Activity recognition using semi-Markov models on real world smart home datasets

van Kasteren, T.L.M.; Englebienne, G.; Kröse, B.J.A.

DOI
10.3233/AIS-2010-0070
Publication date
2010
Document Version
Final published version
Published in
Journal of ambient intelligence and smart environments

Journal of Ambient Intelligence and Smart Environments 2 (2010) 311–325
DOI 10.3233/AIS-2010-0070
IOS Press

Activity recognition using semi-Markov models on real world smart home datasets

T.L.M. van Kasteren *, G. Englebienne and B.J.A. Kröse
Intelligent Systems Lab Amsterdam, Science Park 107, 1098 XG, Amsterdam, The Netherlands

Abstract. Accurately recognizing human activities from sensor data recorded in a smart home setting is a challenging task. Typically, probabilistic models such as the hidden Markov model (HMM) or conditional random fields (CRF) are used to map the observed sensor data onto the hidden activity states. A weakness of these models, however, is that the type of distribution used to model state durations is fixed. Hidden semi-Markov models (HSMM) and semi-Markov conditional random fields (SMCRF) model duration explicitly, allowing state durations to be modelled accurately. In this paper we compare the recognition performance of these models on multiple fully annotated real world datasets consisting of several weeks of data. In our experiments the HSMM consistently outperforms the HMM, showing that accurate duration modelling can result in a significant increase in recognition performance. SMCRFs only slightly outperform CRFs, showing that CRFs are more robust in dealing with violations of the modelling assumptions. The datasets used in our experiments are made available to the community to allow further experimentation.

Keywords: Duration modelling, semi-Markov conditional random fields, hidden semi-Markov model, human activity recognition

1. Introduction

Recognizing human activities from smart home sensor data allows many applications, in areas such as intelligent environments [2,4] and healthcare [1,20,29]. Activities to recognize can vary from basic activities of daily living (ADLs) such as bathing and toiletting to instrumental ADLs such as shaving and brushing teeth [22,26,28].

Typically probabilistic models such as the hidden Markov model (HMM) [12,14,28] or conditional random fields (CRF) [13,23,26] are used to map the observed sensor data onto the hidden activity states. Probabilistic models are advantageous for problems such as activity recognition, because they allow us to deal with the noise and uncertainty in a principled manner. A weakness of HMMs and CRFs, however, is their modelling of state durations. State durations are modelled implicitly by means of self-transitions of states, and this entails a number of important limitations [10,15]. In real-world problems such as activity recognition, it may be important to model state duration accurately. For example, shaving and brushing teeth might both involve the use of the bathroom door and the faucet. Because it is difficult to put sensors on the toothbrush or razor it is difficult to distinguish these activities based on sensor data. However, because shaving typically takes up more time than brushing teeth, the duration of the activity is likely to be very informative for recognition.

Hidden semi-Markov models (HSMM) and semi-Markov conditional random fields (SMCRF) are both semi-Markov models in which duration is modelled explicitly. In this paper we compare the recognition performance of these models on multiple fully annotated real world datasets consisting of several weeks of data. We compare both generative and discriminative semi-Markov models to their conventional counterparts (see Table 1). Generative models such as the HMM are a classic way of modelling a sequential process probabilistically. However, discriminative models such as CRFs have become increasingly popular in re-

[*]Corresponding author. E-mail: tim0306@gmail.com.

Table 1
Categorization of hidden Markov model (HMM), hidden semi-Markov model (HSMM), conditional random field (CRF) and semi-Markov conditional random field (SMCRF)

	Conventional	Semi-Markov
Generative	HMM	HSMM
Discriminative	CRF	SMCRF

cent years because they have been shown to outperform generative models in various domains [6,17].

The main contribution in this paper is a thorough comparison of the performance of HMMs, CRFs, HSMMs and SMCRFs on real world activity recognition data. Previous work has separately evaluated HSMMs [5] and SMCRFs [11,24] on simulated and laboratory data, but our work is the first to evaluate these models on real world activity recognition data and compare their performance in a single paper. By comparing these models using the same experimental setup we are able to draw clear conclusions about their recognition performance and whether they are suitable for activity recognition or not.

The remainder of this paper is organized as follows. In Section 2 we discuss related work. Section 3 describes the HMM and HSMM together with their learning and inference algorithms. Section 4 describes CRFs and SMCRFs together with their learning and inference algorithms. In Section 5 we highlight the differences between these models. And in Section 6 we present the experiments and results using our real world datasets. Finally, in Section 7 we sum up our conclusions.

2. Related work

Most early work on activity recognition used HMMs to recognize the activities from sensor data [14,28]. After CRFs had been shown to outperform HMMs in other areas [6], they were applied to activity recognition as well [25,26]. It become clear that activity recognition models needed a way to model long term dependencies. Rabiner had already pointed out that the modelling of duration was one of the major weaknesses of HMMs and suggested the use of HSMMs [15]. Murphy further generalized the formulation of semi-Markov models by representing them as dynamic Bayesian networks [10]. In work by Duong et al. HSMMs were applied to activity recognition [5]. They compared performance of the model using various duration distributions and suggested the use of the Coxian distribution for computational efficiency. The SMCRF was first applied to information extraction and compared with CRFs. SMCRFs generally outperformed CRFs, although for some topics the gain was minimal. Truyen applied a hierarchical version of SMCRFs to activity recognition using a small dataset recorded in a laboratory setting [24]. The performance was compared to hierarchical CRF and a conventional CRF, the hierarchical SMCRF outperformed both. The models were also applied to POS tagging in which the performance gain was very little. Finally, hierarchical models have also been proposed to deal with the long term dependencies. Although no explicit duration modelling is done in these models, the sequence of substates allows more complex duration modelling than in conventional models [18].

There is a strong need for real world datasets in the activity recognition community. A lot of work is evaluated on laboratory datasets [5,14,24] and simulated data [9,11]. A few large real world datasets are publicly available. A dataset of four weeks of fully annotated data was published at Ubicomp [26], in which a wireless sensor network was used to observe the inhabitants behaviour and annotation was done using a bluetooth headset. Tapia et al. recorded two datasets using a large number of wireless sensor nodes in two separate homes [22]. Each dataset contains two weeks of sensor data and annotation. Annotation was done using hand written activity logs, which was corrected by inspection of the sensor data to improve the quality of the annotation. Finally, a few datasets were recorded in the PlaceLab, a house fully equipped with sensors and cameras. A four hour dataset was recorded in which a researcher was asked to perform a number of activities at his own pace and sequence. Cameras in the house were used to record the behaviour and later annotate the dataset [21]. Another dataset consists of two months of data in which a couple was hired to live in the PlaceLab for the duration of the dataset. Only one of the two people's activities were annotated due to limited funding [8].

This work compares semi-Markov models to their conventional counterparts and uses real world datasets to evaluate the model performances. By comparing all these models in a single paper we can clearly see the impact of modelling duration accurately. The use of real world datasets allows us to see how strong these effects are in a real world setting.

3. Hidden Markov model and hidden semi-Markov model

The hidden Markov model (HMM) and hidden semi-Markov model (HSMM) are both generative models. This means these models fully specify the dependency relations among the variables by defining a factorization of the joint probability over these variables. They differ in the sense that HMMs only model the observations and transitions between hidden states, while HSMMs also model the duration of hidden states explicitly.

In this section we first give the model definitions of the HMM and HSMM and then briefly describe their inference and learning algorithms. We assume our observations are binary, for convenience, and because the datasets used for evaluation consist only of binary observations. However, other types of observations can be used with these models and would only change the observation distribution.

3.1. Hidden Markov model

The HMM is a generative probabilistic model consisting of a hidden variable y and an observable variable x at each time step (Fig. 1). In our case the hidden variable is the activity performed, and the observable variable is the vector of sensor readings. We define a sequence of observations $\mathbf{x}_{1:T} = \{\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_T\}$ with $\vec{x}_t = (x_t^1, x_t^2, \ldots, x_t^N)^T$ and $x_t^i \in \{0, 1\}$. The corresponding sequence of hidden states is represented as $\mathbf{y}_{1:T} = \{y_1, y_2, \ldots, y_T\}$ where for Q possible states $y_t \in \{1 \ldots Q\}$.

Generative models provide an explicit representation of dependencies by specifying the factorization of the joint probability of the hidden and observable variables $p(\mathbf{y}_{1:T}, \mathbf{x}_{1:T})$. In the case of HMMs there are two dependence assumptions that define this model, represented with the directed arrows in Fig. 1.

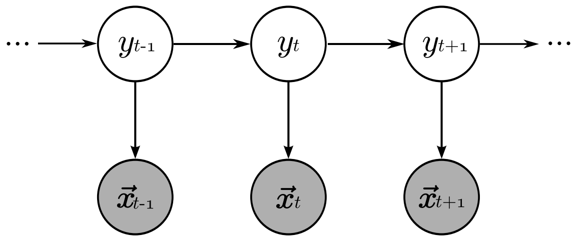


Fig. 1. The graphical representation of a HMM. The shaded nodes represent observable variables, while the white nodes represent hidden ones.

– The hidden variable at time t, namely y_t, depends only on the previous hidden variable y_{t-1} (*first order Markov assumption* [15]).
– The observable variable at time t, namely x_t, depends only on the hidden variable y_t at that time slice.

The joint probability therefore factorizes as follows

$$p(\mathbf{y}_{1:T}, \mathbf{x}_{1:T}) = \prod_{t=1}^{T} p(\vec{x}_t \mid y_t) p(y_t \mid y_{t-1})$$

where we have used $p(y_1 \mid y_0) = p(y_1)$ for the sake of notational simplicity.

The different factors further specify the workings of the model. The initial state distribution $p(y_1)$ is a conditional probability table with individual values denoted as $p(y_1 = i) \equiv \pi_i$. The observation distribution $p(\vec{x}_t \mid y_t)$ represents the probability that the state y_t would generate observation vector $\vec{x}_t$. Each sensor observation is modelled as an independent Bernoulli distribution, where μ_{in} is the parameter of the nth sensor for state i. The transition probability distribution $p(y_t \mid y_{t-1})$ represents the probability of going from one state to the next. This is given by a conditional probability table A where individual transition probabilities are denoted as $p(y_t = j \mid y_{t-1} = i) \equiv a_{ij}$. The HMM is therefore fully specified by the parameters $A = \{a_{ij}\}$, $B = \{\mu_{in}\}$ and $\pi = \{\pi_i\}$.

A weakness of conventional HMMs is its lack of flexibility in modelling state durations. Given an HMM in a known state, the probability that it stays in that state for d timeslices is

$$p_i(d) = (a_{ii})^{d-1}(1 - a_{ii})$$

where $p_i(d)$ is the discrete probability density function (PDF) of duration d in state i and a_{ii} is the self-transition probability of state i [15]. This duration density function takes the form of a geometric distribution with a mode fixed at one. Because it follows implicitly from the model definition, it is an inherent property of the model and we cannot use a different distribution to model duration. This severely limits our modelling options. For example, the activity showering typically takes up several minutes, to shower in one minute is very unlikely. The geometric distribution, however, cannot represent distributions where shorter durations are less probable than longer ones. Modeling the duration of such an activity using, for example, a Gaussian distribution would be more appropriate.

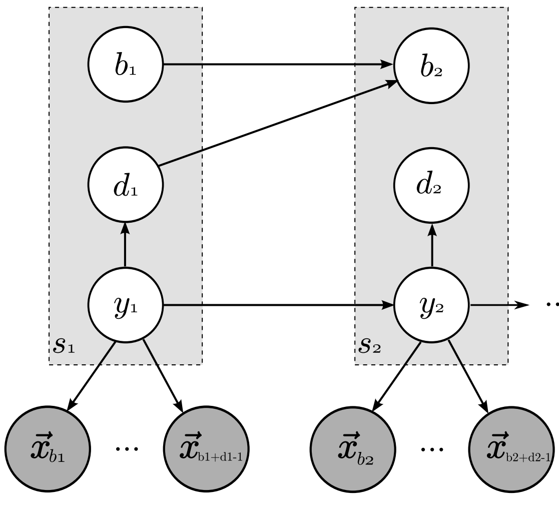


Fig. 2. The graphical representation of a HSMM. The shaded nodes represent observable variables, while the white nodes represent hidden ones.

3.2. Hidden semi-Markov model

Hidden semi-Markov models (HSMMs) are HMMs in which the duration of a state is modelled explicitly. We define a sequence of U segments $\mathbf{s}_{1:U} = \{s_1, s_2, \dots, s_U\}$ where segment $s_j = (b_j, d_j, y_j)$ consists of start position b_j, duration d_j and hidden state $y_j \in \{1 \dots Q\}$. This means timeslices b_j to $b_j + d_j$ (exclusive) have state label y_j. Segments have a positive duration and completely cover the time span $1 : T$ without overlap. Therefore, the following constraints hold: $b_1 = 1$, $\sum_{u=1}^{U} d_u = T$ and $b_{j+1} = b_j + d_j$. For example, the sequence $y_{1:8} = \{1,1,1,2,2,1,2,2\}$ can be represented as segmentation $s_{1:4} = \{(1,3,1),(4,2,2),(6,1,1),(7,2,2)\}$. Note that we do not constrain consecutive segments to differ in label (i.e. we allow self-transition of segments). HSMMs have been proposed before where such a constraint was enforced [15], but in our application activities might be repeated shortly after each other (e.g., getting another drink), so that it makes more sense to model the data with self-transitions.

The hidden variables are now represented as segments $\mathbf{s}_{1:U}$ while the observable variables are still the sensor readings $\mathbf{x}_{1:T}$ (Fig. 2). The joint probability of this model factorizes as follows

$$p(\mathbf{s}_{1:U}, \mathbf{x}_{1:T}) = p(\mathbf{y}_{1:U}, \mathbf{b}_{1:U}, \mathbf{d}_{1:U}, \mathbf{x}_{1:T})$$

$$= p(y_1)p(b_1)p(d_1 \mid y_1) \prod_{t=b_1}^{b_1+d_1-1} p(x_t \mid y_1)$$

$$\prod_{u=2}^{U} p(y_u \mid y_{u-1}) p(b_u \mid b_{u-1}, d_{u-1})$$

$$p(d_u \mid y_u) \prod_{t=b_u}^{b_u+d_u-1} p(x_t \mid y_u) \quad (1)$$

The transition probability distribution $p(y_u \mid y_{u-1})$ now represents the probability of going from one segment to the next. It is still given by a conditional probability table where individual transition probabilities are denoted as $p(y_u = j \mid y_{t-u} = i) \equiv a_{ij}$.

The starting point variable b_u is a bookkeeping variable to keep track of the starting point of a segment. The first segment always starts at 1, consecutive starting points are deterministically calculated from the previous starting point and the previous duration.

$$p(b_1 = m) = \delta(m, 1)$$
$$p(b_u = m \mid b_{u-1} = n, d_{u-1} = l)$$
$$= \delta(m, n + l)$$

where $\delta(i, j)$ is the Kronecker delta function, giving 1 if $i = j$ and 0 otherwise. The duration probability distribution is defined as $p(d_u = l \mid y_u = i) = p_i(l)$. Different distributions are possible, but in this work we use a Gaussian $p_i(l) = \mathcal{N}(\mu, \sigma)$. The initial state distribution and observation probabilities are parameterised in the same way as for HMMs.

3.3. Inference

The inference problem for HMMs consists of finding the single best state sequence (path) that maximizes $p(\mathbf{y}_{1:T}, \mathbf{x}_{1:T})$. Although the number of possible paths grows exponentially with the length of the sequence, the best state sequence can be found efficiently using the Viterbi algorithm. Using dynamic programming, we can discard a number of paths at each time step, resulting in a computational complexity of $O(TQ^2)$ for the entire sequence, where T is the total number of timeslices and Q the number of states [15].

In the case of HSMMs we need to adapt and extend the Viterbi algorithm to deal with segments. This means the algorithm also needs to iterate over all possible durations at each timestep. The complete procedure has a computational complexity of $O(TQ^2D)$, where

D is the maximum duration a segment can have [10]. In Appendix A.1 the details of the Viterbi algorithm for HSMMs is given.

3.4. Parameter learning

The model parameters are learned by finding the maximum likelihood parameters. Given some training data $\mathbf{x}_{1:T}, \mathbf{y}_{1:T}$, we want to find those parameters that maximize $p(\mathbf{x}_{1:T}, \mathbf{y}_{1:T} \mid \theta)$. This is equivalent to finding the maximum likelihood parameters of each of the factors that make up the joint probability.

Our training data are fully labelled, with exact start and end time of each segment, so that we can optimize the parameters in closed form for both HMMs and HSMMs. The observation probability $p(x^n \mid y = i)$ follows a Bernoulli distribution whose maximum likelihood parameter estimation is given by

$$\mu_{ni} = \frac{\sum_{t=1}^{T} x_t^i \delta(y_t, i)}{\sum_{t=1}^{T} \delta(y_t, i)}$$

where T is the total number of data points and $\delta(i, j)$ is the Kronecker delta function. The transition probability $p(y_t = j \mid y_{t-1} = i)$ is a multinomial distribution whose parameters can be calculated by

$$a_{ij} = \frac{\sum_{t=2}^{T} \delta(y_t, j)\delta(y_{t-1}, i)}{\sum_{t=2}^{T} \delta(y_{t-1}, j)}$$

where T is equal to the number of timeslices in the case of HMMs and equal to the number of segments in the case of HSMMs.

4. Conditional random fields and semi-Markov conditional random fields

Conditional random fields (CRFs) and semi-Markov conditional random fields (SMCRFs) are discriminative models. Rather than modelling the full joint probability $p(\mathbf{y}_{1:T}, \mathbf{x}_{1:T})$, discriminative models model the conditional probability $p(\mathbf{y}_{1:T} \mid \mathbf{x}_{1:T})$ directly. Since the observation sequence $\mathbf{x}_{1:T}$ is always given, its distribution needs not be modelled. We first give the definition of CRFs and SMCRFs and then describe their inference and learning algorithms.

4.1. Conditional random fields

To be able to define CRFs for arbitrary graphs we use the clique representation from graph theory. A clique is defined as a subgraph in which every vertex is connected to every other vertex in the subgraph. Furthermore, a maximal clique is a clique to which no other node in the graph can be added without it ceasing to be a clique. We can partition any given graph into a collection of maximal cliques [3]. The conditional probability distribution $p(\mathbf{y}_{1:T} \mid \mathbf{x}_{1:T})$ is factorized as a product of clique potentials

$$p(\mathbf{y}_{1:T} \mid \mathbf{x}_{1:T}) = \frac{1}{Z(\mathbf{x}_{1:T})} \prod_{c \in C} \phi_c(\mathbf{y}_c, \mathbf{x}_c),$$

where C is the set of cliques that make up the entire graph and c is a single clique. The function $\phi_c(\mathbf{y}_c, \mathbf{x}_c)$ is called the clique potential and is a function of the observed nodes $\mathbf{x}_c$ and the hidden nodes $\mathbf{y}_c$ in clique c. The clique potentials are non-negative, and the partition function $Z(\mathbf{x}_{1:T})$ is a normalization term that ensures that the probabilities sum up to one [19]. It is calculated by summing over all possible state sequences

$$Z(\mathbf{x}_{1:T}) = \sum_{\mathbf{y}_{1:T}} \prod_{c \in C} \phi_c(\mathbf{y}_c, \mathbf{x}_c).$$

In general, the computation of $Z(\mathbf{x}_{1:T})$ is intractable, because the number of possible distinct state sequences grows exponentially with the length of the sequence. However, in the case of linear chain CRFs the normalization term can be efficiently calculated using the forward-backward algorithm [19].

Linear-chain CRFs are the discriminative analog of HMMs in which the maximal cliques are restricted to contain either one observation and the corresponding state, or a state and the previous state (see Fig. 3). The clique potentials are, therefore, of the form

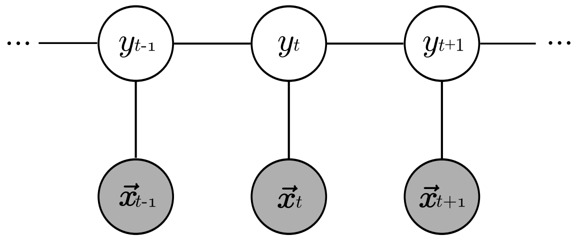


Fig. 3. The graphical representation of a linear-chain CRF. The shaded nodes represent observable variables, while the white nodes represent hidden ones.

$\phi_t(y_t, y_{t-1}, \mathbf{x}_t)$, so that the conditional likelihood of a CRF can be written as

$$p(\mathbf{y}_{1:T} \mid \mathbf{x}_{1:T}) = \frac{1}{Z(\mathbf{x}_{1:T})} \prod_{t=1}^{T} \phi_t(y_t, y_{t-1}, \mathbf{x}_t)$$

The clique potentials are parameterised as

$$\phi_t(y_t, y_{t-1}, \mathbf{x}_t) = \exp \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, \mathbf{x}_t),$$

where K is the number of feature functions used to parameterise the distribution, λ_k is a weight parameter and $f_k(y_t, y_{t-1}, \mathbf{x}_t)$ a feature function. The exponent enforces the non-negativity of the potential. This choice results in using a PDF from the exponential family to model the distribution $p(\mathbf{y}_{1:T} \mid \mathbf{x}_{1:T})$. We will see below how the parameters of this PDF are optimized by gradient descent.

4.2. Semi-Markov conditional random fields

Similarly to the HSMMs, the semi-Markov Conditional Random Field (SMCRF) models the duration of states explicitly by considering the probability of segments rather than individual timeslices. We use the same notation as with HSMMs in which segment $s_j = (b_j, d_j, y_j)$ consists of start position b_j, duration d_j and state $y_j \in \{1 \ldots Q\}$. The factorization of the conditional probability is defined as

$$\begin{aligned} p(\mathbf{s}_{1:U} \mid \mathbf{x}_{1:T}) &= p(\mathbf{y}_{1:U}, \mathbf{b}_{1:U}, \mathbf{d}_{1:U} \mid \mathbf{x}_{1:T}) \\ &= \frac{1}{Z(\mathbf{x}_{1:T})} \\ &\quad \prod_{u=1}^{U} \phi_u(y_u, y_{u-1}, \mathbf{x}_u, d_u) \end{aligned}$$

resulting in the model depicted in Fig. 4.

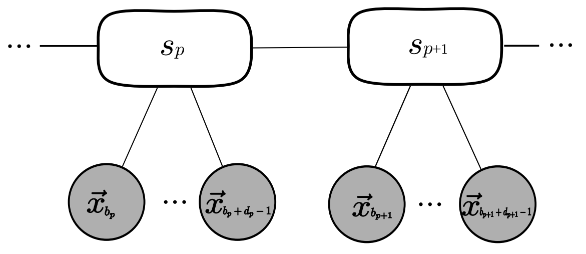


Fig. 4. The compact graphical representation of a semi-Markov CRF, showing segments s and observations $\vec{x}$. The shaded nodes represent observable variables, while the white nodes represent hidden ones.

The segment duration can be modelled using distributions from the exponential family, by selecting appropriate feature functions. We use feature functions of the form $g_i(y_u, d_u) = \delta(y_u, i) \cdot d_u^2$, $g_i'(y_u, d_u) = \delta(y_u, i) \cdot d_u$ and $g_i''(y_u, d_u) = \delta(y_u, i) \cdot 1$ which gives a contribution proportional to $\exp(d - \mu)^2$, for appropriate values of λ_i,λ_i' and λ_i''. Notice that this does not enforce a Gaussian distribution; the parameter learning will find the best distribution, which will only be Gaussian if that is what is present in the data.

4.3. Inference

To perform inference in CRFs the Viterbi algorithm is used, which for CRFs also has a computational complexity of $O(TQ^2)$ [19]. Inference in SMCRFs is done using Viterbi similar to HSMMs and has a computational complexity of $O(TQ^2D)$ [16]. The details of the Viterbi algorithm for SMCRFs are given in Appendix A.2.

4.4. Parameter learning

The parameters $\theta = \{\lambda_1, \ldots, \lambda_K\}$ of CRFs are learned by maximizing the conditional log likelihood $l(\theta) = \log p(\mathbf{y}_{1:T} \mid \mathbf{x}_{1:T}, \theta)$ given by

$$\begin{aligned} l(\theta) =& \sum_{t=1}^{T} \sum_{k=1}^{K} \lambda_k f_k(y_t, y_{t-1}, \vec{x}_t) \\ & - \log Z(\mathbf{x}_{1:T}) - \sum_{k=1}^{K} \frac{\lambda_k^2}{2\sigma^2} \end{aligned}$$

where the final term is a regularization term penalizing large values of λ to prevent overfitting. The constant σ is set beforehand and determines the strength of the penalization [19].

The function $l(\theta)$ is concave, which follows from the convexity of $\log Z(\mathbf{x}_{1:T})$ [19]. A useful property of convex functions in parameter learning is that any local optimum is also a global optimum. Quasi-Newton methods such as BFGS have been shown to be suitable for CRFs [17,27]. These methods approximate the Hessian, the matrix of second derivatives, by analyzing successive gradient vectors. Because the size of the Hessian is quadratic in the number of parameters, storing the full Hessian is memory-intensive. We therefore use a limited-memory version of BFGS [7]. The partial derivative of $l(\theta)$ with respect to λ_i, is given by

$$\frac{\partial l}{\partial \lambda_i} = -\frac{\lambda_i}{\sigma^2} + \sum_{t=1}^{T} f_i(y_t, y_{t-1}, \vec{x}_t) - \sum_{t=1}^{T} \sum_{y_t, y_{t-1}} p(y_t, y_{t-1} \mid \vec{x}_t) f_i(y_t, y_{t-1}, \vec{x}_t)$$

For semi-CRF the same techniques can be used except states are replaced by segments [16].

5. Model comparison

The models introduced in the previous sections differ in terms of duration modelling, learning method and computational complexity for learning and inference. In this section we highlight these differences and discuss their consequences.

5.1. Learning in generative models

The difference between generative models (e.g. HMM, HSMM) and discriminative models (e.g. CRF, SMCRF) lies primarily in the way the model parameters are learned. The generative approach of learning the model parameters θ is by maximizing the joint probability $p(\mathbf{y}_{1:T}, \mathbf{x}_{1:T} \mid \theta) = p(\mathbf{x}_{1:T} \mid \mathbf{y}_{1:T}, \theta) p(\mathbf{y}_{1:T} \mid \theta)$.

In Section 3.1 we have seen that duration in an HMM is modelled as a geometric distribution. Here, we illustrate how the use of maximum likelihood parameters of the geometric PDF can result in classification errors. We illustrate this using an artificial example in which we try to classify two states that can only be distinguished based on their duration. The state durations have truncated Gaussian distributions, but we model them using geometric distributions.

The geometric distribution in an HMM takes the form $p_i(d) = (a_{ii})^{d-1}(1-a_{ii})$, where d is the duration of a state and a_{ii} the self-transition probability of that state. The maximum likelihood estimation for this distribution can be found using moment matching on the first moment (the mean) of the distributions. The mean of the geometric distribution is calculated as follows:

$$\mathbf{E}[d] = \sum_{d=1}^{\infty} d(a_{ii})^{d-1}(1 - a_{ii}) = \frac{1}{1 - a_{ii}}.$$

To illustrate how the use of the geometric distribution can lead to classification errors, we have plotted the real distribution of the duration of two states, given

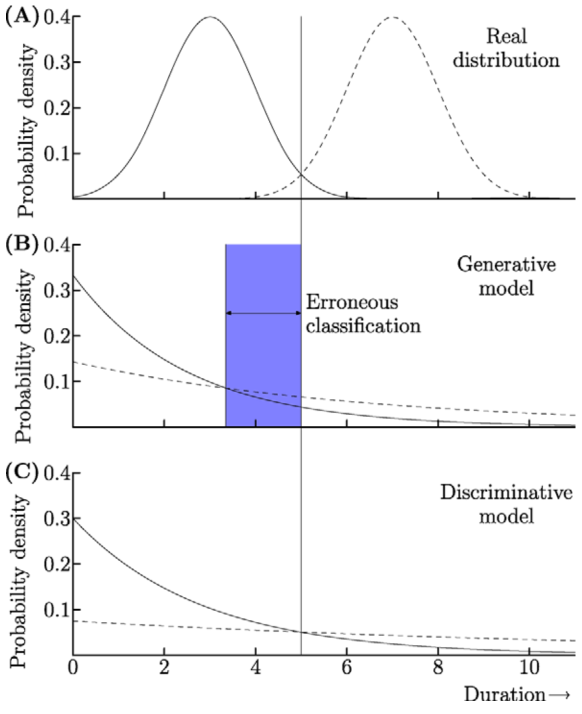


Fig. 5. Plots of (a) Gaussian distribution for means 3 (straight line) and 7 (dashed line). (b) Geometric distribution with a set of parameters learned using maximum likelihood estimation, typically used in generative models. The shaded area shows where the incorrect use of the geometric distribution leads to misclassification. (c) Geometric distribution with a set of parameters learned using discriminative models.

by two Gaussians with unit variance and means 3 and 7, in Fig. 5a. The two geometric distributions with corresponding means are shown in Fig. 5b. This would be the distribution learned by the HMM. The shaded area shows where the use of the geometric distribution results in misclassification. This illustrates how the modelling of duration with an incorrect PDF can lead to misclassification in the HMM.

5.2. Learning in discriminative models

In discriminative models we learn the model parameters by maximizing the conditional distribution $p(\mathbf{y}_{1:T} \mid \mathbf{x}_{1:T}, \theta)$. This distribution is also used for inference and directly models how the classes are distributed given an observation. Because we are directly optimizing this quantity we obtain the set of parameters that discriminates the classes as well as possible. In terms of the example given in the previous subsection this corresponds to minimizing the erroneous classification area and can therefore yield a solution as shown in Fig. 5c.

This example shows an important property of discriminative models, namely their robustness towards violations of the modelling assumptions. Even though we model Gaussian distributed data using a geometric distribution, still a set of parameters is found to correctly classify the data.

5.3. Computational complexity

An important consequence of using discriminative models is the increase in computational complexity during learning. In generative models the parameters of the distributions used can usually be estimated using a closed form solution. Discriminative models on the other hand require numerical methods because no closed form solution is available. This requirement is especially costly because during each iteration of the learning phase we need to perform inference, to calculate the normalization term.

The use of semi-Markov models introduces an additional computational complexity. Because the durations of activities are not known for a novel sequence of observations, all possible durations need to be considered at each timestep. This makes the computational complexity of doing inference in semi-Markov models a factor D higher than in conventional models, where D is the maximum duration of one segment. This affects both HSMMs and SMCRFs, but when fully labelled data are available, the HSMM does not need to perform inference during training. Since the inference step is performed at each iteration during learning in the SMCRF, finding the model parameters for this model is very expensive.

In the experiments section we report the amount of time needed for inference and learning in each of the models to illustrate these differences.

6. Experiments

The goal of our experiments is to evaluate which model performs best in activity recognition on a real world data, and why it performs best. To this end we evaluate the performance of each of the models on four real world datasets. In this section we first give a description of the data and provide details of our experimental setup. Then we present the results and discuss the outcome.

Previous work on activity recognition shows that most recognition confusion occurs between activities taking place in the same room using the same set of sensors [26]. Installing extra sensors to further distinguish between activities is not always possible because it is difficult to install a sensor on, for example, a tooth brush or to sense which item is retrieved from a refrigerator. To clearly show how duration modelling can help recognition performance we used a total of four datasets in two houses. Two datasets were extracted from the publicly available Ubicomp dataset [26] recorded in the home of a 26 year old male (house 1). We extracted a kitchen dataset and a bathroom dataset, both consisting of four weeks of sensor data and annotation. The other two datasets were recorded in the home of a 57 year old male (house 2), also in the kitchen and bathroom, and both consist of at least two weeks of sensor data and annotation (Table 2). These datasets are available for download from http://sites.google.com/site/tim0306/.

Table 2
Details of the houses in which the datasets were recorded

	House 1	House 2
Age	26	57
Gender	Male	Male
Setting	Apartment	House
Rooms	3	6
Duration	25 days	19 days

6.1. Real world datasets

The sensor data are recorded using a wireless sensor network consisting of a number of nodes which communicate with a central gateway. Each node consists of a small wireless network device (Fig. 6) and a sensor. Sensors we used include: reed switches to measure the open-closed state of doors and cupboards; mercury contacts for movement of objects (e.g. drawers); pas-

Fig. 6. Wireless network node to which sensors can be attached.

sive infrared (PIR) to detect motion in a specific area; float sensors to measure the toilet being flushed. The sensors all output binary values, either because they are binary in nature or because some threshold is applied to the analog value. An overview of the sensors for each dataset is given in Table 4.

Annotation for the house 1 datasets was recorded using a bluetooth headset combined with speech recognition. The inhabitant could record the start and end point of an activity by pressing a button on the head set and saying which activity was being performed [26]. In house 2 a handwritten activity diary was used for annotation. A complete list of the annotated activities per dataset is shown in Table 5. Timeslices for which no annotation is available are collected in a separate activity labelled as 'other activity'. This activity takes up an average of 95% of the time in each dataset, since most of the time people are not involved in kitchen or bathroom activities. Sensors are primarily installed on doors and cupboards. In the toilet cisterns float sensors were installed and in the dataset 'Bathroom2' a PIR sensor aimed at the bathtub and shower area was used. The floor plan, showing the location of the sensors, for house 1 can be found in Fig. 7 and for house 2 in Fig. 8.

Data obtained from the sensors are discretised in timeslices of length $\Delta t = 60$ seconds. This time slice length is long enough to provide a discriminative sensor pattern and short enough to provide high resolution labelling results. We do not use the raw sensor data as observations, instead we use the *change point* and *last sensor* representations, which have been shown to give much better results in activity recognition [26]. The change point representation assigns a 1 to timeslices where the sensor changes state and a 0 otherwise.

Table 3

Confusion Matrix showing the true positives (TP), total of true labels (TT) and total of inferred labels (TI) for each class

	Inferred			
True	1	2	3	
1	TP_1	ϵ_{12}	ϵ_{13}	TT_1
2	ϵ_{21}	TP_2	ϵ_{23}	TT_2
3	ϵ_{31}	ϵ_{32}	TP_3	TT_3
	TI_1	TI_2	TI_3	$Total$

Table 4

List of sensors used in each of the datasets. PIR is short for 'passive infrared', the toilet flush uses a float sensor, all other sensors use reed switches to measure the open-close state

Bathroom1 Sensors	Bathroom2 Sensors	Kitchen1 Sensors	Kitchen2 Sensors
Bathroom door	Bathroom door	Microwave	Microwave
Toilet door	Toilet flush	Refrigerator	Refrigerator
Bedroom door	Bathtub PIR	Freezer	Freezer
	Dresser PIR	Cupboard with plates	Cupboard with plates
		Cupboard with cups	Cupboard with cups
		Cupboard with pans	Cupboard with pans
		Cupboard with groceries	Cupboard with boxes
		Dishwasher	Cutlery drawer

Table 5

The activities that were annotated in the different datasets. The 'Num.' column shows the number of times the activity occurs in the dataset. All unannotated timeslices were collected in a single 'Other' activity. The bathroom1 and kitchen1 datasets were recorded in the home of a 26 year old male; the bathroom2 and kitchen2 datasets were recorded in the home of a 57 year old male. The word 'dw.' is short for dishwasher

Bathroom1		Bathroom2		Kitchen1		Kitchen2	
Activity	Num.	Activity	Num.	Activity	Num.	Activity	Num.
Brush teeth	16	Brush teeth	26	Breakfast	20	Breakfast	18
Showering	23	Showering	10	Dinner	9	Dinner	11
Toileting	114	Bathing	4	Snack	12	Snack	9
Other	-	Shaving	7	Drink	20	Drink	10
		Other	-	Load dw.	5	Other	-
				Unload dw.	4		
				Other	-		

While the last sensor representation continues to assign a 1 to the last sensor that changed state until a new sensor changes state.

We split our data into a test and training set using a 'leave one day out' approach. In this approach, one full day of sensor readings is used for testing and the remaining days are used for training. A day of sensor data starts at 8 am and ends at 12 pm, since hardly any activities take place outside these hours. We cycle over all the days and report the average performance measure.

We evaluate the performance of our models using precision, recall and F-measure. These measures can be calculated using the confusion matrix shown in Table 3. The diagonal of the matrix contains the true positives (TP), while the sum of a row gives us the total of true labels (TT) and the sum of a column gives us the total of inferred labels (TI). We calculate the precision and recall for each class separately and then take the average over all classes.

$$\text{Precision} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TI_i}$$

$$\text{Recall} = \frac{1}{N}\sum_{i=1}^{N}\frac{TP_i}{TT_i}$$

$$\text{F-Measure} = \frac{2 \cdot precision \cdot recall}{precision + recall}$$

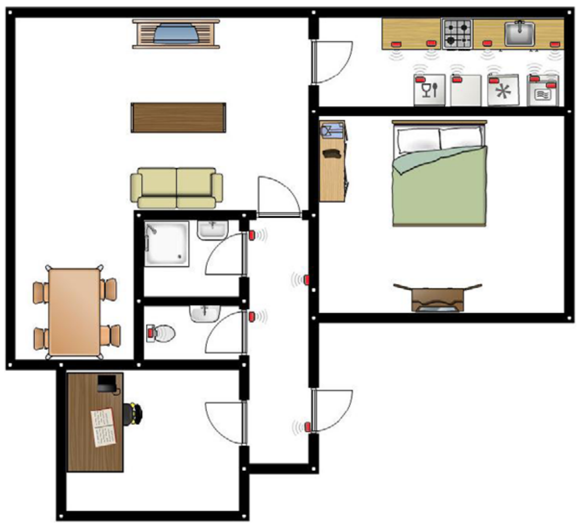
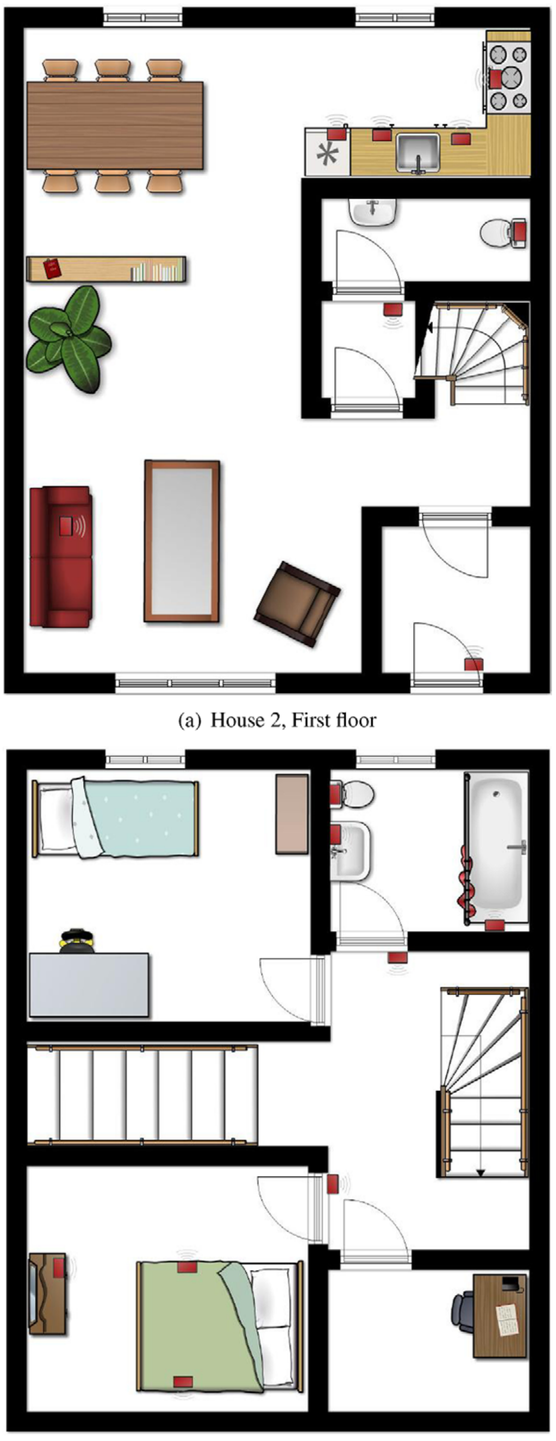


Fig. 7. Floor plan of house 1, the small rectangular boxes show the locations of the sensors.



(a) House 2, First floor

(b) House 2, Second floor

Fig. 8. Floorplan of the first and second floor of house 2, the small rectangular boxes show the locations of the sensors.

Table 6

Precision, recall and F-measure for hidden Markov model (HMM), hidden semi-Markov model (HSMM), conditional random field (CRF) and semi-Markov conditional random field (SMCRF). Experiments were performed on four different real world datasets: Bathroom1, Kitchen1, Bathroom2 and Kitchen2. Bold value indicates the highest value in the column

	Bathroom1			Kitchen1		
Model	Precision	Recall	F-Measure	Precision	Recall	F-Measure
HMM	50.2	66.9	57.3	53.4	66.3	59.2
HSMM	69.5	**84.7**	**76.4**	58.6	**71.8**	64.5
CRF	72.6	73.7	73.1	**63.4**	70.9	**67.0**
SMCRF	**75.3**	74.7	75.0	63.1	70.9	66.8
	Bathroom2			Kitchen2		
Model	Precision	Recall	F-Measure	Precision	Recall	F-Measure
HMM	48.4	57.3	52.5	45.9	49.0	47.4
HSMM	60.0	**68.9**	**64.1**	53.8	**61.0**	**57.1**
CRF	53.3	61.7	57.2	**55.0**	56.7	55.8
SMCRF	**60.3**	65.2	62.7	53.1	57.1	55.0

Table 7

Computation time for learning and inference on the ‘Bathroom1’ dataset

Model	HMM	HSMM	CRF	SMCRF
Learning Time	2.9 seconds	5.5 seconds	63.6 minutes	54.9 hours
Inference Time	21.6 seconds	37.4 seconds	26.7 seconds	35.3 seconds

The maximum duration D used by the semi-Markov models is determined by taking the maximum duration in the training set and adding 25% to account for outliers.

6.2. *Results on real world data*

The results of our models on the four real world datasets are shown in Table 6. They show that HSMMs, CRFs and SMCRFs outperform HMMs in terms of precision and recall on all datasets. Furthermore, HSMMs consistently outperform CRFs and SMCRFs in terms of recall and in three of the four datasets in terms of F-measure, but never on precision. Finally, SMCRFs outperform CRFs on two of the four datasets in terms of precision and recall.

We performed a one-tail student t-test over all of the days of the cross validation. At a confidence interval of 95% the HSMM significantly outperforms the HMM on all the datasets and in all the measures. CRFs and SMCRFs significantly outperform HMMs in all measures on the ‘bathroom1’ dataset and in terms of recall and F-measure on the ‘kitchen2’ dataset.

The confusion matrices for all models on the ‘bathroom1’ dataset further illustrate the differences between the models (Tables 8, 9, 10 and 11). We see the CRF and SMCRF perform much better than the HMM with respect to the ‘other’ activity, but perform worse in terms of the remaining activities. In fact they completely fail to model the ‘brush’ activity. The HSMM also performs much better than the HMM with respect to the ‘other’ activity, but also performs better in recognizing the ‘toilet’ and ‘brush’ activity. Finally, the SMCRF performs slightly better than the CRF in all activities.

Table 7 shows the computation times for learning and inference on the ‘Bathroom1’ dataset. We see that learning model parameters of generative models is considerably faster than discriminative models. Furthermore, inference in semi-Markov models takes longer than conventional models.

6.3. *Discussion*

Our results show that the HSMM consistently outperforms the HMM. This finding shows that accurate duration modelling is important in real world activity recognition as it can lead to significantly better performance. When the sensor data does not provide enough information to distinguish between activities, the duration helps in the classification. Note, that the only difference the HMM and the HSMM is that HSMMs

Table 8

Confusion Matrices for the HMM for the Bathroom1 dataset. The values are timeslices summed over all days. The activity ‘Brush’ stands for ‘Brush teeth’

	other	toilet	shower	brush	total
other	8432	196	12542	2	21172
toilet	23	233	30	10	296
shower	1	13	237	0	251
brush	2	13	6	6	27
total	8458	455	12815	18	21746

Table 9

Confusion Matrices for the CRF for the Bathroom1 dataset. The values are timeslices summed over all days. The activity ‘Brush’ stands for ‘Brush teeth’

	other	toilet	shower	brush	total
other	21128	26	17	1	21172
toilet	58	214	24	0	296
shower	67	11	173	0	251
brush	9	17	1	0	27
total	21262	268	215	1	21746

Table 10

Confusion Matrices for the HSMM for the Bathroom1 dataset. The values are timeslices summed over all days. The activity ‘Brush’ stands for ‘Brush teeth’

	other	toilet	shower	brush	total
other	20868	85	214	5	21172
toilet	28	238	17	13	296
shower	10	10	231	0	251
brush	5	13	2	7	27
total	20911	346	464	25	21746

Table 11

Confusion Matrices for the SMCRF for the Bathroom1 dataset. The values are timeslices summed over all days. The activity ‘Brush’ stands for ‘Brush teeth’

	other	toilet	shower	brush	total
other	21131	22	19	0	21172
toilet	59	216	21	0	296
shower	59	8	184	0	251
brush	9	17	1	0	27
total	21258	263	225	0	21746

model duration explicitly. The increase in performance of HSMMs therefore must be due to the its more accurate modelling of duration.

By comparing the confusion matrices of the HMM and HSMM we see that the gain in performance is mainly due to a more accurate recognition of the ‘other’ activity. Although, we have only included the confusion matrices for the ‘Bathroom1’ dataset, the performance increase in the other datasets is also caused to a large extend because of the more accurate recognition of the ‘other’ activity. Interestingly enough, the ‘other’ activity is the most difficult activity to model in terms of duration. The ‘other’ activity is not one type of activity, but rather a collection of activities such as ‘being idle’. Therefore, its duration is not likely to take the form of a Gaussian distribution. The reason for the increase in recognition of the ‘other’ activity is therefore caused not because of the more accurate duration modelling of the ‘other’ activity, but rather because of the more accurate duration modelling of the remaining activities. Because certain durations are very unlikely for activities such as ‘toiletting’ and ‘showering’, the ‘other activity’ is the only activity left because it can practically have any duration.

We also saw that the performance difference between SMCRFs and CRFs was much smaller. This result tells us that CRFs are able to learn a good discriminative model of durations in this scenario, and the explicit duration modelling done in SMCRFs does not result in a significant performance improvement. As we discussed in Section 5.1, the parameters of a CRF are learned by optimizing the posterior probability instead of the joint probability. It is therefore not limited to the maximum likelihood model of the data. If there exists a set of parameters that better discriminate the provided training data it will use that set. It has been shown in previous work that CRFs are more robust in dealing with violations of the modelling assumptions, but that this also holds for an inherent property such as duration modelling is a novel contribution and an interesting find.

It should be noted that the flexibility of discriminative models such as CRFs in dealing with violations of the modelling assumptions comes at a price. First, discriminative models take much longer to train than their generative counterpart. Second, discriminative models are more prone to overfitting. This second problem can be clearly seen from the confusion matrices. Both CRFs and SMCRFs completely fail to model the ‘brush’ activity, because there are very few examples of that activity in the training data. In general the problem of overfitting is more likely to occur in unbalanced datasets, that is, datasets in which one or more classes occur much more frequently than the other classes. During learning in discriminative models each class gets weighted according to the available number of datapoints in the training data. Frequent classes therefore get a bigger weight than infrequent classes.

Whether the improved recognition performance of CRFs is worth the extra computational cost depends on the application. Modelling the data more accurately using the HSMM allows both speedy learning and good performance, and is less prone to overfitting. It does, however, result in slower inference and is dependent on correct modelling assumptions for the durations.

We did not look at hierarchical models in this work. Hierarchical models can model a state using a number of substates in which the duration of the state is a convolution of geometric distributions [18]. Although this allows better duration modelling than conventional models, it also requires a lot more parameters (proportional to the number of substates). It has been shown that semi-Markov models can significantly outperform hierarchical models, despite their better duration modelling [5].

The choice of using the Gaussian distribution for modelling durations in semi-Markov models was primarily made because it is easily implemented in SMCRFs using feature functions. We have experimented with other duration distributions such as the gamma and Poisson distributions when using the HSMM and this resulted in similar performance as using the Gaussian distribution. Previous work has shown that the Coxian distribution can perform equally well as other distributions, but can save significant computational cost [5]. This would be an interesting alternative when speedy inference is needed.

In terms of future work, further extensions of semi-Markov models are possible. In this work we have restricted ourselves to accurately modelling duration. However, because in semi-Markov models states are represented as segments instead of time slices, it is possible to include dependencies that are impossible to model with HMMs and CRFs. For example, using higher-order Markov models would in the case of HMMs and CRFs mean dependency on the previous timeslices, while in the case of semi-Markov models it means dependency on the previous segments, which is most likely much more informative. Another example is to use a more complicated observation model. Because observations are now modelled for each segment, we could incorporate features that describe that a sensor should fire at least once for the duration of the activity.

7. Conclusions

In this work we have provided a thorough comparison of the performance of HMMs, CRFs, HSMMs and SMCRFs on real world activity recognition data. Our experiments on both novel and publicly available real world datasets show that modelling duration can lead to significantly better performance in activity recognition. The novel dataset will be made publicly available. HSMMs, CRFs and SMCRFs all significantly outperform HMMs. We have shown that, for CRFs, this is caused by their ability to deal with violations of the modelling assumptions, including assumptions on duration modelling. Furthermore, we have shown that SMCRFs, despite their greater expressive power, do not result in significant performance increase. What model to use will depend on practical considerations, such as the availability of labelled data, the importance of training time and the importance of speedy inference.

Appendix

A. Inference in semi-Markov models

A.1. Viterbi for hidden semi-Markov model

We use our segment notation to represent the state sequence for HSMMs. We wish to find the sequence of segments that maximizes $p(\mathbf{s}_{1:U}, \mathbf{x}_{1:T} \mid \theta)$. Because the duration of the segments is not known for a novel sequence of observations, we have to determine which duration gives the highest probability. Therefore, at each timestep we iterate over all possible durations, from 1 to a predefined maximum D, and store the probability in the following variable

$$\tau_{t,d}(i) = \max_{s_1,\dots,s_{k-1}} p(\mathbf{x}_{1:t}, s_1, \dots, s_k = (t-d+1, d, i) \mid \theta).$$

This represents the highest probability of a sequence of segments where the final segment started at $t-d+1$, has a duration of d and a label i. Just as in the original Viterbi algorithm for the HMM it is sufficient to keep track of the maximum probability of ending up in state s_{k-1} to compute the maximum probability of ending up in s_k. The state label of this previous segment is stored in array $\zeta_t(d, i)$.

Once $\tau_{t,d}$ has been calculated for every duration, we can determine which duration gives the maximum probability. We define

$$\delta_t(i) = \max_{s_1,\dots,s_{k-1}} p(\mathbf{x}_{1:t}, s_1, \dots, s_k = (t - d^* + 1, d^*, i) \mid \theta)$$

in which d^* is the duration with the highest probability at time t for state i. We keep track of the best duration and the label of the previous segment that led to this maximum using the variables $\upsilon_t(i)$ and $\psi_t(i)$, respectively. The complete procedure for finding the best sequence of segments can now be stated as follows:

1. Initialization: The probability of the label of the first segment is given by the initial state distribution π.

$$\tau_{d,d}(i) = \pi_i p_i(d) \prod_{t=1}^{d} p(\vec{x}_t \mid y_t = i)$$
$$\zeta_d(d, i) = 0$$

2. Recursion: At each timeslice first iterate over all possible durations.

$$\tau_{t,d}(j) = \max_{1\leqslant i\leqslant Q} [\delta_{t-d}(i)a_{ij}]\, p_j(d) \prod_{m=t-d+1}^{t} p(\vec{x}_m \mid y_m = j)$$
$$\zeta_t(d, j) = \operatorname*{argmax}_{1\leqslant i\leqslant Q} [\delta_{t-d}(i)a_{ij}]$$

Then determine which duration gives the highest probability.

$$\delta_t(i) = \max_{1\leqslant d\leqslant D} \tau_{t,d}(i)$$
$$\upsilon_t(i) = \operatorname*{argmax}_{1\leqslant d\leqslant D} \tau_{t,d}(i)$$
$$\psi_t(i) = \zeta_t(\upsilon_t(i), i)$$

3. Termination: Determine which state has the highest probability in the final timeslice.

$$P^* = \max_{1\leqslant i\leqslant Q} [\delta_T(i)]$$
$$y_T^* = \operatorname*{argmax}_{1\leqslant i\leqslant Q} [\delta_T(i)]$$
$$t = T$$
$$u = 0$$

4. Sequence backtracking: Backtrack from there, looking up the duration and previous state that were stored earlier.

$$d_t^* = \upsilon_t(y_t^*)$$
$$s_u^* = (t - d_t^* + 1, d_t^*, y_t^*)$$
$$t = t - d_t^*$$
$$u = u - 1$$
$$y_t^* = \psi_{t+d}(y_{t+d}^*)$$

It should be noted that a negative index is used for the segments because the number of segments is not known beforehand. This is easily corrected afterwards by adding $|s^*|$ to all indices.

A.2. Viterbi for semi-Markov conditional random field

1. Initialization:

$$\tau_{d,d}(i) = \phi_d(i, 0, \mathbf{x}_{1:d}, d)$$
$$\zeta_d(d, i) = 0$$

2. Recursion: At each timeslice first iterate over all possible durations.

$$\tau_{t,d}(j) = \max_{1\leqslant i\leqslant Q} [\delta_{t-d}(i)\phi_t(j, i, \mathbf{x}_{t-d+1:t}, d)]$$
$$\zeta_t(d, j) = \operatorname*{argmax}_{1\leqslant i\leqslant Q} [\delta_{t-d}(i)\phi_t(j, i, \mathbf{x}_{t-d+1:t}, d)]$$

Then determine which duration gives the highest probability.

$$\delta_t(i) = \max_{1\leqslant d\leqslant D} \tau_{t,d}(i)$$
$$\upsilon_t(i) = \operatorname*{argmax}_{1\leqslant d\leqslant D} \tau_{t,d}(i)$$
$$\psi_t(i) = \zeta_t(\upsilon_t(i), i)$$

3. Termination: Determine which state has the highest probability in the final timeslice.

$$P^* = \max_{1\leqslant i\leqslant Q} [\delta_T(i)]$$
$$y_T^* = \operatorname*{argmax}_{1\leqslant i\leqslant Q} [\delta_T(i)]$$
$$t = T$$
$$u = 0$$

4. Sequence backtracking: Backtrack from there, looking up the duration and previous state that

were stored earlier.

$$d_t^* = \upsilon_t(y_t^*)$$

$$s_u^* = (t - d_t^* + 1, d_t^*, y_t^*)$$

$$t = t - d_t^*$$

$$u = u - 1$$

$$y_t^* = \psi_{t+d}(y_{t+d}^*)$$

We can use a slightly modified form of this algorithm to compute the normalization constant efficiently, by replacing all the $\max$ operations in the above algorithm by summations. The value of $Z(\mathbf{x}_{1:T})$ is then calculated by summing over $\delta_t(i)$ in the last timeslice.

$$Z(\mathbf{x}_{1:T}) = \sum_{i=1}^{Q} \delta_T(i)$$

References

[1] G. Abowd, A. Bobick, I. Essa, E. Mynatt, and W. Rogers. The aware home: Developing technologies for successful aging. In *Proceedings of AAAI Workshop and Automation as a Care Giver*, 2002.

[2] J.C. Augusto and C.D. Nugent, editors. *Designing Smart Homes, The Role of Artificial Intelligence*, volume 4008 of *Lecture Notes in Computer Science*. Springer, 2006.

[3] C.M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics).* Springer, August 2006.

[4] D.J. Cook and S.K. Das. *Smart Environments: Technology, Protocols and Applications*. Wiley-Interscience, 2004.

[5] T. Duong, D. Phung, H. Bui, and S. Venkatesh. Efficient duration and hierarchical modeling for human activity recognition. *Artif. Intell.*, 173(7-8):830–856, 2009.

[6] J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. 18th International Conf. on Machine Learning*, pages 282–289. Morgan Kaufmann, San Francisco, CA, 2001.

[7] D.C. Liu and J. Nocedal. On the limited memory bfgs method for large scale optimization. *Math. Program.*, 45(3):503–528, 1989.

[8] B. Logan, J. Healey, M. Philipose, E. Munguia-Tapia, and S. Intille. A long-term evaluation of sensing modalities for activity recognition. In *Ubicomp*, 2007.

[9] E. Marhasev, M. Hadad, and G.A. Kaminka. Non-stationary hidden semi Markov models in activity recognition. In *MOO*, 2006.

[10] K.P. Murphy. Hidden semi-Markov models (hsmms). Technical report, University of British Columbia, 2002.

[11] P. Natarajan and R. Nevatia. Coupled hidden semi Markov models for activity recognition. In *WMVC '07: Proceedings of the IEEE Workshop on Motion and Video Computing*, page 10, Washington, DC, USA, 2007. IEEE Computer Society.

[12] N. Oliver, A. Garg, and E. Horvitz. Layered representations for learning and inferring office activity from multiple sensory channels. *Comput. Vis. Image Underst.*, 96(2):163–180, 2004.

[13] D.L. Vail. Conditional Random Fields for Activity Recognition. PhD Thesis, Carnegie Mellon University, 2008.

[14] D.J. Patterson, D. Fox, H.A. Kautz, and M. Philipose. Fine-grained activity recognition by aggregating abstract object usage. In *ISWC*, pages 44–51. IEEE Computer Society, 2005.

[15] L.R. Rabiner. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[16] S. Sarawagi and W.W. Cohen. Semi-Markov conditional random fields for information extraction. In *In Advances in Neural Information Processing Systems 17*, pages 1185–1192, 2004.

[17] F. Sha and F. Pereira. Shallow parsing with conditional random fields. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 134–141, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

[18] A. Subramanya, A. Raj, J. Bilmes, and D. Fox. Hierarchical models for activity recognition. In *IEEE Multimedia Signal Processing (MMSP) Conference*, Victoria, CA, October 2006.

[19] C. Sutton and A. McCallum. *Introduction to Statistical Relational Learning*, chapter 1: An introduction to Conditional Random Fields for Relational Learning, page (Available online). MIT Press, 2006.

[20] R. Suzuki, M. Ogawa, S. Otake, T. Izutsu, Y. Tobimatsu, S.-I. Izumi, and T. Iwaya. Analysis of activities of daily living in elderly people living alone. *Telemedicine*, 10:260, 2004.

[21] E.M. Tapia, S. Intille, L. Lopez, and K. Larson. The design of a portable kit of wireless sensors for naturalistic data collection. In *PERVASIVE*, 2006.

[22] E.M. Tapia, S.S. Intille, and K. Larson. Activity recognition in the home using simple and ubiquitous sensors. In *Pervasive Computing, Second International Conference, PERVASIVE 2004*, pages 158–175, Vienna, Austria, April 2004.

[23] T.T. Truyen, H.H. Bui, and S. Venkatesh. Human activity learning and segmentation using partially hidden discriminative models. In *HAREM*, 2005.

[24] T.T. Truyen, D.Q. Phung, H.H. Bui, and S. Venkatesh. Hierarchical semi-Markov conditional random fields for recursive sequential data. In *Neural Information Processing Systems (NIPS)*, 2008.

[25] D.L. Vail, M.M. Veloso, and J.D. Lafferty. Conditional random fields for activity recognition. In *International Conference on Autonomous Agents and Multi-agent Systems (AAMAS)*, 2007.

[26] T. van Kasteren, A. Noulas, G. Englebienne, and B. Kröse. Accurate activity recognition in a home setting. In *UbiComp '08: Proceedings of the 10th international conference on Ubiquitous computing*, pages 1–9, New York, NY, USA, 2008. ACM.

[27] H. Wallach. Efficient training of conditional random fields. Master's thesis, University of Edinburgh, 2002.

[28] D. Wilson and C. Atkeson. Simultaneous tracking and activity recognition (star) using many anonymous binary sensors. In *Pervasive Computing, Third International Conference, PERVASIVE 2005*, pages 62–79, Munich , Germany, 2005.

[29] D.H. Wilson. *Assistive Intelligent Environments for Automatic Health Monitoring*. PhD thesis, Carnegie Mellon University, 2005.