edge [1]. As shown in Fig. 5(a) in Section 5, GPS traces can deviate significantly from the street map, mostly because of measurement errors and inaccuracies in

[1] In [20], we showed how to perform such an association using Rao-Blackwellised particle filters with multiple Kalman filters moving through the street graph. Since the focus of this work is on high level activities and places rather than accurate tracking, we use this more straightforward and efficient approach to trace segmentation.

street maps. One straightforward way to perform this association is to snap each GPS reading to the nearest street patch. However, such an approach would clearly give wrong results in situations such as the one shown in Fig. 5(a). To generate a consistent association, we construct a CRF that takes into account the spatial relationship between GPS readings. The structure of this CRF is shown in Figure 2. The observed, solid nodes correspond to GPS readings g_t where the subscript t indicates the index of time, and the white nodes represent the street patches s_t, which correspond to the hidden state $\mathbf{y}$ in Section 3. The values of each s_t range over the street patches in the map that are within a certain distance of the GPS reading g_t. The lines in Figure 2 define the clique structure of the CRF. We distinguish three types of cliques, for which potentials are defined via the following feature functions:

- **Measurement cliques** (dark grey in Figure 2): GPS noise and map uncertainty are considered by cliques whose features measure the squared distance between a GPS measurement and the center of the patch it is associated with:

  $$\mathbf{f}_{\text{meas}}(g_t, s_t) \quad = \quad \frac{||g_t - s_t||^2}{\sigma^2}$$

  where g_t is the location of the t-th GPS reading. With slight abuse of notation, we denote by s_t the center of one of the street patches in the vicinity of g_t. σ is used to control the scale of the distance (note that this feature function corresponds to a Gaussian noise model for GPS measurements). Obviously, when combined with a negative weight, this feature prefers associations in which GPS readings are snapped to nearby patches. The feature $\mathbf{f}_{\text{meas}}$ is used for the potential of all cliques connecting GPS readings and their street patches.

- **Consistency cliques** (light grey in Figure 2): Temporal consistency is ensured by four node cliques that compare the spatial relationship between consecutive GPS readings and the spatial relationship between their associated patches. The more similar these relationships, the more consistent the association. This comparison is done via a feature function that compares the vectors between GPS readings and associated patches:

  $$\mathbf{f}_{\text{cons}}(g_t, g_{t+1}, s_t, s_{t+1}) \quad = \quad \frac{||(g_{t+1} - g_t) - (s_{t+1} - s_t)||^2}{\sigma^2}$$

  Here, s_t and s_{t+1} are the centers of street patches associated at two consecutive times.

- **Smoothness cliques** (medium grey in Figure 2): These cliques prefer traces that do not switch frequently between different streets. For instance, it is very unlikely that a person drives down a street and switches for two seconds to another street at an intersection. To model this information, we use binary features that test whether consecutive patches are on the same street, on neighboring streets, or in the same direction. For example, the following binary feature examines if both street blocks and directions are identical:

  $$\mathbf{f}_{\text{smooth}}(s_t, s_{t+1}) \quad = \quad \delta(s_t.\text{street}, s_{t+1}.\text{street}) \cdot \delta(s_t.\text{direction}, s_{t+1}.\text{direction}) \qquad (25)$$

  where $\delta(u, v)$ is the indicator function which equals 1 if $u = v$ and 0 otherwise.

During inference, the CRF uses these features to estimate distributions over the association between the GPS trace and street patches. Using the feature functions defined above, this conditional distribution can be written as

$$p(\mathbf{s}|\mathbf{g}) = \frac{1}{Z(\mathbf{x})} \exp \left\{ \sum_{t=1}^{T} \mathbf{w}_{\text{m}} \cdot \mathbf{f}_{\text{meas}}(g_t, s_t) + \sum_{t=1}^{T-1} \left(\mathbf{w}_{\text{c}} \cdot \mathbf{f}_{\text{cons}}(g_t, g_{t+1}, s_t, s_{t+1}) + \mathbf{w}_{\text{s}} \cdot \mathbf{f}_{\text{smooth}}(s_t, s_{t+1}) \right) \right\}, \quad (26)$$

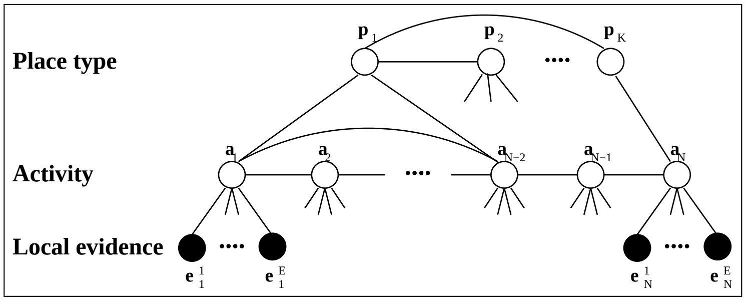

Figure 3: CRF for labeling activities and places. Activity nodes a_i range over activities, and place nodes p_i range over types of places. Each activity node is connected to E observed local evidence nodes e_i^1 to e_i^E. Local evidence comprises information such as time of day, duration, and motion velocity. Place nodes are generated based on the activities inferred at the activity level. Each place is connected to all activity nodes that are within a certain range.

where $\mathbf{w}_\mathrm{m}, \mathbf{w}_\mathrm{c}$ and $\mathbf{w}_\mathrm{s}$ are the corresponding feature function weights. The reader may notice that the weights and feature functions are independent of the time index, and this independence is achieved using parameter sharing discussed in Section 3.3.3. Figure 5(a) illustrates the maximum a posteriori association of a GPS trace to a map. Intuitively, this sequence corresponds to the MAP sequence that results from tracking a person's location on the discretized street map. Such an association also provides a unique segmentation of the GPS trace. This is done by combining consecutive GPS readings that are associated to the same street patch.

4.2 Inferring activities and types of significant places

Once a GPS trace is segmented, our system estimates the activity performed at each segment and a person's significant places. To do so, it generates a new CRF that contains a hidden activity node for every segment extracted from the GPS trace. This CRF consists of the two lower levels of the one shown in Figure 3. Each activity node is connected to various features, summarizing information resulting from the GPS segmentation. These features include:

- Temporal information such as time of day, day of week, and duration of the stay. These measures are discretized in order to allow more flexible feature functions. For example, time of day can be *Morning*, *Noon*, *Afternoon*, *Evening*, or *Night*. The feature functions for the cliques connecting each activity node to one of the solid nodes in the CRF shown in Figure 3 are binary indicator functions, one for each possible combination of temporal feature and activity. For instance, the following function returns 1 if the activity is work and the time of day is morning, and 0 otherwise: $\mathbf{f}(a_i, d_i) = \delta(a_i, Work) \cdot \delta(d_i, Morning)$.

- Average speed through a segment, which is important for discriminating different transportation modes. The speed value is also discretized and indicator features are used, just as with temporal information. This discretization has the advantage over a linear feature function that it is straightforward to model multi-modal velocity distributions.

- Information extracted from geographic databases, such as whether a patch is on a bus route, whether it is close to a bus stop, and whether it is near a restaurant or grocery store. Again, we use indicator features to incorporate this information.

- Additionally, each activity node is connected to its neighbors. These features measure compatibility between types of activities at neighboring nodes in the trace. For instance, it is extremely unlikely that a person will get on the bus at one location and drive a car at the neighboring location right afterwards. The corresponding feature function is $\mathbf{f}(a_i, a_{i+1}) = \delta(a_i, OnBus) \cdot$

$\delta(a_{i+1}, Car)$, where a_i and a_{i+1} are specific activities at two consecutive activity nodes. The weight of this feature should be a negative value after supervised learning, thereby giving a labeling that contains this combination a lower probability.

Our model also aims to determine those places that play a significant role in the activities of a person, such as home, workplace, friends' home, grocery stores, restaurants, and bus stops. The nodes representing such *significant places* comprise the upper level of the CRF shown in Figure 3. However, since these places are not known *a priori*, we must additionally detect a person's significant places. To incorporate place detection into our system, we use an iterative algorithm that re-estimates activities and places. Before we describe this algorithm, let us first look at the features that are used to determine the types of significant places under the assumption that the number and locations of these places are known. In order to infer place types, we use the following features for the cliques connected to the place nodes p_i in the CRF:

- The activities that occur at a place strongly indicate the type of the place. For example, at grocery stores people mainly do shopping, and at a friends' home people either visit or pick up / drop off someone. Our features consider the weekly *frequency* of the different activities occurring at a place. This is done by generating a clique for each place that contains all activity nodes in its vicinity. For example, the nodes p_1, a_1, and a_{N-2} in Figure 3 form such a clique. The model then counts the different activities occurring at each place. In our experiments, we discretize the counts into four categories: count $= 0$, count $= 1$, $2 \leq$ count ≤ 3, and count ≥ 4. Then for each combination of type of place, type of activity, and frequency category, we have an indicator feature.

- A person usually has only a limited number of different homes or work places. To use this knowledge to improve labeling places, we add two additional summation cliques that count the number of different homes and work places. These counts provide soft constraints that bias the system to generate interpretations that result in reasonable numbers of different homes and work places. The features are simply the counts, which make the likelihood of labellings decrease exponentially as the counts increase.

Note that the above two types of features can generate very large cliques in the CRF. This is because we must build a clique for all the activities at a place to count the frequencies of activities, and connect all the place nodes to count the number of homes or workplaces. Standard belief propagation would be intractable for such cliques. Fortunately, it is possible to convert cliques generated for summation features to tree-structured CRFs. In such structures, BP inference can be done in polynomial time, and for sum-product it is even possible to apply the Fast Fourier Transform (FFT) to further speed up message passing (see [22] for details).

4.3 Place Detection and Labelling Algorithm

The CRF discussed so far assumes that the number and locations of a person's significant places are known in advance. Since these places are not known, however, it is necessary to infer the structure of the hierarchical CRF. Table 1 summarizes our algorithm for efficiently constructing this CRF. The algorithm takes as input a GPS trace. In Step 3, this trace is segmented into activity nodes a_i. Each such node is characterized by local evidence e_i^j, which is extracted from the GPS readings associated to it. As discussed above, segmentation of a trace is performed by either clustering consecutive GPS readings that are nearby or associating the GPS trace to a discretized street map using the CRF shown in Figure 2. The activity nodes and their evidence are then used in Step 4 to generate a CRF such as the one shown in Figure 3. However, since significant places are not yet known at this stage, CRF$_0$ contains no place nodes. Maximum a posteriori inference is then performed in this restricted CRF so as

```
1.  **Input:** GPS trace $\langle g_1, g_2, \ldots, g_T \rangle$
2.  $i := 0$
3.  // Generate activity segments and local evidence by grouping consecutive GPS readings
    $\big(\langle a_1, \ldots, a_N \rangle, \langle e_1^1, \ldots, e_1^E, \ldots, e_N^1, \ldots, e_N^E \rangle\big)$ := spatial_segmentation($\langle g_1, g_2, \ldots, g_T \rangle$)
4.  // Generate CRF containing activity and local evidence nodes (lower two levels in Figure 3)
    $\mathrm{CRF}_0$ := instantiate_crf$\big(\langle\ \rangle, \langle a_1, \ldots, a_N \rangle, \langle e_1^1, \ldots, e_1^E, \ldots, e_N^1, \ldots, e_N^E \rangle\big)$
5.  // Determine MAP sequence of activities
    $\mathbf{a_0^*}$ := MAP_inference( $\mathrm{CRF}_0$)
6.  **do**
7.      i := i + 1
8.      // Generate places by clustering significant activities
        $\langle p_1, \ldots, p_K \rangle_i$ := generate_places($\mathbf{a_{i-1}^*}$)
9.      // Generate complete CRF with instantiated places
        $\mathrm{CRF}_i$ := instantiate_crf$\big(\langle p_1, \ldots, p_K \rangle_i, \langle a_1, \ldots, a_N \rangle, \langle e_1^1, \ldots, e_1^E, \ldots, e_N^1, \ldots, e_N^E \rangle\big)$
10.     // Perform MAP inference in complete CRF
        $\langle \mathbf{a_i^*}, \mathbf{p_i^*} \rangle$ := MAP_inference( $\mathrm{CRF}_i$)
11. **until** $\mathbf{a_i^*} = \mathbf{a_{i-1}^*}$
12. **return** $\langle \mathbf{a_i^*}, \mathbf{p_i^*} \rangle$
```

Table 1: Algorithm for jointly detecting significant places and inferring activities and types of places.

to determine the MAP activity sequence $\mathbf{a_0^*}$, which consists of a sequence of locations and the activity performed at that location (Step 5). Within each iteration of the loop starting at Step 6, such an activity sequence is used to extract a set of significant places. This is done by classifying individual activities in the sequence according to whether or not they belong to a significant place. For instance, while walking, driving a car, or riding a bus is not associated with significant places, working or getting on or off the bus indicates a significant place. All instances at which a *significant activity* occurs generate a place node. Because a place can be visited multiple times within a sequence, we perform spatial clustering and merge duplicate places into the same place node. This classification and clustering is performed by the algorithm generate_places(), which returns a set of K place nodes p_j in Step 8. These places, along with the activity nodes a_i and their local evidence e_i^j are used to generate a complete CRF. Step 10 performs MAP estimation in this new CRF. Since this CRF has a different structure than the initial CRF_0, it might generate a different MAP activity sequence. If this is the case, then the algorithm returns to Step 6 and re-generates the set of places using this improved activity sequence. This process is repeated until the activity sequence does not change, which is tested in Step 11. Finally, the algorithm returns the MAP activity sequence along with the set of places and their MAP types. In our experiments we observed that this algorithm converges very quickly, typically after three or four iterations.

Our experiments show that this algorithm is extremely efficient and robust. We are currently investigating an alternative approach that performs a search through the space of CRF structures by replacing the MAP sequences generated in Steps 5 and 10 by multiple sequences generated from the posterior. This can be done via MCMC sampling [10] or k-best algorithms for graphical models [39]. Each sample would correspond to a CRF structure, which can then be evaluated according to the data likelihood.

5 Experimental Results

In our experiments we evaluate how well our system can *extract and label a person's activities and significant places*. Furthermore, we demonstrate that it is feasible to learn models from data collected by a set of people and to apply this model to another person. That is, our system can recognize a person's activities without requiring any manual labeling of that person's data.

Specifically, we collected GPS data traces from four different persons, approximately seven days of data per person. The data from each person consisted of roughly 40,000 GPS measurements, resulting in about 10,000 segments. We then manually labeled all activities and significant places in these traces [2]. We used leave-one-out cross-validation for evaluation, that is, learning was performed based on the data collected by three persons and the learned model was evaluated on the fourth person. We used maximum pseudo-likelihood for learning, which took (on a 1.5 GHz PC) about one minute to converge on the training data. Pseudo-likelihood converged in all our experiments. We did not use maximum likelihood for learning since it did not always converge (even after several hours). This is most likely due to the fact that the approximation of the loopy BP inference is not good enough to provide accurate gradients for learning. However, we successfully used loopy BP as inference approach in all our evaluation runs. For each evaluation, we used the algorithm described in Table 1, which typically extracted the MAP activities and places from one week's trace within one minute of computation. When a street map was used, the association between GPS trace and street map performed in Step 3 of the algorithm took additional four minutes (see also Section 4.1).

Example analysis

The different steps involved in the analysis of a GPS trace are illustrated in Figure 4. The second panel (b) shows the GPS trace snapped to 10m patches on the street map. This association is performed by Step 3 of the algorithm given in Table 1, using the CRF discussed in Section 4.1. The visited patches, along with local information such as time of day or duration, are used to generate the activity CRF. This is done by Step 4 in Table 1, generating the activity level of Figure 3. MAP inference in this CRF determines one activity for each patch visit, as shown in panel (c) of Figure 4 (Step 5 of the algorithm). Note that this example analysis misses the get-off-bus activity at the left end of the bus trip. The significant activities in the MAP sequence are clustered and generate additional place nodes in a new CRF (Steps 8 and 9 in Table 1). MAP inference in this CRF provides labels for the detected places, as shown in Figure 4(d). The algorithm repeats generation of the CRFs until the MAP activity sequence does not change any more. In all experiments, this happens within the first four iterations of the algorithm.

Figure 5(a) provides another example of the quality achieved by our approach to snapping GPS traces to street maps. Note how the complete trace is snapped consistently to the street map. Table 2 shows a typical summary of a person's day provided by the MAP sequence of activities and visited places. Note that the system determines where the significant places are, how the person moves between them, and what role the different places play for this person.

Extracting significant places

In this experiment we compare our system's ability to detect significant places to the results achieved with a widely-used approach that applies a time threshold to determine whether or not a location is significant [3, 13, 20, 23, 11]. Our approach was trained on data collected by three people and tested on the fourth person. For the threshold method, we generated results for different thresholds from 1

[2] Even though we performed the manual labeling as thoroughly as possible, we might have "missed" some significant places and activities. This might result in a slightly lower false negative rate in our experiments.

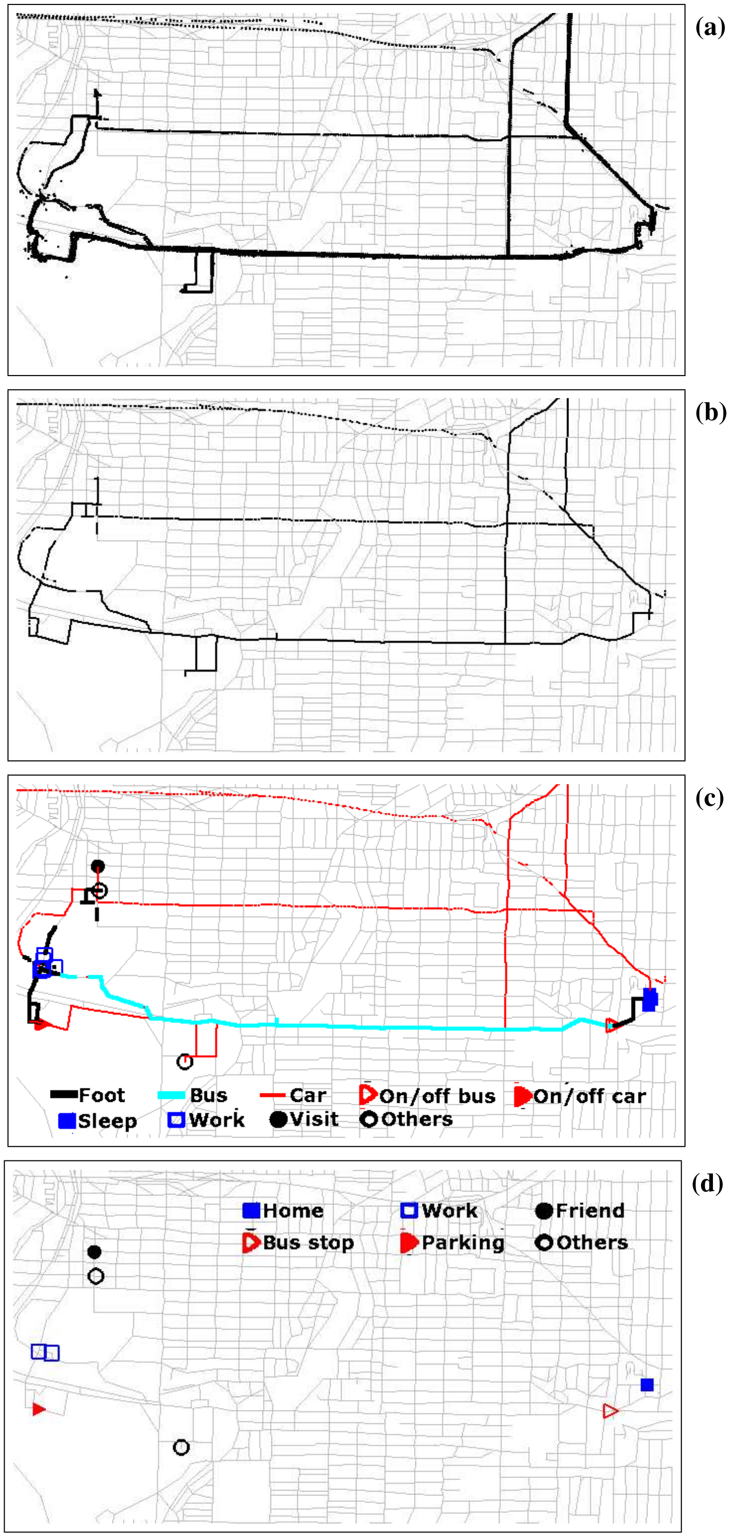

Figure 4: Illustration of inference on part of a GPS trace, which visited this 4km x 2km area several times. (a) The raw GPS data has substantial variability due to sensor noise. (b) GPS trace snapped to 10m street patches, multiple visits to the same patch are plotted on top of each other. (c) Activities estimated for each patch. (d) Places generated by clustering significant activities, followed by a determination of place types.

Time	Activity and transportation
8:15am - 8:34am	Drive from home$_1$ to parking lot$_2$, walk to workplace$_1$;
8:34am - 5:44pm	Work at workplace$_1$;
5:44pm - 6:54pm	Walk from workplace$_1$ to parking lot$_2$, drive to friend$_3$'s place;
6:54pm - 6:56pm	Pick up/drop off at friend$_3$'s place;
6:56pm - 7:15pm	Drive from friend$_3$'s place to other place$_2$;
7:15pm - 9:01pm	Other activity at other place$_2$;
9:01pm - 9:20pm	Drive from other place$_2$ to friend$_1$'s place;
9:20pm - 9:21pm	Pick up/drop off at friend$_1$'s place;
9:21pm - 9:50pm	Drive from friend$_1$'s place to home$_1$;
9:50pm - 8:22am	Sleep at home$_1$.

Table 2: Summary of a typical day based on the inference results.

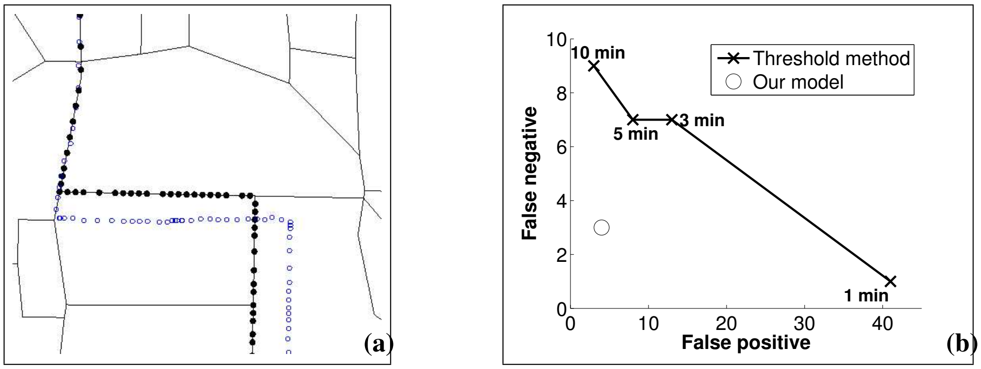

Figure 5: (a) GPS trace (gray circles) and the associated grid cells (black circles) on the street map (lines). (b) Accuracy of extracting significant places.

minute to 10 minutes. The data contained 51 different significant places. Figure 5(b) shows the false positive and false negative rates achieved with the two approaches. As can seen, our approach clearly outperforms the threshold method. Any fixed threshold is not satisfactory: low thresholds have many false negatives, and high thresholds result in many false positives. In contrast, our model performs much better: it only generates 4 false positives and 3 false negatives.

Labeling places and activities using models learned form others

Table 3 through Table 5 summarize the results achieved with our system on the cross-validation data. Table 3 shows activity estimation results on the significant activities only. An instance was considered a false positive (FP) if a significant activity was detected when none occurred, and was considered false negative (FN) if a significant activity occurred but was labeled as non-significant such as walking. The results are given for models with and without taking the detected places into account. More specifically, without places are results achieved by CRF$_0$ generated by Step 5 of the algorithm in Table 1, and results with places are those achieved after model convergence. When the results of both approaches are identical, only one number is given; otherwise, the first number gives results achieved with the complete model. The table shows two main results. First, the accuracy of our approach is quite high, especially when considering that the system was evaluated on only one week of data and was trained on only three weeks of data collected by different persons. Second, performing joint inference over activities and places increases the quality of inference. The reason for this is that a place node connects all the activities occurring in its spatial area so that these activities can be labeled in a more consistent way.

These results were generated when taking a street map into account. We also performed an analysis

	Inferred labels							
Truth	Work	Sleep	Leisure	Visit	Pickup	On/off car	Other	FN
Work	12 / 11	0	0 / 1	0	0	0	1	0
Sleep	0	21	1	2	0	0	0	0
Leisure	2	0	20 / 17	1 / 4	0	0	3	0
Visiting	0	0	0 / 2	7 / 5	0	0	2	0
Pickup	0	0	0	0	1	0	0	2
On/Off car	0	0	0	0	1	13 / 12	0	2 / 3
Other	0	0	0	0	0	0	37	1
FP	0	0	0	0	2	2	3	-

Table 3: Activity confusion matrix of cross-validation data with (left values) and without (right values) considering places for activity inference.

Experiment	No. of activities	Correctly labeled	FP	FN
Activities and places, with street map	129	111 (86.0 %)	7	5
Activities only, with street map	129	104 (80.6 %)	7	6
Activities and places, without street map	135	115 (85.2 %)	7	5

Table 4: Performance of activity estimation using different variants of our approach.

of the system without using the street map. In this case, the GPS trace was segmented into 10m segments solely based on the raw GPS values. We found that the results achieved without the street map were consistently almost identical to those achieved when a street map is available. Table 4 provides the activity inference accuracy of the different evaluations. Again, we only show significant activities, the system achieves above 90% accuracy for navigation activities such as car, walk, or bus. The first two rows provide the accuracy values for the confusion matrix shown in Table 3. The last row shows the values when no street map is available (note that the number of activities is slightly different, since the GPS trace is segmented differently). As can be seen, our approach achieves above 85% accuracy in estimating significant activities, both with and without a street map.

	Inferred labels					
Truth	Work	Home	Friend	Parking	Other	FN
Work	5	0	0	0	0	0
Home	0	4	0	0	0	0
Friend	0	0	3	0	2	0
Parking	0	0	0	8	0	2
Other	0	0	0	0	28	1
FP	0	0	1	1	2	-

Table 5: Place confusion matrix.

Finally, the confusion matrix shown in Table 5 summarizes the results achieved on detecting and labeling significant places. As can be seen, the approach commits zero errors in labeling the home and work locations of the persons used for testing. The overall accuracy in place detection and labeling is 90.6%. The place detection results were identical with and without using a street map.

6 Conclusions

We provided a novel approach to performing location-based activity recognition. In contrast to existing techniques, our approach uses one consistent framework for both low-level inference and the extraction of a person's significant places. This is done by iteratively constructing a hierarchical conditional random field, where the upper level is generated based on MAP inference on the lower level. Once a complete model is constructed, we perform joint inference in the complete CRF. Discriminative learning using pseudo-likelihood and inference using loopy belief propagation can be performed extremely efficiently in our model: The analysis of a GPS trace collected over a week takes approximately one minute on a standard desktop PC.

Our experiments based on traces of GPS data show that our system significantly outperforms existing approaches. In addition to being able to learn a person's significant locations, it can infer low level activities such as walking, working, or getting into a bus. We demonstrate that the model can be trained from a group of persons and then applied successfully to a different person, achieving more than 85% accuracy in determining low-level activities and above 90% accuracy in detecting and labeling significant places. Our model achieves virtually identical accuracy both with and without a street map. The output of our system can also be used to generate textual summaries of a person's daily activities.

The system described here opens up various research directions. For instance, our algorithm constructs the hierarchical CRF using MAP estimation. We are currently investigating a technique that generates multiple models using an MCMC or a k-best approach. The different models can then be evaluated based on their overall data likelihood. We expect this more flexible model searching approach to generate better results especially with more complex models. We are currently adding more types of sensors to our model, including data collected by a wearable multi-sensor board [19, 35]. This sensor device collects measurements such as 3-axis acceleration, audio signals, barometric pressure, and light. Using the additional information provided by these sensors, we will be able to perform extremely fine-grained activity recognition.

7 Acknowledgments

This work has partly been supported by DARPA's ASSIST and CALO Programmes (contract numbers: NBCH-C-05-0137, SRI subcontract 27-000968).

References

[1] P. Abbeel, A. Coates, M. Montemerlo, A. Ng, and S. Thrun. Discriminative training of Kalman filters. In *Proc. of Robotics: Science and Systems*, 2005.

[2] D. Anguelov, B. Taska, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng. Discriminative learning of Markov random fields for segmentation of 3d range data. In *Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[3] D. Ashbrook and T. Starner. Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, 7(5), 2003.

[4] M. Bennewitz, W. Burgard, G. Cielniak, and S. Thrun. Learning motion patterns of people for compliant robot motion. *International Journal of Robotics Research (IJRR)*, 24(1), 2005.

[5] J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24, 1975.

[6] B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer. Easyliving: Technologies for intelligent environments. *Handheld and Ubiquitous Computing*, 2000.

[7] H.H. Bui, S. Venkatesh, and G. West. Tracking and surveillance in wide-area spatial environments using the abstract hidden markov model. *International Journal of Pattern Recognition and Artificial Intelligence*, 15(1), 2001.

[8] M. Collins. Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. In *Proc. of the Conference on Empirical Methods in Natural Language Processing*, 2002.

[9] H. Dee and D. Hogg. On the feasibility of using a cognitive model to filter surveillance data. In *Proc. of the IEEE International Conference on Advanced Video and Signal Based Surveillance*, 2005.

[10] W.R. Gilks, S. Richardson, and D.J. Spiegelhalter, editors. *Markov Chain Monte Carlo in Practice*. Chapman and Hall/CRC, 1996.

[11] V. Gogate, R. Dechter, C. Rindt, and J. Marca. Modeling transportation routines using hybrid dynamic mixed networks. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005.

[12] K. Gopalratnam, H. Kautz, and D. Weld. Extending continuous time bayesian networks. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2005.

[13] R. Hariharan and K. Toyama. Project Lachesis: parsing and modeling location histories. In *Geographic Information Science*, 2004.

[14] K. Kanazawa, D. Koller, and S. J. Russell. Stochastic simulation algorithms for dynamic probabilistic networks. In *Proc. of the 11th Annual Conference on Uncertainty in AI (UAI)*, Montreal, Canada, 1995.

[15] Trausti Kristjannson, Aron Culotta, Paul Viola, and Andrew McCallum. Interactive information extraction with constrained conditional random fields. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2004.

[16] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 2001.

[17] S. Kumar and M. Hebert. Discriminative random fields: A discriminative framework for contextual interaction in classification. In *Proc. of the International Conference on Computer Vision (ICCV)*, 2003.

[18] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of the International Conference on Machine Learning (ICML)*, 2001.

[19] J. Lester, T. Choudhury, N. Kern, G. Borriello, and B. Hannaford. A hybrid discriminative-generative approach for modeling human activities. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

[20] L. Liao, D. Fox, and H. Kautz. Learning and inferring transportation routines. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2004.

[21] L. Liao, D. Fox, and H. Kautz. Hierarchical conditional random fields for GPS-based activity recognition. In *Proc. of the International Symposium of Robotics Research (ISRR)*, 2005.

[22] L. Liao, D. Fox, and H. Kautz. Location-based activity recognition. In *Advances in Neural Information Processing Systems (NIPS)*, 2005.

[23] L. Liao, D. Fox, and H. Kautz. Location-based activity recognition using relational Markov networks. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2005.

[24] A. McCallum. Efficiently inducing features of conditional random fields. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2003.

[25] K. Murphy. *Dynamic Bayesian Networks: Representation, Inference and Learning*. PhD thesis, UC Berkeley, Computer Science Division, 2002.

[26] K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 1999.

[27] A. Ng and M. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In *Advances in Neural Information Processing Systems (NIPS)*, 2002.

[28] D. Patterson, O. Etzioni, D. Fox, and H. Kautz. Intelligent ubiquitous computing to support Alzheimer's patients: Enabling the cognitively disabled. In *UbiCog '02: First International Workshop on Ubiquitous Computing for Cognitive Aids*, 2002.

[29] D. Patterson, L. Liao, K. Gajos, M. Collier, N. Livic, K. Olson, S. Wang, D. Fox, and H. Kautz. Opportunity Knocks: a system to provide cognitive assistance with transportation services. In *International Conference on Ubiquitous Computing (UbiComp)*, 2004.

[30] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., 1988.

[31] F. Peng and A. McCallum. Accurate information extraction from research papers using conditional random fields. In *HLT-NAACL*, 2004.

[32] A. Pfeffer and T. Tai. Asynchronous dynamic bayesian networks. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2005.

[33] A. Quattoni, M. Collins, and T. Darrell. Conditional random fields for object recognition. In *Advances in Neural Information Processing Systems (NIPS)*, 2004.

[34] L. R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. In *Proceedings of the IEEE*. IEEE, 1989. IEEE Log Number 8825949.

[35] A. Raj, A. Subramanya, J. Bilmes, and D. Fox. Rao-Blackwellized particle filters for recognizing activities and spatial context from wearable sensors. In *Experimental Robotics: The 10th International Symposium*, Springer Tracts in Advanced Robotics (STAR). Springer Verlag, 2006.

[36] M. Richardson and P. Domingos. Markov logic networks. Technical report, Department of Computer Science and Engineering, University of Washington, Seattle, WA, 2004. Conditionally accepted for publication in *Machine Learning*.

[37] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *Proc. of Human Language Technology-NAACL*, 2003.

[38] B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2002.

[39] C. Yanover and Y. Weiss. Most probable configurations using loopy belief propagation. In *Advances in Neural Information Processing Systems (NIPS)*, 2003.

[40] J. S. Yedidia, W. T. Freeman, and Y. Weiss. *Exploring Artificial Intelligence in the New Millennium*, chapter Understanding Belief Propagation and Its Generalizations. Morgan Kaufmann Pub, 2001.