Semi-supervised and Active Training of Conditional Random Fields for Activity Recognition

by

Maryam Mahdaviani

B.Sc., The University of British Columbia, 2005

A THESIS SUBMITTED IN PARTIAL FULFILMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Master of Science

in

The Faculty of Graduate Studies

(Computer Science)

The University Of British Columbia

Abstract

Automated human activity recognition has attracted increasing attention in the past decade. However, the application of machine learning and probabilistic methods for activity recognition problems has been studied only in the past couple of years. For the first time, this thesis explores the application of semi-supervised and active learning in activity recognition. We present a new and efficient semi-supervised training method for parameter estimation and feature selection in conditional random fields (CRFs), a probabilistic graphical model. In real-world applications such as activity recognition, unlabeled sensor traces are relatively easy to obtain whereas labeled examples are expensive and tedious to collect. Furthermore, the ability to automatically select a small subset of discriminatory features from a large pool can be advantageous in terms of computational speed as well as accuracy. We introduce the semi-supervised virtual evidence boosting (sVEB) algorithm for training CRFs – a semi-supervised extension to the recently developed virtual evidence boosting (VEB) method for feature selection and parameter learning. sVEB takes advantage of the unlabeled data via minimum entropy regularization. The objective function combines the unlabeled conditional entropy with labeled conditional pseudo-likelihood. The sVEB algorithm reduces the overall system cost as well as the human labeling cost required during training, which are both important considerations in building real world inference systems. Moreover, we propose an active learning algorithm for training CRFs is based on virtual evidence boosting and uses entropy measures. Active virtual evidence boosting (aVEB) queries the user for most informative examples, efficiently builds up labeled training examples and incorporates unlabeled data as in sVEB. aVEB not only reduces computational complexity of training CRFs as in sVEB, but also outputs more accurate classification results for the same fraction of labeled data. In a set of experiments we illustrate that our algorithms, sVEB and aVEB, benefit from both the use of unlabeled data and automatic feature selection, and outperform other semi-supervised and active training approaches. The proposed methods could also be extended and employed for other classification problems in relational data.

Contents

List of Tables

List of Figures

Acknowledgements

First and foremost, I would like to thank my supervisor, Nando de Freitas. In the past few years, he has been my teacher, my advisor, my boss and my mentor. He has always been supportive and open to new ideas. For the completions of my thesis and the masters program, I am also indebted to two other great people. I was very fortunate to work with Tanzeem Choudhury. She constantly encouraged me to keep up the good work. Jim Little helped me to get through the hard times and kept me on track. He was also the second examiner of this thesis and I am grateful to his comments and suggestions.

During the past two years I worked and lived with a number of great people. My friends at Intel Research Seattle made my internship experience very enjoyable. My house-mates, Sarah and Liz, helped me through the rough times and put up with my late night festivities. My best friend and companion, Reza, made my life fun and stimulating. Meeting him is one of the best things that has ever happened to me.

I thank my family here for their great understanding and never-ending love. For the past seven years my older sister, Sheila, has been far from me, but I never forget her help and support throughout my life. My caring younger sister, Dorri, is the sunshine of my life; she cheers up my gloomy days. This acknowledgement is incomplete without thanking my life-time teachers, my parents. My mother, Tahereh Ghoddousi, is my best advisor. My father, Mohammad Mahdaviani, helps me believe in myself and possibilities. This thesis is dedicated to you, Mamani and Babaie! Thank you!!

This work was funded by Intel Research and National Science and Engineering Research Council of Canada. I sincerely thank them for selecting and supporting me.

MARYAM MAHDAVIANI

Strange, is it not? that of the myriads who
Before us pass'd the door of Darkness through
Not one returns to tell us of the Road,
Which to discover we must travel too.

–Omar Khayyám (1048-1122)

Chapter 1

Introduction

Human beings are trained to recognize other people's activities. Whether at home or in the work place, everyday we are constantly inferring others' behaviours and intentions. At the office you need to know if your coworker is on a meeting, having lunch or talking on the phone. At home, you would like to know if other family members are watching TV, cooking or taking a nap. Without understanding and reasoning about other people's activities we cannot effectively communicate and interact in a community.

As understanding others' activities is essential for our intelligent communications, automated activity recognition is needed for any effective human-computer interaction. In order to give intelligent suggestions to the user we need to understand her behaviours and intentions. A key to understanding the user's intentions is the ability to infer her activities. With the rise of ubiquitous computing, and having computers in every corner of our lives, intelligent systems care about human activities beyond mouse clicks and key board usage. An intelligent ubiquitous home needs to know if the user is cooking, eating or sleeping. An automated trainer needs to know if the athlete is running, walking, bicycling or sprinting. Activity recognition could also play a role in health-care technologies; especially in supporting elder care, managing cognitive disabilities, and monitoring long-term health. Every year, a hefty budget is spent on nurses and health care facilities that are employed to monitor small groups of the elderly and handicapped. Mental patients also need a great deal of attention. Some of them are monitored by family members or nurses. Some are asked to keep record of their daily activities and report them to their therapists. These diaries are often inaccurate and biased mostly because of their behaviourial problems. Automating human monitoring systems could radically reduce these costs and in many cases improve the performance. Activity recognition systems could also have extensive applications in surveillance and military missions. For all these reasons, an automated reasoning system for inferring and understanding human behaviours has become one of the goals of Artificial Intelligence.

Activity recognition might seem easy and innate to human beings. However, it consists of complicated tasks of sensing, learning and inferring. There is much room left for improving each of these tasks. In the next chapter we talk about some of the works on designing sensors and processing their outputs for activity recognition. However, the focus of this thesis is on the learning component; how we could efficiently and cost-effectively train an activity model that could operate in real-world applications.

Considering the sequential nature of human activities, temporal models are naturally adopted for effective activity recognition systems. Recent work by Liao *et al.* [22] has shown very promising results for employing Conditional Random Fields, for activity recognition. Conditional random fields (CRFs) are undirected graphical models that have been successfully applied to the classification of relational and temporal data [15]. However, training complex CRF models with large numbers of input features is slow, and exact inference is often intractable. The ability to select the most informative features as needed can reduce the training time and the risk of over-fitting of parameters. The work of Liao *et al.* addresses this issue by introducing a training algorithm that simultaneously performs feature selection and learns the model parameters.

The goal of our work is to build an activity recognition system that is not only accurate but also scalable, efficient, and easy to train and deploy. Some of the main challenges faced by current activity inference systems are the amount of human effort spent in labeling and feature engineering as well as the computational complexity and cost associated with training. Data labeling also has privacy implications because it often requires human observers or recording videos. In this thesis, we introduce a fast, efficient and scalable semi-supervised training algorithm for CRFs. By training CRFs in semi-supervised and active-learning frame-works we reduce the number of labeled training examples and take advantage of unlabeled training data. This is particularly beneficial in applications for which labeling is expensive and collecting unlabeled training data is easy such as activity recognition. Although we are driven by activity recognition applications, the methodologies developed in this thesis could be applied to other applications such as text/video processing or bio-informatics.

1.1 Related Work

To the best of our knowledge there is no work done on semi-supervised and active learning for activity recognition. However, there have been works on related topics, semi-supervised and active training of CRFs, feature-selection for CRFs and temporal learning for activity recognition:

- Feature selection for CRFs: CRFs are often dealing with domains with very large number of features. A good number of features are usually irrelevant and redundant. Several supervised techniques have been proposed for feature selection in CRFs. For discrete features, McCallum [27] suggested an efficient method for feature induction by iteratively increasing conditional log-likelihood. Dietterich [5] applied gradient tree boosting to select features in CRFs by combining boosting with parameter estimation for 1D linear-chain models. Boosted random fields (BRFs) [42] combine boosting and belief propagation for feature selection and parameter estimation for densely connected graphs that have weak pairwise connections. Recently, Liao *et al.* [22] developed a more general version of BRFs, called virtual evidence boosting (VEB) that does not make any assumptions about graph connectivity or the strength of pairwise connections. The objective function in VEB is a soft version of maximum pseudo-likelihood (MPL), where the goal is to maximize the sum of local log-likelihoods given soft evidence from its neighbors. This objective function is similar to that used in boosting, which makes it suitable for unified feature selection and parameter estimation. This approximation applies to any CRF structure and leads to a significant reduction in training complexity and time. The algorithms introduced in this thesis are extensions of VEB.

- Semi-supervised and active training of CRFs: Semi-supervised training techniques have been extensively explored in the case of generative models and naturally fit under the expectation maximization framework [31]. However, it is not straightforward how to incorporate unlabeled data in discriminative models using the traditional conditional likelihood criteria. Kernel CRFs proposed by Lafferty *et al.* [16] is a discriminative model of graph structured data. The authors propose a nearest neighbor semi-supervised training for CRFs by transforming CRF features to a kernel space. Although kernel CRFs show some promising results for synthetic data, the high computational cost of kernel methods makes kernel CRFs not a suitable candidate for

real time activity recognition system. A few semi-supervised training methods for CRFs have been proposed that introduce dependencies between nearby data points [20, 50]. More recently, Grandvalet and Bengio [9] propose a minimum entropy regularization framework for incorporating unlabeled data into the training of logistic regressions. The regularizer in the objective function is the conditional entropy of unlabeled data. Jiao *et al.* [10] extend this framework to train 1D Conditional Random Fields. The work is extended to 2D lattice structures by Lee *et al.* [17].

Our work is similar to the approach of Jiao *et al.* [10] in the sense that we are also inspired by semi-supervised logistic regression proposed by Grandvalet and Bengio [9]. However, we are using the entropy regularizer in the boosting objective function used in virtual evidence boosting [22]. In other words, our method performs feature selection and incorporates unlabeled data in a unified frame-work. As a result the semi-supervised training algorithm no longer needs an optimizer and training becomes faster. Moreover since our approach does not deal with the local minima involved in the objective function of [10] and [26], the solution becomes more stable. The experimental results in chapter 4 demonstrate these advantages.

To the best of our knowledge there are only two works discussing active learning for training CRFs [12, 13]. Kim and Song [12] propose an entropy-based measure for a confidence score by which the system selects the most informative samples and prompts for their labels. Kristjannson *et al.* also propose the use of a confidence measure for finding the most informative sequence [13]. But they use the normalization constant in constrained CRFs as a score to select these instances. Similar to the work of Kim and Song [12], we use an entropy-based measure as the selection criteria. However, we compute the entropy based on virtual evidence, in a semi-supervised virtual evidence boosting framework [25].

- Temporal learning for activity recognition: The use of learning algorithms for activity recognition problems has attracted some attention in the past few years. Choudhury *et al.* and Lester *et al.* propose an activity recognition system that uses boosting for feature selection and then uses HMMs for smoothing [4, 18]. Liao *et al.* propose us-

ing CRFs for activity recognition [22]. They introduce virtual evidence boosting, an algorithm that performs feature selection and learns CRF model parameters simultaneously. In chapter 3 we talk about VEB in details. We should note here that in this thesis we extend the supervised method of Liao *et al.* to semi-supervised and active-learning frame-works.

1.2 Contributions

The goal of this thesis is to develop efficient training probabilistic reasoning techniques for conditional random fields that are applied in activity recognition systems. We believe that these techniques could be of much use in other application domains. We can summarize the contributions of this thesis in the following way:

(1) Comparing the performance of supervised CRF training algorithms for activity recognition tasks. We provide extensive results for two real-world activity datasets.

(2) Semi-supervised virtual evidence boosting (sVEB) – an efficient technique for simultaneous feature selection and semi-supervised training of CRFs, which to the best of our knowledge is the first method of its kind.

(3) Experimental results that demonstrate the strength of sVEB, which consistently outperforms other training techniques on synthetic data and real-world activity classification tasks.

(4) Analysis of the time and complexity requirements of our algorithm, and comparison with other existing techniques that highlight the significant computational advantages of our approach.

(5) Active virtual evidence boosting (aVEB), an active learning way of training of CRFs. By using active learning, we are able to query the user for the most informative instances and minimize labeling effort. aVEB is an extension of sVEB. It uses an entropy-based score to find the most informative instance. After acquiring the label, the algorithm uses both labeled and unlabeled data in the training. To evaluate aVEB, we provide experimental results for its performance on real-world activity data.

1.3 Notational Remarks

The following is a table of parameters and abbreviations that we use in this thesis.

ML	maximum likelihood
MAP	maximum a posteriori
MPL	maximum pseudo likelihood
BRF	boosted random fields
MAP+Boost	maximum a posteriori with boosting
sMAP	semi-supervised maximum a posteriori
sMAP+Boost	semi-supervised MAP with boosting
VEB	virtual evidence boosting
sVEB	semi-supervised virtual evidence boosting
aVEB	active virtual evidence boosting
$\mathbf{x}$	observations
$\mathbf{y}$	labels sequence
M	number of labeled and unlabeled training examples
N	number of labeled training examples
θ	weight parameter
θ_k	weight parameter for clique k
$\mathbf{C}$	set of cliques in the graph
f_k	feature k, corresponding to clique k
n_{ki}	number of counts for feature k in data instance i
c_u	number of unlabeled training sequences
w	boosting working response
z	boosting weight
T	number of boosting iterations
t	boosting iteration
F	ensemble of weak learners
L_F	likelihood function with weak learners of F
s	number of states
ve	virtual evidence
α	the tuning parameter used in sMAP and sVEB
σ	the variance of gaussian prior used in the MAP and sMAP regularizes

D	dimension of observations
D_b	dimension of observations after feature selection
g	first derivative of L_F w.r.t. f
H	second derivative of L_F w.r.t. f
q	a CRF subsequence
q^u	an unlabeled CRF subsequence
q^l	a labeled CRF subsequence
U	number of unlabeled training subsequences
L	number of labeled training subsequences
J	number of active learning querys each time

1.4 Outline

In chapter 2 we will give an overview of activity recognition and describe how the activity datasets of this thesis were collected. Chapter 3 will be about different supervised techniques for training CRFs, maximum likelihood, maximum a posteriori, maximum pseudo likelihood and virtual evidence boosting. In Chapter 4 we will talk about semi-supervised training methods for CRFs. We will give an overview of recently developed semi-supervised maximum a posteriori and will introduce semi-supervised virtual evidence boosting. Chapter 5 will be on training CRFs in active-learning frame-work. At the end of chapters 3-5 there are experiments sections where we evaluate different algorithms on activity and synthetic datasets. In experimental sections of chapter 4 and 5 we also provide a complexity analysis of supervised and semi-supervised techniques. At the end we will conclude and discuss future directions.

Chapter 2

Activity Recognition

In this chapter we provide a general overview of activity recognition and a specific description of the set of problems we are interested in. Although the main contribution of this thesis is not about the design of activity recognition systems, it is driven and inspired by the challenges involved in this application domain.

The work of this thesis started at Intel Research Seattle as part of a large multi-disciplinary project on human activity recognition. The activity datasets were collected by Intel researchers and students at the university of Washington. In this chapter we talk about their work at the Seattle lab and how we use the collected data in our experiments.

2.1 An Overview to Activity Recognition

One of the goals of Artificial Intelligence is automated recognition of human behaviours. Activity recognition systems can be employed in numerous applications, including:

- smart human-computer interactions such as in user interfaces and ubiquitous computing [19],
- supervising the disabled and elderly and assisting them through their daily activities [4],
- monitoring patients for diagnoses and their progress in recovery [47],
- surveillance applications [29],
- better understanding human behaviours and interactions [2].

We distinguish 3 different components in a complete activity recognition system:

(a) Sensing: Sensors can range from complex devices such as cameras to simpler ones such as accelerometers. There is an ongoing area of research to design and integrate accurate, small and portable/wearable sensors for activity recognition [23, 24, 32]. Moreover, the research has shown that for different application domains different sensors might be useful. For example, video might be a great choice for surveillance applications, but is impractical in applications that users' privacy matters. In this thesis we do not talk about the sensing component nor we talk about wearable sensors. We limit ourselves to briefly describing the specific sensors that are used for collecting our two activity datasets.

(b) Processing sensor traces: engineering the output of sensors and deriving informative data features is the next step. Based on the nature of data we should use different preprocessing tools. For examples for video streams we use different visual cues whereas for audio data using frequency coefficients is common.

(c) Building a trainable computational model for inferring activities: once we have the data features, we need to learn the activity model. the contributions of this thesis is about this step of the work. Having collected two activity datasets using state-of-the-art wearable sensors and preprocessing techniques, we investigate a number of machine learning methods to train a suitable computational model. At the inference step we compare the performance of these training methods based on their classification accuracy. In the rest of this section we will talk about step (a) and (b) of our activity recognition system. In the following chapters, however, we will talk about step (c).

2.2 Activity Recognition Datasets

In this section we describe how human activity recognition data has been collected at Intel Research. We then talk about two specific datasets that have been used for our experiments in this thesis.

Since it is not clear what sensor modalities are useful for human activity inference, we would like to gather data simultaneously from a large set of sensors. As a result at Intel Research Seattle a multi-modal sensor board (MSB) has been implemented. The MSB is a 2.8 cm by 5.8 cm kit that is designed to attach to the Intel Mote (iMote), a Bluetooth/32-bit ARM7-based wireless sensor node. It can also communicate with handheld, desktop

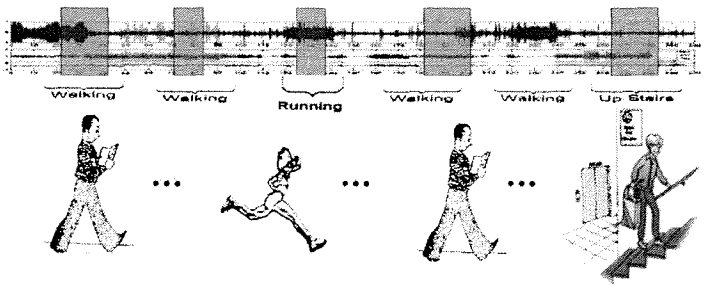

Figure 2.1: The figure illustrates a subset of sensor traces corresponding to different activities. In a semi-supervised setting, the algorithm takes advantage of both labeled (shaded) and unlabeled data.

computers, and cell phones via Bluetooth, serial, or USB. The iMote allows us to send the sensing data of MSB to any commodity Bluetooth device like a cell phone and our other interfaces allow us to connect to handhelds, laptops, and other devices [18].

The sensor board contains seven different sensors: Electric Microphone, Visible Light Phototransistor, 3-Axis Digital Accelerometer, 2-Axis Digital Compass, Digital Barometer/Temperature, Digital Ambient(IR and Visible+IR) Light and Digital Humidity/Temperature. The MSB is capable of sampling the data all simultaneously at fairly high sampling rates [18]. These sensors were selected for their general usefulness based on the related work in activity inference. Moreover, small footprint, low power consumption, and availability of digital interfaces were other main reasons. Since the MSB is small and light (the complete package with iMote and battery board is only 25g) one can comfortably wear it for long time periods. By using the MSB we collect approximately 18,000 samples of data per second. However, we do not use the samples directly for classification since in that case we will end up having an extremely high dimensional feature space. Instead we summarize the data by computing data features, bringing out the important details of the data and reducing the dimensionality.

For the current system the outputs of different sensors are processed to compute data properties, including linear and logscale FFT frequency coefficients, cepstral coefficients, spectral entropy, band-pass filter coefficients, correlations, integrals, means, and variances [18], giving us a high dimensional feature vector. For a more detailed description of the system we refer the reader to further references such as [18]. Intuitively, some features are more useful for particular activities. For example, FFT coefficients of acceleration will likely capture walking manner and as a result they are likely to be useful for distinguishing between walking/jogging, walking/standing and so forth. However, for some activities, such as riding an elevator, it might not be clear which data features should be used. Including all features in the classification process might solve the feature selection problem, but we usually do not have enough training data to reliably learn the model parameters for such a high dimensional domain. Moreover, using some irrelevant features might confuse the classifier to discriminate between various activities. The challenge of feature selection in high dimensional domains is known to the Machine Learning community and a number of theoretical and application-based solutions have been proposed for this problem. However, very little work has been done for feature selection in human activity inference. Most recently some solutions have been proposed in [4, 10, 18]. We will elaborate on them in next chapters.

To train and test our human activity recognition model we collected two datasets with the system we described above. There are three main differences between these datasets:

- Type of activities: one of the datasets consists of low-level activities such as *walking* and *riding elevator*. The other dataset contains higher level activities such as *meeting* and *having meal*; they are more abstract and often contain sub-activities. For example *meeting* and *having meal* usually also contain the basic *sitting* and *talking* activities.

- Durations: due to the nature of these activities, the activity durations are different for these datasets. For example *cooking* usually takes longer than *brushing teeth*. As a result the features for each dataset are computed for different time frame windows.

- Number of participants: a group of 7 people collected data for lower level activities but only one person collected data for the other dataset. For the first dataset we are able to train on a group of people and test on a different group. Whereas, for the second dataset we train on a

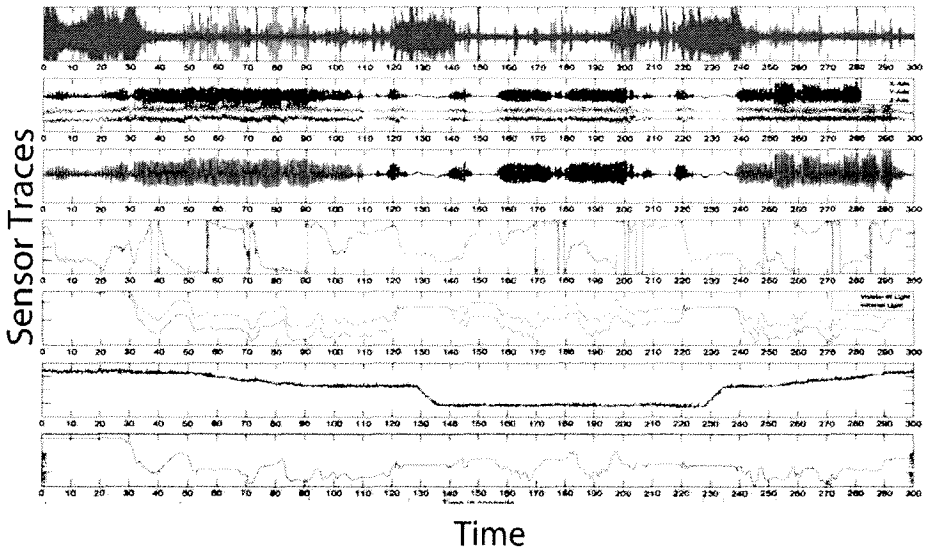

Figure 2.2: An example of sensor traces for Audio, Acceleration, Compass, Light and Pressure (from top to bottom). The horizontal line shows time in seconds. In the preprocessing step linear and logscale FFT frequency coefficients, cepstral coefficients, spectral entropy, band-pass filter coefficients, correlations, integrals, means, and variances are computed, giving us a 651 dimensional feature vector trace

set of days and test on different days. This way we are verifying our system on two different ways of training which are similar to the real world settings.

In the next section we describe the activity dataset in more detail.

2.3 Low-level and High-level Human Activity Datasets

For the low-level activity dataset, a group of 7 people collected the data while doing 8 basic physical activities. Two-thirds of the data was collected from a computer science building and the other third was collected in an office building. Volunteers collected data wearing three MSBs: one on a shoulder

strap, one on the side of their waist, and one on their right wrist. As a result we have three sets of the data corresponding to each location. The volunteers were asked to perform a specific sequence of activities, like sitting on a couch for a few minutes before walking upstairs to brush their teeth. The activities consist of 8 basic activities: *walking*, *standing*, *walking up stairs*, *walking down stairs*, *riding elevator down*, *riding elevator up* and *Brushing teeth*. A total of 12 hours of data has been recorded over all participants. The duration of activities ranges from $14 - 143$ minutes giving the average of 40 minutes for each activity.

For the high-level dataset, one person collected audio, acceleration, and light sensor data as he stayed indoors using a small wearable device. The total length of the dataset is about $1,100$ minutes, recorded over a period of 12 days. The goal is to recognize the person's major indoor activities including *computer usage*, *having meal*, *meeting*, *TV watching* and *sleeping*. We segmented the data into one-minute chunks and manually labeled the activity at each minute. For each chunk of data, we computed 315 feature values. A detailed numerical description of the activity datasets is presented in tables (2.1) and (2.2). Labeling activity data is a time-consuming and boring task. While collecting the data at the Intel lab, we also captured the ground truth video. After finishing each activity scenario, a person watched the video and labeled each segment. Figure (2.3) shows the labeled video streams for three tasks of *walking*, *walking up* and *sitting*.

2.4 Summary

In this chapter we gave an overview to activity recognition and motivated the reader about our application domain. We talked about the multi-sensor board (MSB) that is developed at Intel Research Seattle and described how we have collected and labeled our activity datasets. We do not talk about the application domain of this thesis, activity recognition, any further. In the next few chapters, we discuss supervised, semi-supervised and active training of conditional random fields. In particular we will focus on chain CRFs since they are suitable for our application. Although our experimental results show that these methods are helpful for the task of activity recognition we believe that they could be also applicable in many other application domains such as text processing and computational vision.

Activity	*P1*	*P2*	*P3*	*P4*	*P5*	*P6*	*P7*	*Total*
Sitting	0	1439	2015	1620	3820	2271	2622	13787
Standing	838	688	2285	441	892	1883	921	7948
Walking	1636	2674	2122	1898	2466	3273	3173	17242
Walking up stairs	0	61	531	564	579	522	459	2716
Walking down stairs	321	422	344	387	437	479	379	2679
Riding elevator up	354	73	308	238	93	175	110	1351
Riding elevator down	0	117	286	206	202	287	130	1228
Brushing teeth	0	0	573	577	277	697	538	2662
Total	3149	5474	8464	5931	8766	9587	8332	49703

Table 2.1: The per-class count of collected activity data instances for dataset 1 (multi-person). This table shows the amount of collected data for each activity per person. There is a total of 49703 data points which is about 3.5 hours.

Computer usage	*Having meal*	*Meeting*	*Watching TV*	*Sleeping*	*Total*
241	135	148	136	247	907

Table 2.2: This table shows the details of dataset 2. This dataset is collected by one person during few days. Compared to dataset 1, it is smaller and has longer duration activities. The total duration of this dataset is about 15.1 hours.

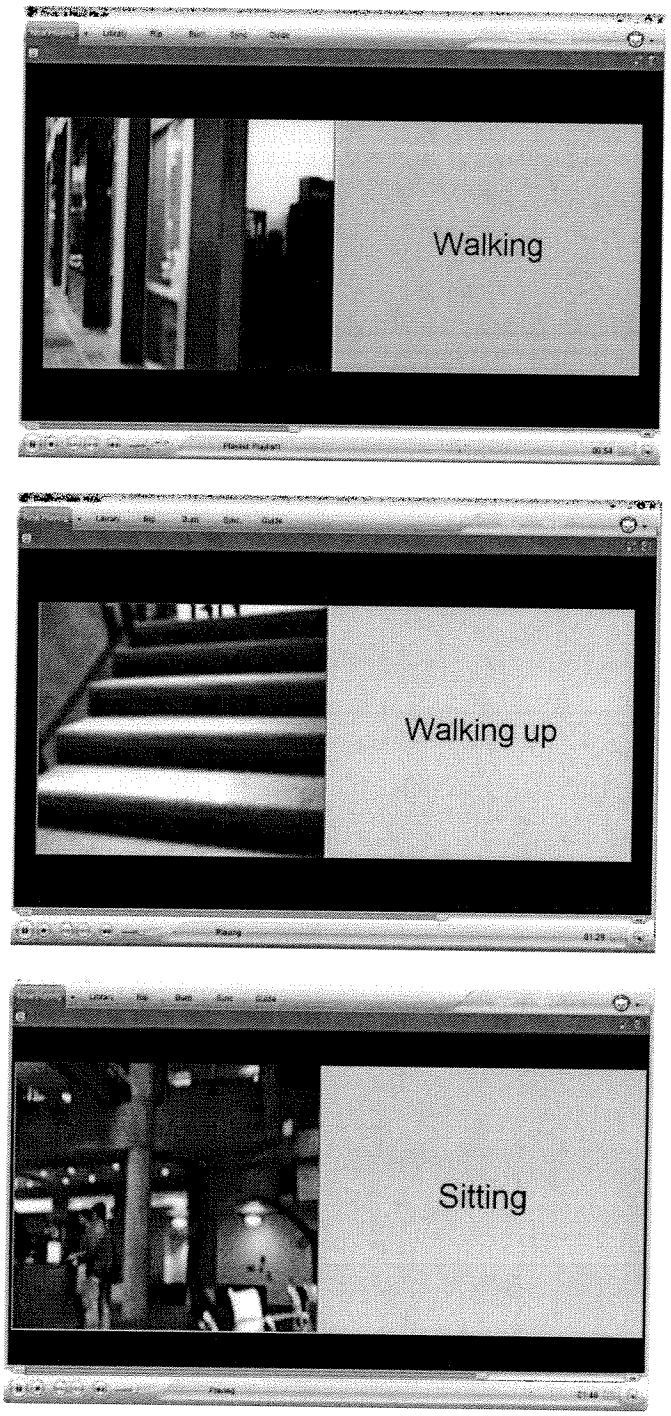

Figure 2.3: An illustration of labeling activity data based on the ground truth video. In each picture the left frame show the video stream and the right frame shows the human labeled activity name.

Chapter 3

Supervised Training of Conditional Random Fields

Conditional Random Fields(CRFs) are undirected graphical models, extensively used for classification of structured, relational data [14, 15, 36, 39]. CRFs have shown to be successful for classification problems in different application domains such as natural language processing [15, 39, 41], information extraction [13, 28, 35] and computer vision [14, 40, 45]. Similar to Hidden Markov Model(HMM) [37] nodes in CRFs represent hidden states and observations. In this thesis we denote observations by $\mathbf{x}$ and hidden states by $\mathbf{y}$.

Employing CRFs involves two steps:

- learning the model and
- inference based on observations and the trained model.

There are a number of methods proposed for inference in undirected graphical models and particularly CRFs [48]. In this thesis we do not talk about inference methods, but we will talk about the learning approaches. For a fair comparison we use same inference method, Belief Propagation [33], for all the training methods proposed in in the current and next chapters. Based on the connectivity structure of the graph, CRFs define the probability of labels given the observations, $p(\mathbf{y}|\mathbf{x})$. In a simple activity recognition problem based on time, location and duration (demonstrated in figure (3)), the coloured nodes represent 4 different type of observations and the blank nodes represent the labels, hidden states or in this example the types of activities [21]. In general CRFs could have any structure and they are not limited to chains. Due to the sequential nature of our activity datasets, in this thesis we will focus on chain CRFs. However, the proposed methods could be easily extended for other structures.

CRFs directly represent the conditional probability of hidden states given the observations. Unlike HMMs, they do not make any assumption on the

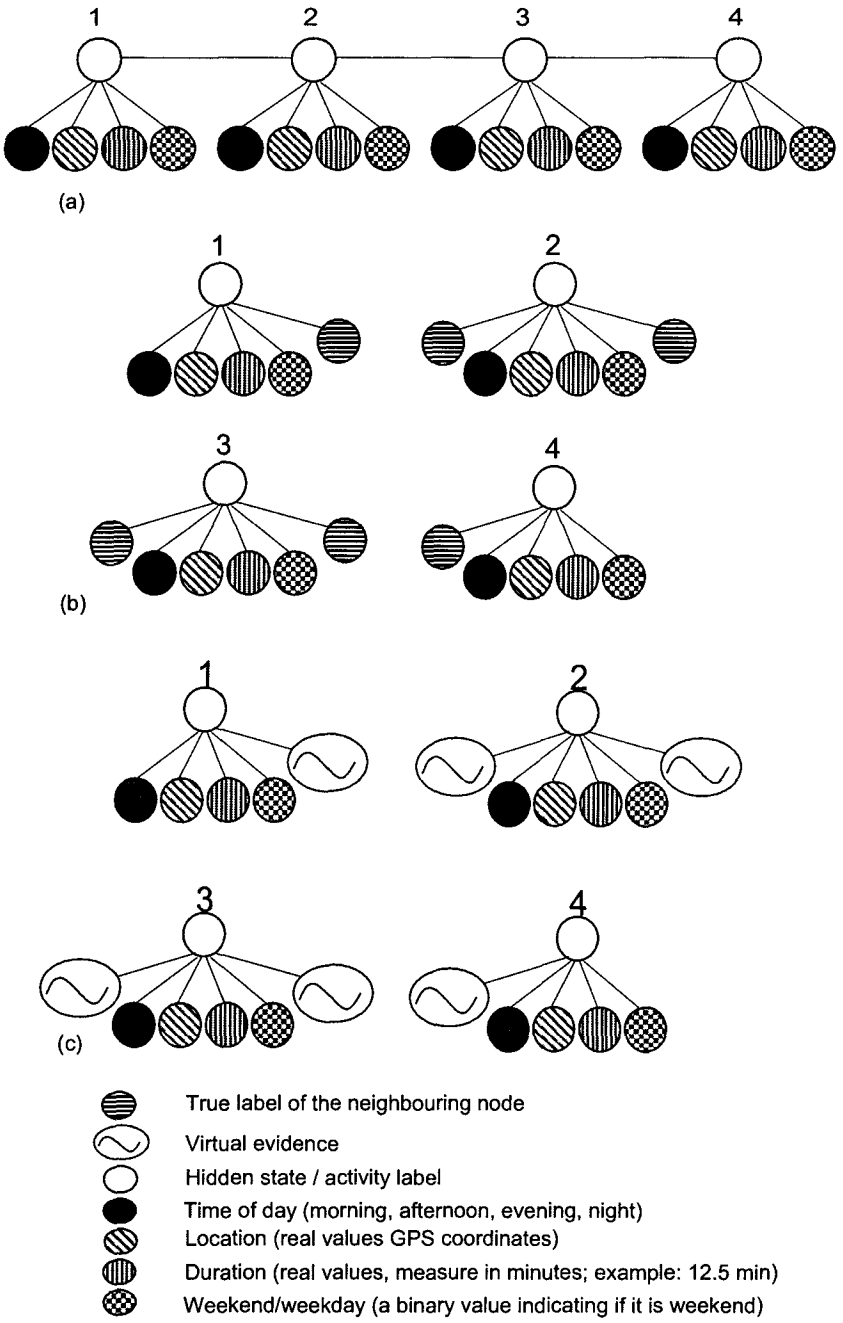

Figure 3.1: Examples of training CRFs in maximum likelihood(a), maximum pseudo likelihood(b) and virtual evidence boosting(c) frameworks.

dependency structure between the observations. This property makes CRFs very suitable for classification tasks in complex domains where we have many complex and overlapping observational features. Each fully connected subgraph in CRFs is called a clique. We could factorize CRFs as a set of cliques, $\mathbf{C}$, where each clique is denoted as $(\mathbf{x_k}, \mathbf{y_k})$, $\mathbf{k} \in \mathbf{C}$. In a chain CRF, each observation with its corresponding hidden state form a clique. Also every two neighboring hidden states form a clique. In figure (3), the observation *time of the day* with the hidden state 2, and hidden state 2 with hidden state 3 are two cliques. Factorizing graphs to cliques enables us to write the corresponding probability distributions as a product of clique potentials. A Clique potential is a function that based on variable configurations returns a non-negative value. One could think of clique potentials as a value for capturing the compatibility between the variable(nodes) within the clique. For example, for our low-level activity dataset, (*meeting, computer usage*) is stronger that (*meeting, sleeping*).

In terms of clique potentials, we could write the probability distribution, $p(\mathbf{y}|\mathbf{x})$ in the following way:

$$p(\mathbf{y}|\mathbf{x}, \theta) = \prod_{\mathbf{k} \in \mathbf{C}} \phi_{\mathbf{k}}(\mathbf{x_k}, \mathbf{y_k})/\mathbf{Z}, \tag{3.1}$$

where $\mathbf{Z}$ is a normalizing constant, $\mathbf{Z} = \sum_{\mathbf{y}'} \prod_{\mathbf{k} \in \mathbf{C}} \phi_{\mathbf{k}}(\mathbf{x_k}, \mathbf{y}'_{\mathbf{k}})$. The cost of computing normalizing constant is exponential in the number of hidden states since we are summing over all possible configurations. As a result, exact computation becomes impossible for dense graphs. In this thesis we are mostly dealing with chain CRFs, for which $\mathbf{Z}$ could be computed exactly. Therefore we do not talk about approximation methods available for computing normalizing constants.
Without loss of generality, we could write each potential function as a log-linear combination of simpler functions called *feature functions*. These feature functions could be binary or real valued, designed by the user and tailored for the application. For each feature function we assign a weight parameter, θ, which captures the importance of the function:

$$\phi_{\mathbf{k}}(\mathbf{x_k}, \mathbf{y_k}) = exp\{\theta_k^T \mathbf{f_k}(\mathbf{x_k}, \mathbf{y_k})\}. \tag{3.2}$$

Knowing the structure of CRFs, the task of training is basically learning the weight parameters, θ's, of the model. Learning could be thought as solving

an optimization problem for θ's, where we are trying to maximize the joint likelihood, $p(\mathbf{x}, \mathbf{y})$.

In this chapter we first talk about three common ways of training CRFs, maximum likelihood (ML), maximum a posteriori (MAP) and maximum pseudo likelihood (MPL). We then talk about recently developed virtual evidence boosting (VEB), a supervised algorithm for simultaneous feature selection and training. The main contributions of this thesis (which are discussed in the next chapters) are built on VEB.

3.1 Maximum a Posteriori

Given labeled training data, $(\mathbf{x}, \mathbf{y})$, we could think of the probability distribution over the data, $p(\mathbf{y}|\mathbf{x}, \theta)$, as a function of θ. In most training methods for CRFs the goal is to find θ's that maximize this likelihood function. In Maximum likelihood we are solving a maximization problem in which the objective function is the log likelihood of the labeled training data:

$$L(\theta) = \log(\, p(\mathbf{y}|\mathbf{x}, \theta)\,) \tag{3.3}$$

$$= \log \frac{\exp(\sum_{k=1}^{K} \theta_k f_k(\mathbf{x}, \mathbf{y}))}{\sum_{\mathbf{y}'} \exp(\sum_{k=1}^{K} \theta_k f_k(\mathbf{x}, \mathbf{y}'))}, \tag{3.4}$$

where we have $\mathbf{K}$ cliques in the graph. Note that since *log* is a monotone function, maximizing log-likelihood is the same as maximizing likelihood and yields same values for θ's. Maximum likelihood is a popular point estimation method. However, we are often dealing with large number of parameters in CRFs. To avoid over-fitting, we incorporate a *regularizer* into our object function. A common regularizer is Gaussian prior, $\exp(\frac{\|\theta\|}{2\sigma})$ which penalizes the weight parameters whose norms are too large. When we add a regularizer to the ML objective function, we call the new training approach maximum a posteriori (MAP). More specifically a MAP objective function with a Gaussian prior is defined as following:

$$L(\theta) = \log(p\,(\mathbf{y}|\mathbf{x}, \theta)\,) - \frac{\|\theta\|}{2\sigma^2} \tag{3.5}$$

$$= \log \frac{\exp(\sum_{k=1}^{K} \theta_k f_k(\mathbf{x}, \mathbf{y}))}{\sum_{\mathbf{y}'} \exp(\sum_{k=1}^{K} \theta_k f_k(\mathbf{x}, \mathbf{y}'))} - \frac{\|\theta\|}{2\sigma^2}. \quad (3.6)$$

The regularizer term in the objective function is a Gaussian prior with mean zero and variance of σ^2 on all weight parameters. $\|\theta\|$ is the l_2 *norm* of weight parameter vector, θ. It is possible to use other regularizers for the ML objective function. One of the common alternatives is *lasso* or l_1 *norm* prior. *lasso* is known as a sparse promoting prior and in a number of works has been proposed as a feature selection method [30, 43]. Having an l_1 *norm* makes small weight parameters shrink to zero. As a result, less significant features which usually have smaller weights will be automatically discarded. *lasso* has been used for CRFs in a few different applications [34, 43]. However, it has been shown that using lasso instead of Gaussian prior in the ML objective function does not always result in better performance or higher classification accuracy [34].

It is also possible to use a prior on training unlabeled data. In semi-supervised approach of [10] the conditional entropy of unlabeled data is used as a regularizer and incorporated into the MAP objective function. We will talk about the use of such regularizer in the next chapter.

For large CRFs exact training is often computationally intractable and various approximate methods are used, such as mean field approximation or pseudo likelihood maximization [46, 49]. In the next section we will talk about maximum pseudo likelihood. We then talk about virtual evidence boosting [22], one of the most recent approximation methods for CRF.

3.2 Pseudo Likelihood

An alternative to approximating the conditional likelihood is to change the objective function, that is instead of maximizing the likelihood function we maximize the *pseudo likelihood* of the training data. Maximum pseudo likelihood, MPL [1], is such a technique. For MPL the CRF is cut into a set of independent patches; each patch consists of a hidden node or class label y_i,

the true value of its direct neighbors and the observations, i.e., the Markov Blanket(MB_{y_i}) of the node. To understand how MPL approximation works let us look at the simple CRF example we introduced at the beginning of this chapter. As shown in figure (3), for MPL the CRF is cut into patches such that there is no more than one hidden label in each patch. To contain the information from the neighbors, the true labels of neighboring nodes will be added to the local attributes of the hidden node. By doing this the structure of the CRF becomes simplified and turns into a set of patches. MPL estimation is in fact MAP estimation on the simplified model.

The parameter estimation then becomes the pseudo log-likelihood maximization:

$$L_{pseudo}(\theta) = \sum_{i=1}^{N} \log(p(y_i|MB_{y_i}, \theta)) \tag{3.7}$$

$$= \sum_{i=1}^{N} \log \frac{\exp(\sum_{k=1}^{K} \theta_k f_k(MB_{y_i}, y_i))}{\sum_{y_i'} \exp(\sum_{k=1}^{K} \theta_k f_k(MB_{y_i'}, y_i'))} \tag{3.8}$$

MPL has been known to over-estimate the dependency parameters in some cases and there is no general guideline on when it can be safely used [8]. In the next section we will talk about virtual evidence boosting (VEB) , another approximation technique. At the end of this chapter we will provide empirical result comparing the performance of MPL and VEB.

3.3 Virtual Evidence Boosting

Limited work has been done on feature selection in CRFs [27, 42]. In this section we describe a supervised training algorithm, virtual evidence boosting (VEB), that simultaneously performs feature selection and learns the model. At each training iteration, VEB uses boosting to select a set of features. Boosting has been extensively used for many supervised problems in various domains [6, 7, 42]. Given independent training instances, a boosting algorithm learns an ensemble of weak learners. The final classification

decision is made based on the weighted combination of these weak learners. Using boosting for training CRFs might sound unusual since in boosting, contrary to CRFs, we assume that labels are independent. However, inspired by MPL, if we chunk a CRF to a set of patches we could use boosting on each patch as if the labels are independent. We are not talking about boosting much in this thesis since there are a good number of references on this subject. We will just briefly describe a particular boosting algorithm, LogitBoost [7]. We then talk about an extension of LogitBoost which is used for training CRFs. Similar to other boosting algorithms, in Logitboost [7] we are dealing with an ensemble of weak learners F. We start with an empty ensemble and at each boosting iteration, we add the best found weak learner. The best ensemble of weak learners is the one that maximizes per-label-log-likelihood:

$$L_{logit} = \sum_{i=1}^{N} \log p(y_i) \tag{3.9}$$

$$p(y_i) = \begin{cases} \frac{e^{-F(\mathbf{x_i})}}{e^{F(\mathbf{x_i})}+e^{-F(\mathbf{x_i})}} & \text{if } y_i = 0 \\ \frac{e^{F(\mathbf{x_i})}}{e^{F(\mathbf{x_i})}+e^{-F(\mathbf{x_i})}} & \text{if } y_i = 1 \end{cases}$$

where without loss of generality we assign reverse ensembles for $y_i = 0$ and $y_i = 0$. Logitboost is very similar to logistic regression. We should just think of $e^{F(\mathbf{x_i})}$ and $e^{-F(\mathbf{x_i})}$ as potentials for $y_i = 1$ and $y_i = 0$ respectively. In LogitBoost using a Newton step, we minimize the objective function of (3.9) with respect to the ensemble F. In each step the next weak learner is obtained by solving the following weighted least square problem:

$$f_t(\mathbf{x}) = \arg\min_f \sum_{i=1}^{N} w_i (f(\mathbf{x}_i) - z_i)^2,$$

where $w_i = p(y_i)(1 - p(y_i))$ and $z_i = \frac{y_i - 0.5}{p(y_i)}$ are respectively the weight and working response for instance i.

Instead of observations we could use a given distribution over the observation domain, $\{1, \ldots, D\}$ [21]. That is, we could present the training data

as $(\mathbf{ve}(\mathbf{x}_i), \mathbf{y}_i)$'s, where $\mathbf{ve}_i$ is the virtual evidence or the distribution of instance i. In this case while minimizing the negative per-label-log-likelihood, the computation of the posterior probability, $p(\mathbf{y}_i)$, should take virtual evidence into account as well. The new weighted least square problem also incorporates the expectation of the residuals and becomes [21]:

$$f_t(\mathbf{x}) = \arg\min_f \sum_{i=1}^{N} w_i E(f(\mathbf{x}_i) - z_i)^2.$$

Virtual evidence boosting, VEB is an extension of LogitBoost algorithm [7]. VEB integrates boosting based feature selection into CRF training and maximizes its objective by sequentially learning a set of weak learners f_ts. The objective function used in VEB is very similar to MPL, except that VEB uses the messages from the neighboring nodes as virtual evidence instead of using the true labels of neighbors. The use of virtual evidence helps to reduce over-estimation of neighborhood dependencies. We briefly explain the approach here but please refer to [22] for more detail.

VEB incorporates two types of observations nodes: (i) *hard* evidence corresponding to the observations $\mathbf{ve}(\mathbf{x}_i)$, which are indicator functions at the observation values and (ii) *soft* evidence, corresponding to the messages from neighboring nodes $\mathbf{ve}(\mathbf{n}(y_i))$, which are discrete distributions over the hidden states. Let $\mathbf{ve_i} \triangleq \{\mathbf{ve}(\mathbf{x}_i), \mathbf{ve}(n(y_i))\}$. The objective function of VEB is as follows:

$$L_{VEB}(\theta) = \sum_{i=1}^{N} \log(p(y_i|\mathbf{ve_i}, \theta)), \ \textit{where} \tag{3.10}$$

$$p(y_i|\mathbf{ve_i}, \theta) = \frac{\sum_{\mathbf{ve_i}} \mathbf{ve_i} \exp(\sum_{k=1}^{K} \theta_k f_k(\mathbf{ve_i}, y_i))}{\sum_{y_i'} \sum_{\mathbf{ve_i}} \mathbf{ve_i} \exp(\sum_{k=1}^{K} \theta_k f_k(\mathbf{ve_i}, y_i'))} \tag{3.11}$$

VEB learns a set of weak learners f_ts iteratively and estimates the combined feature $F_t = F_{t-1} + f_t$ by solving the following weighted least square error(WLSE) problem:

$$f_t(\mathbf{ve}_i) = \arg\min_f \sum_{i=1}^{N} w_i E(f(\mathbf{ve}_i) - z_i)^2 \tag{3.12}$$

$$= \arg\min_f [\sum_{i=1}^{N} \sum_{\mathbf{ve}_i} w_i p(y_i|\mathbf{ve_i})(f(\mathbf{ve}_i) - z_i)^2] \tag{3.13}$$

$$where\ w_i = p(y_i|\mathbf{ve_i})(1 - p(y_i|\mathbf{ve_i})), \tag{3.14}$$

$$z_i = \frac{y_i - 0.5}{p(y_i|\mathbf{ve_i})} \tag{3.15}$$

The w_i and z_i in equation (3.15) are the boosting weight and working response respectively for the i^{th} data point, exactly as in LogitBoost. However, the least square problem for VEB (equation 3.13) involves ND (D is the dimension of the observations) points because of virtual evidence as opposed to N points in LogitBoost. Although equation (3.15) is given for the binary case (i.e. $y_i \in \{0, 1\}$), it is easily extendible to the multi-class case and we have done that in our experiments. At each iteration, $\mathbf{ve}_i$ is updated as messages from $\mathbf{n}(y_i)$ change with the addition of new features. We run belief propagation (BP) to obtain the virtual evidence before each iteration. The CRF feature weights, θ's, are computed by solving the WLSE problem:

$$\theta_k = \frac{\sum_{i=1}^{N} w_i z_i n_{ki}}{\sum_{i=1}^{N} w_i n_{ki}} \tag{3.16}$$

In case of local features, n_{ki} is the count of feature k in data instance i. For the compatibility features, n_{ki} is the virtual evidence from the neighbors.

Algorithm 1: Training CRFs using VEB

inputs : structure of CRF and training data $(\mathbf{x}_i, y_i)$, with $y_i \in \{0, 1\}$, $1 \leq i \leq M$, and $F_0 = 0$

output: Learned F_T and their corresponding weights, θ

```
for t = 1, 2, ..., T do
    Run BP using F_t to get virtual evidence ve_i;
    for i = 1, 2, ..., N do
        Compute likelihood p(y_i | ve_i);
        Compute w_i and z_i using equation (3.15)
    end
    Obtain "best" weak learner f_t according to equation (3.13);
    Update F_t = F_{t-1} + f_t ;
end
```

3.4 Experiments

In the first set of experiments we evaluate supervised training methods on activity datasets. These experiments demonstrate the superiority of CRFs over other proposed methods [4, 18] for activity recognition applications. They also provide an empirical comparison between different training algorithms for CRFs. Moreover, by providing results for supervised techniques we can benchmark the performance of the semi-supervised methods, described in the next chapter.

We evaluate four different supervised training approaches, namely maximum a posteriori using all observed features(MAP), (MAP+Boost) using a subset of features selected in advance, maximum pseudo likelihood(MPL) and virtual evidence boosting (VEB). We already explained MAP, MPL and VEB in this chapter. The fourth method, MAP+Boost, is similar to MAP with one difference. Before training via MAP+Boost, we perform feature selection with decision stumps and select the top 50 features corresponding to each activity. We then union these features (about 200 of them) and feed them to the CRF. This simple approach was also used in [4, 22]. All the learned models (trained by MAP, MPL, VEB and ML+Boost) are tested using standard MAP estimate and belief propagation. We used a l_2-norm shrinkage prior as a regularizer for the MAP and MPL methods.

Labeled	*Average Accuracy (%) - Dataset 1*				
	MAP	MAP+Boost	MPL	MPL+Boost	VEB
40%	58.4 ± 5.8	63.0 ± 6.3	59.2 ± 5.8	63.8 ± 4.5	67.9 ± 9.9
60%	62.7 ± 6.6	69.4 ± 3.9	66.9 ± 4.8	70.2 ± 3.1	82.6 ± 7.3
80%	73.0 ± 4.2	81.8 ± 4.7	78.2 ± 4.3	80.9 ± 5.7	90.3 ± 4.7
100%	77.8 ± 3.4	87.0 ± 2.3	80.6 ± 4.9	88.2 ± 5.5	91.5 ± 3.8

Table 3.1: Accuracy ± 95% confidence interval of the supervised algorithms on activity dataset 1 (multi-person shorter duration data; for details refer to chapter 2). Note that VEB outperforms other supervised training methods.

Labeled	*Average Accuracy (%) - Dataset 2*				
	MAP	MAP+Boost	MPL	MPL+Boost	VEB
40%	59.4 ± 5.3	60.1 ± 3.0	58.2 ± 3.5	59.6 ± 4.2	67.9 ± 6.7
60%	74.3 ± 3.7	75.8 ± 3.3	74.3 ± 2.9	75.9 ± 2.5	88.5 ± 5.1
80%	80.6 ± 2.9	84.8 ± 2.9	81.1 ± 5.3	85.2 ± 4.7	93.4 ± 3.8
100%	86.2 ± 3.1	87.5 ± 3.1	87.8 ± 3.2	88.6 ± 5.2	93.8 ± 4.6

Table 3.2: Accuracy ± 95% confidence interval of the supervised algorithms on activity dataset 2 (multi-day, single person, longer duration dataset). Note that VEB outperforms other supervised methods

We compare the supervised techniques, ML, MAP+Boost, and VEB, with increasing amount of labeled data.

In these experiments we evaluate supervised training methods on activity datasets. These experiments demonstrate the superiority of CRFs over other proposed methods [4, 18] for activity recognition applications. Moreover, by providing results for supervised techniques we can benchmark the performance of the semi-supervised methods, described in the next chapter.

3.5 Complexity Analysis

Since VEB does not need to use optimizers such as quasi-Newton methods to learn the weight parameters, VEB is much faster than MAP. For each training iteration in MAP the cost of running BP is $O(c_l n s^2)$ [10] whereas

n	length of training sequence
c_l	number of labeled training sequences
s	number of states
D	dimension of observations

Table 3.3: Description of variables in this section.

the cost of each boosting iteration in sVEB is $O(c_l n s^2)$. Moreover, the number of training iterations needed is usually much higher than the number of boosting iterations because optimizers such as L-BFGS require many more iterations to reach convergence in high dimensional spaces. For example, for dataset 1, we needed about 600 iterations for MAP to converge but we ran VEB for only 50 iterations. Table (4.3) shows the time for performing the experiments on activity datasets (as described in the previous section) [1].

The space complexity of VEB is linearly smaller than MAP. MAP has the space complexity of $O(ns^2D)$ in the best case [10], whereas VEB has a lower space cost of $O(ns^2D_b)$, because usually in the feature selection step $D_b \ll D$. Therefore, the difference becomes significant when we are dealing with high dimensional data, particularly if they include a large number of redundant features.

	Time (hours)		
	MAP	MAP+Boost	VEB
Dataset 1	34	18	2.5
Dataset 2	7.5	4.25	0.4

Table 3.4: Training time in hours for the supervised algorithms. Note that the experiments were performed in a Matlab environment. In a more optimized development environment, the algorithms would be much faster. However, we could still compare the time complexity of algorithms based on their relative time.

[1] The experiments were run in Matlab environment and as a result they took longer.

3.6 Summary

In this chapter we gave a brief overview of two well-known CRF training algorithms, maximum likelihood and maximum a posteriori estimation as well as a famous approximation algorithm, maximum pseudo likelihood. We then described virtual evidence boosting(VEB). VEB is a state-of-the-art supervised training algorithm that performs feature-selection at the same time as training. In next two chapters we will talk about, semi-supervised and active learning extensions of VEB. In this chapter we also provided a comparison of supervised approaches and evaluated them on two activity datasets.

Chapter 4

Semi-supervised Training of Conditional Random Fields

Semi-supervised learning methods have been extensively studied in the past decade [3]. In the semi-supervised paradigm, the algorithm not only takes advantage of labeled data but also incorporates unlabeled data into training as well. Given a small amount of labeled data, the classification accuracy is typically low. However, by employing semi-supervised approaches and incorporating a large amount of unlabeled data the performance can be significantly improved. In those applications that involve easy data collection but expensive labeling, semi-supervised methods are highly attractive. In activity recognition, as we described in chapter 2, training data is abundant, but labeling is expensive and in many cases impossible.

In this chapter, we give a brief overview of semi-supervised maximum a posteriori [10, 26]. We then introduce our novel CRF training algorithm, semi-supervised virtual evidence boosting (sVEB). This algorithm is an extension of VEB [22] which we described in the last chapter. At the end, we will provide extensive experimental results to evaluate the proposed semi-supervised methods.

4.1 Semi-supervised Maximum a Posteriori

For semi-supervised training of CRFs, Jiao *et al.* [10] have proposed an algorithm that utilizes unlabeled data via entropy regularization – an extension of the approach proposed by [9] to structured CRF models. The objective function that is maximized during semi-supervised training of CRFs is given below, where $(\mathbf{x}_l, \mathbf{y}_l)$ and $(\mathbf{x}_u, \mathbf{y}_u)$ represent the labeled and unlabeled data

respectively:

$$L_{SS}(\theta) = \log p(\mathbf{y}_l|\mathbf{x}_l, \theta) + \alpha \sum_{\mathbf{y}_u} p(\mathbf{y}_u|\mathbf{x}_u, \theta) \log p(\mathbf{y}_u|\mathbf{x}_u, \theta) - \frac{\|\theta\|}{2\sigma^2} \quad (4.1)$$

This objective function resembles the function of maximum a posteriori training with this difference that here we have a second regularizer,

$$\alpha \sum_{\mathbf{y}_u} p(\mathbf{y}_u|\mathbf{x}_u, \theta) \log p(\mathbf{y}_u|\mathbf{x}_u, \theta).$$

By using the conditional entropy of unlabeled data as a regularizer in the objective function, the algorithm incorporates unlabeled data into training. By minimizing the conditional entropy of the unlabeled data, the algorithm will generally find a labeling of the unlabeled data that mutually reinforces the supervised labels. One drawback of this objective function is that it is no longer concave and in general there will be local maxima. The authors [10] showed that this method, semi-supervised maximum a posteriori (sMAP), is still effective in improving an initial supervised model, MAP. However, sMAP is extremely slow because of the computational bottlenecks involved in training. Mann *et al.* proposed a more efficient sMAP by taking advantage of some mathematical simplifications in computing the gradient. Although the new approach is faster it is still as slow as MAP and does not perform feature selection.

In the next section, we will introduce semi-supervised VEB which also uses the conditional entropy of unlabeled data into training. sVEB is the semi-supervised extension of VEB. Similar to VEB, within the training algorithm, sVEB has a feature-selection routine that makes it fast and efficient.

4.2 Semi-supervised Virtual Evidence Boosting

In this work, we develop semi-supervised virtual evidence boosting (sVEB) that combines feature selection with semi-supervised training of CRFs. sVEB extends the VEB framework to take advantage of unlabeled data via minimum entropy regularization similar to [9, 10, 17]. The new objective func-

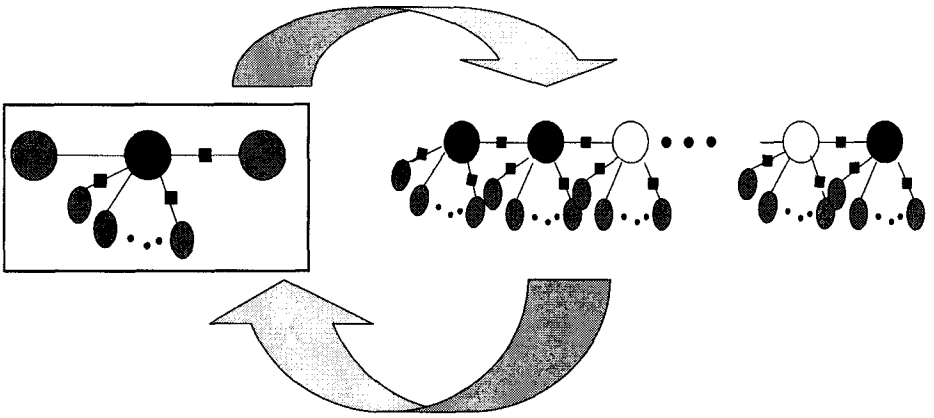

Figure 4.1: A graphical illustration of semi-supervised VEB. On the right a chain CRF is shown with some labeled(coloured hidden states) and some unlabeled instances. On the left a patch from the chain, consisting a state and corresponding virtual evidence (observations and neighbouring messages) is shown. In each iteration of VEB we run BP on the chain to compute likelihood measures and virtual evidence (on right); then we perform feature selection on a patch (on left). Having the new ensemble of features we run BP on the chain (right) and we keep iterating until we reach the maximum number of iterations.

tion L_{sVEB} we propose is as follows, where $(i = 1 \cdots N)$ are labeled and $(i = N+1 \cdots M)$ are unlabled examples:

$$L_{sVEB} = \sum_{i=1}^{N} \log p(y_i|\mathbf{ve_i}) + \alpha \sum_{i=N+1}^{M} \sum_{y'_i} \log p(y'_i|\mathbf{ve_i}) p(y'_i|\mathbf{ve_i}) \qquad (4.2)$$

The sVEB algorithm, similar to VEB, maximizes the conditional soft pseudo-likelihood of the labeled data but in addition minimizes the conditional entropy over unlabeled data. The α is a tuning parameter for controlling how much influence the unlabeled data will have.

By considering the soft pseudo-likelihood in L_{sVEB} and using BP to estimate

$p(y_i|\mathbf{ve_i})$, sVEB can use boosting to learn the parameters of CRFs. The virtual evidence from the neighboring nodes captures the label dependencies. There are three different types of feature functions, f's: for continuous observations $f_1(x_i)$ is a linear combination of decision stumps, for discrete observations the learner $f_2(x_i)$ is expressed as indicator functions, and for virtual evidence the weak learner $f_3(x_i)$ is the weighted sum of two indicator functions (for binary case). These functions are computed as follows, where δ is an indicator function, h is a threshold for the decision stump, and D is the number of dimensions of the observations:

$$f_1(x_i) = \theta_1\delta(x_i \geq h) + \theta_2\delta(x_i < h) \tag{4.3}$$

$$f_2(x_i) = \sum_{k=1}^{D} \theta_k\delta(x_i = d) \tag{4.4}$$

$$f_3(y_i) = \sum_{k=0}^{1} \theta_k\delta(y_i = k) \tag{4.5}$$

Similar to LogitBoost and VEB, the sVEB algorithm estimates a combined feature function F that maximizes the objective by sequentially learning a set of weak learners, f_t's (i.e. iteratively selecting features). In other words, sVEB solves the following weighted least-square error (WLSE) problem to learn f_ts:

$$f_t = \arg\min_f [\sum_{i=1}^{N} \sum_{\mathbf{ve}_i} w_i p(y_i|\mathbf{ve_i})(f(\mathbf{x_i}) - z_i)^2 +$$

$$\sum_{i=N+1}^{M} \sum_{y_i'} \sum_{\mathbf{ve}_i} w_i p(y_i'|\mathbf{ve_i})(f(\mathbf{x_i}) - z_i)^2] \tag{4.6}$$

For labeled data (first term in equation 4.6), boosting weights, w_i's, and working responses, z_i's, are computed as described in equation (3.15). But for the case of unlabeled data the expression for w_i and z_i becomes more complicated because of the entropy term. We present the equations for w_i and z_i below, please refer to the Appendix for the derivations:

$$\begin{aligned} w_i &= \alpha^2(1-p(y_i|\mathbf{ve_i}))[p(y_i|ve_i)(1-p(y_i|\mathbf{ve_i}))+\log p(y_i|\mathbf{ve_i})] \\ z_i &= \frac{(y_i-0.5)p(y_i|\mathbf{ve_i})(1-\log p(y_i|\mathbf{ve_i}))}{\alpha[p(y_i|\mathbf{ve_i})(1-p(y_i|\mathbf{ve_i}))+\log p(y_i|\mathbf{ve_i})]} \end{aligned} \tag{4.7}$$

The soft evidence corresponding to messages from the neighboring nodes is obtained by running BP on the entire training dataset (labeled and unlabeled). The CRF feature weights θ_ks are computed by solving the WLSE problem (e.q.(4.6)),

$$\theta_k = \frac{\sum_{i=1}^{M}\sum_{y_i} w_i z_i n_{ki}}{\sum_{i=1}^{M}\sum_{y_i} w_i n_{ki}} \tag{4.8}$$

Algorithm 2 gives the pseudo-code for sVEB. The main difference between VEB and sVEB are steps $7-10$, where we compute w_i's and z_i's for all possible values of y_i based on the virtual evidence and observations of unlabeled training cases. The boosting weights and working responses are computed using equation (4.7). The weighted least-square error (WLSE) equation (4.6) in step 10 of sVEB is different from that of VEB and the solution results in slightly different CRF feature weights, θ's. One of the major advantages of VEB and sVEB over MAP and sMAP is that the parameter estimation is done by mainly performing feature counting. Unlike MAP and sMAP, we do not need to use an optimizer to learn the model parameters which results in a huge reduction in the time required to train the CRF models. Please refer to the complexity analysis section for details.

4.3 Experiments

In this section we will perform extensive experiments on synthetic and real activity datasets to evaluate the proposed semi-supervised CRF training algorithm, sVEB.

Algorithm 2: Training CRFs using semi-supervised VEB

inputs : structure of CRF and training data $(\mathbf{x}_i, y_i)$, with $y_i \in \{0, 1\}$, $1 \leq i \leq M$, and $F_0 = 0$

output: Learned F_T and their corresponding weights, θ

```
for t = 1, 2, ..., T do
    Run BP using F_t to get virtual evidence ve_i;
    for i = 1, 2, ..., N do
        Compute likelihood p(y_i|ve_i);
        Compute w_i and z_i using equation (3.15)
    end
    for i = N + 1, ..., M and y_i = 0, 1 do
        Compute likelihood p(y_i|ve_i);
        Compute w_i and z_i using equation (4.7)
    end
    Obtain "best" weak learner f_t according to equation (4.6) and
    update F_t = F_{t-1} + f_t ;
end
```

In the first set of experiments, we analyze how much the complexity of the underlying CRF affects the performance using synthetic data. We would like to evaluate the performance of sMAP and sVEB when the number of states and the observation dimensions increase. We also investigate the effect of the tuning parameter α on the performance of the training algorithm. In the second set of experiments, we evaluate the benefit of feature selection and using unlabeled data on two real-world activity datasets.

We compare the performance of the semi-supervised virtual evidence boosting(sVEB) presented in this paper to the semi-supervised maximum a posteriori(sMAP) method of Jia *et al.* [10]. We evaluate a variation of sMAP, sMAP+Boost, on the activity dataset. In this variation, we first perform feature selection with decision stumps, similar to the work of Choudhury *et al.* [4], and then feed the selected features to sMAP. By comparing the performance of sMAP+Boost, we show the effectiveness of simultaneous feature selection in the training.

4.3.1 Experiments on Synthetic Data

The synthetic data is generated using a first-order Markov Chain with self-transition probabilities set to 0.9. For each model, we generate five sequences of length 4,000 and divide each trace into sequences of length 200. We randomly choose 50% of them as the labeled and the other 50% as unlabeled training data. We perform leave-one-out cross-validation and report the average accuracies.

To measure how the complexity of the CRFs affects the performance of the different semi-supervised methods, we vary the number of local features and the number of states. First, we compare the performance of sVEB and sMAP on CRFs with increasing the number of features. The number of states is set to 10 and the number of observation features is varied from 20 to 400 observations. Figure (1a) shows the average accuracy for the two semi-supervised training methods and their confidence intervals. The experimental results demonstrate that sVEB outperforms sMAP as we increase the dimension of observations (i.e. the number of local features). In the second experiment, we increase the number of classes and keep the dimension of observations fixed to 100. Figure (1b) demonstrates that sVEB again outperforms sMAP as we increase the number of states. Given the same amount of training data, sVEB is less likely to overfit because of the feature selection step. In both these experiments we set the value of tuning parameter, α, to 1.5. To explore the effect of tuning parameter α, we vary the value of α from 0.1 to 10 , while setting the number of states to 10 and the number of dimensions to 100. Figure (1c) shows that the performance of both sMAP and sVEB depends on the value of α but the accuracy decreases for large α's similar to the sMAP results presented in [10].

4.3.2 Experiments on Activity Datasets

We collected two activity datasets using wearable sensors, which include audio, acceleration, light, temperature, pressure, and humidity. As described in chapter 2, the first dataset contains instances of 8 basic physical activities (e.g. walking, running, going up/down stairs, going up/down elevator, sitting, standing, and brushing teeth) from 7 different users. There is on average 30 minutes of data per user and a total of 3.5 hours of data that is

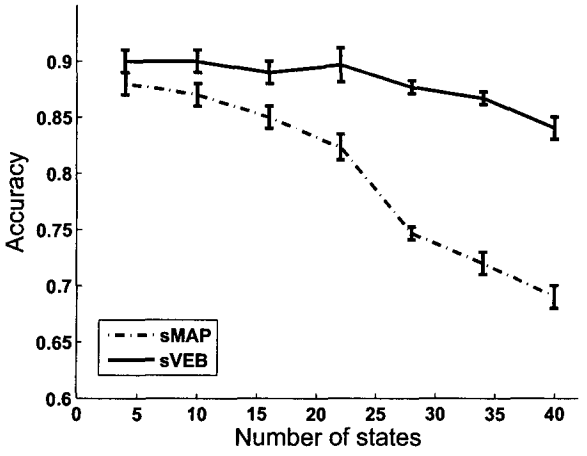

Figure 4.2: Accuracy of sMAP and sVEB for different number of states.

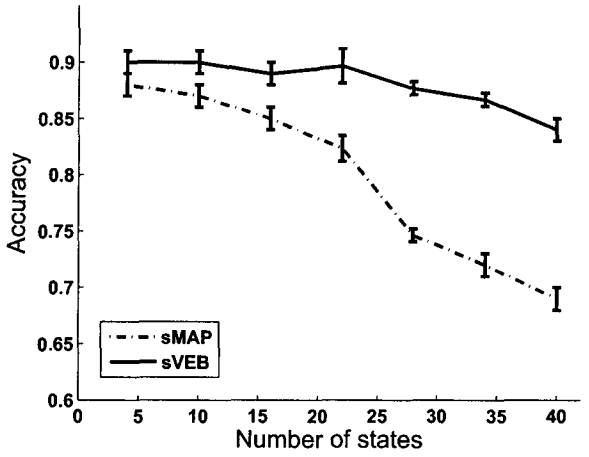

Figure 4.3: Accuracy of sMAP and sVEB for different numbers of local features.

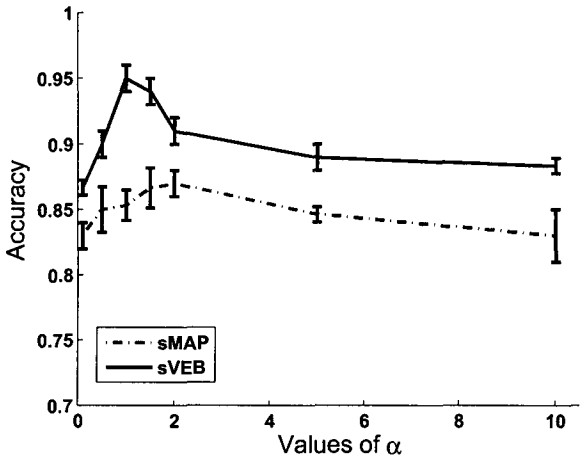

Figure 4.4: Accuracy of sMAP and sVEB for different values of α.

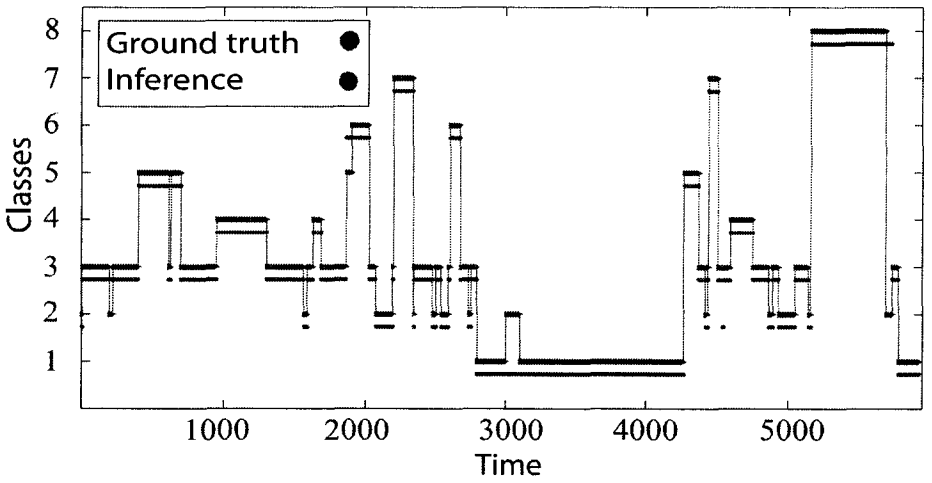

Figure 4.5: An example of a classification trace. The blue(lower) line represents the ground truth and the red(upper) line represents the classification based on semi-supervised VEB where all unlabeled data is incorporated. Note that the distance between these lines is just for presentation purposes and does not indicate a bias (i.e. the value of the blue(lower) line is equal to the integer above it).

Un-labeled	*Average Accuracy (%) - Dataset 1*		
	sMAP+all obs	sMAP+Boost	sVEB
20%	60.8 ± 5.4	66.4 ± 4.2	72.6 ± 2.3
40%	68.1 ± 4.8	76.8 ± 3.4	78.5 ± 3.4
60%	74.9 ± 3.1	81.3 ± 3.9	85.3 ± 4.1

Table 4.1: Accuracy ± 95% confidence interval of semi-supervised algorithms on activity datasets 1.

manually labeled for training and testing purposes. The data is segmented into $0.25s$ chunks resulting in a total of 49613 data points. For each chunk, we compute 651 features. During training, the data from each person is divided into sequences of length 200 and fed into linear chain CRFs as observations.For more details on the dataset please see chapter 2.

The second dataset contains instances of 5 different indoor activities (e.g. computer usage, meal, meeting, watching TV and sleeping) from a single user. We recorded 15 hours of sensor traces over 12 days. As this set contains longer time-scale activities, the data is segmented into 1 minute chunks and 321 different features are computed, similar to the first dataset. There are a total of 907 data points. These features are fed into CRFs as observations, one linear chain CRF is created per day.

We randomly select 40% of the sequences for a given person or a given day as labeled and a different subset as the unlabeled training data. We compare the performance sMAP and sVEB as we incorporate more unlabeled data (20%, 40% and 60%) into the training process. For all the experiments, the tuning parameter α is set to 1.5. We perform leave-one-person-out cross-validation on dataset 1 and leave-one-day-out cross-validation on dataset 2 and report the average the accuracies (please see supplementary material for per class breakdown). The number of features chosen (i. e. the boosting iterations) is set to 50 for both the datasets – including more features did not significantly improve the classification performance.

For both datasets, incorporating more unlabeled data improves accuracy. The sMAP estimate of the CRF parameters performs the worst. Even with

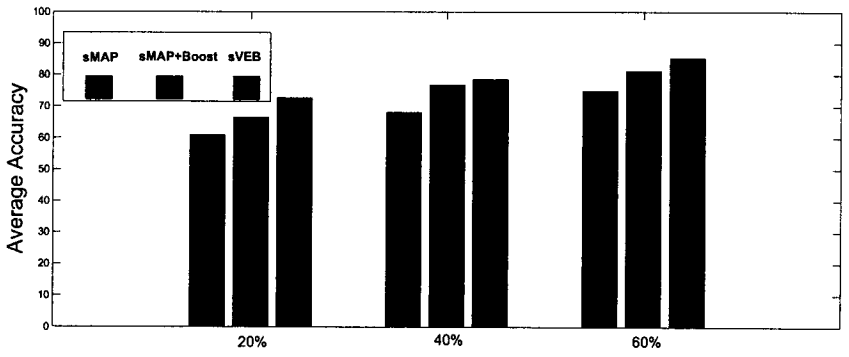

Figure 4.6: Average accuracy of sMAP, sMAP+Boost and sVEB (bars from left to right) on dataset 1 for different amount of unlabeled data incorporated into training. This bar plot shows 20% (left), 40% (middle) and 60% (right).

Un-	*Average Accuracy (%) - Dataset 2*		
labeled	sMAP+all obs	sMAP+Boost	sVEB
20%	71.4 ± 3.2	70.5 ± 5.3	79.9 ± 4.2
40%	73.5 ± 5.8	74.1 ± 4.6	83.5 ± 6.3
60%	75.6 ± 3.9	77.8 ± 3.2	87.4 ± 4.7

Table 4.2: Accuracy ± 95% confidence interval of semi-supervised algorithms on activity datasets 2.

	Average Accuracy (%) - Dataset 1		
labeled	sMAP+all obs	sMAP+Boost	sVEB
5%	59.2 ± 6.5	65.7 ± 8.3	71.2 ± 5.7
20%	66.9 ± 5.9	67.3 ± 8.5	77.4 ± 3.6

Table 4.3: Accuracy ± 95% confidence interval of semi-supervised algorithms on activity datasets 1. As numbers show, sVEB results in higher accuracy.

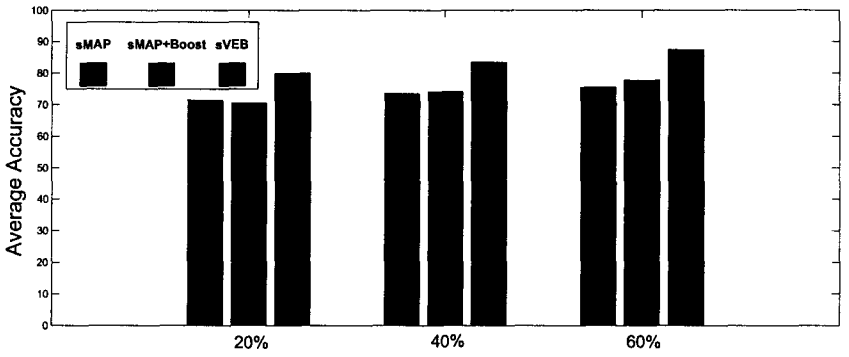

Figure 4.7: Average accuracy of sMAP, sMAP+Boost and sVEB (bars from left to right) on dataset 2 for different amount of unlabeled data incorporated into training. This bar plot shows 20% (left), 40% (middle) and 60% (right).

	Average Accuracy (%) - Dataset 2		
labeled	sMAP+all obs	sMAP+Boost	sVEB
5%	71.2 ± 4.1	68.3 ± 6.7	79.7 ± 7.9
20%	71.4 ± 6.3	73.8 ± 5.2	83.1 ± 6.4

Table 4.4: Accuracy ± 95% confidence interval of semi-supervised algorithms on activity datasets 2. sVEB outperforms other semi-supervised approaches here as well.

the shrinkage prior, the high dimensionality can still cause over-fitting and lower the accuracy. Whereas parameter estimation and feature selection via sVEB consistently results in the highest accuracy. The (sMAP+Boost) method performs better than sMAP but does not perform as well as when feature selection and parameter estimation is done within a unified framework as in sVEB. Table 8 summarize our results. The results of supervised learning algorithms are presented in Table 8. Similar to the semi-supervised results, the VEB method performs the best, the MAP is the worst performer, and the accuracy numbers for the (MAP+Boost) method is in between. The accuracy increases if we incorporate more labeled data during training. Our experiments clearly demonstrate that although adding more unlabeled data is not as helpful as incorporating more labeled data, the use of cheap unlabeled data along with feature selection can certainly boost the performance of the models significantly.

4.4 Complexity Analysis

n	length of training sequence
c_l	number of labeled training sequences
c_u	number of unlabeled training sequences
s	number of states
D	dimension of observations
D_b	dimension of observations in selected features

Table 4.5: Description of variables in this section.

The sVEB algorithm is significantly faster than sMAP because it does not need to use optimizers such as quasi-newton methods to learn the weight parameters. This is similar to the speed-up we gain by choosing VEB over MAP. For each training iteration in sMAP the cost of running BP is $O(c_l n s^2 + c_u n^2 s^3)$ [10] whereas the cost of each boosting iteration in sVEB is $O((c_l + c_u) n s^2)$. The efficient entropy gradient computation that is proposed in [26] reduces the cost of sMAP to $O((c_l + c_u) n s^2)$ but still requires an optimizer to maximize the log-likelihood. Moreover, the number of training iterations needed is usually much higher than the number of boosting iterations because optimizers such as L-BFGS require many more iterations to reach convergence in high dimensional spaces. For example, for dataset

1, we needed about 1000 iterations for sMAP to converge but we ran sVEB for only 50 iterations. Table (8) shows the time for performing the experiments on activity datasets (as described in the previous section) [1]. On the other hand the space complexity of sVEB is linearly smaller to sMAP and MAP. sMAP has the space complexity of $O(ns^2D)$ in the best case [10]. sVEB have a lower space cost of $O(ns^2D_b)$, because of the feature selection step $D_b \ll D$ usually. Therefore, the difference becomes significant when we are dealing with high dimensional data, particularly if they include a large number of redundant features.

	Time (hours)			
	sMAP	sMAP+Boost	Fast sMAP	sVEB
Dataset 1	96	48	37	4
Dataset 2	10.5	8	5.5	0.6

Table 4.6: Training time in hours for the semi-supervised algorithms. Note that the experiments were performed in matlab environment and as a result they take longer than real world implementations. However, we could still compare the time complexity of algorithms based on their relative time.

4.5 Summary

In this chapter we gave an overview of some of the semi-supervised learning algorithm for training CRFs. We empirically compare these algorithms in chapter 6. The computational complexity of the above methods is analyzed in chapter 7. We extensively talked about semi-supervised virtual evidence boosting(sVEB), one of the main contributions of this thesis. sVEB can be seen as an extension of VEB [22] in which the conditional entropy of unlabeled data is incorporated into training as a regularizer for the training objective function. With the same amount of labeled training data, sVEB outperforms VEB (results are mentioned in chapter 7). The shorter version of this chapter has been published [25] by the author and her mentor at Intel Research Seattle. In the next chapter, we will talk about extending VEB in an active-learning frame-work.

[1] The experiments were run in Matlab environment and as a result they took longer.

Chapter 5

Active Training of CRFs

There are many learning situations in which unlabeled data is abundant and easy to collect, however, data labeling is expensive. For example in case of text processing, it is easy to feed text corpora to the system, but going through pages and tagging or labeling the text is time-consuming and error-prone.

In case of activity recognition we have a similar problem. Collecting activity data from wearable sensors is cheap and convenient. We could easily ask a group of volunteers to use wearable sensors for few days while performing different activities. On the other hand, labeling the collected activity data is hard if not impossible in some situations. One way of labeling data is to ask volunteers to keep a diary while performing activities. Although this might seem possible for simple and short-period data collecting, it interferes with the natural flow of activities and it becomes complicated when activities overlap. One of the problems with diary keeping is that the labeling is not reliable particulary if we are training the system for a disabled person. Installing a small camera on patients' body or their wheelchair is much easier than asking them to keep a diary. As a result diary keeping is not applicable for labeling activity data. The other approach for labeling activity data is capturing ground truth video. There are two main problems with this approach. One is that it is very time consuming and boring (not to mention expensive) to go through days of activity videos and segment each part. The second and more important problem is about the privacy issues. For many activities, although the volunteers do not have any problem with collecting the data, they are not comfortable with sharing the captured video. For example many people do not like other people watching them sleeping or brushing teeth.

Employing semi-supervised approaches for activity recognition enables us to reduce the amount of required labeled data for a desired level of accuracy. However, we could reduce the labeling effort even more if we could label the most informative activity segments. Also it will be helpful in many applications if the system could improve its performance online while it is in use. For all these reasons, we desire an active training system that queries users for labels to improve the performance.

Active learning has attracted an increasing attention over past few years [11, 51] and few different approaches have been proposed [11]. However, there has been little work done on active learning for discriminative classifiers such as logistic regression or Conditional Random Fields [12, 38]. In this thesis we aim to propose an active training algorithm for CRFs that is based on Virtual Evidence Boosting. Since VEB has shown to be the most successful for training CRFs particularly in activity recognition [21, 22, 25] we became motivated to develop the the active-learning extension.

In this thesis we talk about active learning of CRFs based on entropy measures. In the rest of this chapter, We describe a simple active maximum likelihood training for which we use an entropy-based confidence score to label more training data and improve the performance. We then introduce the active virtual evidence boosting(aVEB) algorithm which is an extension of semi-supervised virtual evidence boosting. At the end the we evaluate sVEB by providing empirical results on activity datasets.

5.1 Uncertainty-based Sample Selection

A number of active learning systems work based on an uncertainty (or certainty) score. In such systems, given an initial classifier, uncertainty scores are computed for unannounced examples. Instances with highest scores (the most uncertain ones) are regarded as the most informative ones and selected for human labeling. One of the ways to quantify the uncertainty is to come up with an entropy-based measure [12]. The labels for samples with the highest entropy measure change the trained model the most. Therefore the high entropy samples are considered as the most informative ones. In CRFs, the training data consist of subsequences of observations (we have

subsequences of labels only for labeled training data).

Kim *et al.* propose an active learning frame-work for training CRFs for a computational linguistics application [12]. They introduce a mixed score based on the entropy and novelty of the sample. They provide a definition for the sample novelty according to word similarity distance meters. The system queries for the most diverse sample (the one that has low similarity measure to the already queried samples) that has the highest entropy measure. In activity recognition, we do not have words and similarity measures. However, we could measure the entropy of sequences and use that as a measuring stick.

Given an observation subsequence, $\mathbf{x}$, in unlabeled training data and its fixed length, we could come up with the corresponding finite set of all possible label subsequences, $\mathcal{Y}$. Each element of set $\mathcal{Y}$, $\mathbf{y}$, denotes a possible label subsequence. The probability distribution of having $\mathbf{y}$ as the label sequence of $\mathbf{x}$ is $p(\mathbf{y}|\mathbf{x})$ which is s computed using equation (3.1), similar to any other CRF. The entropy of the probability distribution $p(\mathbf{y}|\mathbf{x})$ is the following:

$$H_1 = -\sum_{\mathbf{y}\in\mathcal{Y}} p(\mathbf{y}|\mathbf{x}) \log p(\mathbf{y}|\mathbf{x}) \tag{5.1}$$

The number of possible labeled sequences and the size of $\mathcal{Y}$ grows exponentially as we increase the length of the observation sequence. The complexity of computing the entropy measure by equation (5.1) is also becomes exponentially hard. To overcome this difficulty, instead of computing H_1 for all possible state subsequences, we could find the K highest probability state subsequences and compute the entropy for them. Searching for K highest probable subsequences is cheap when it is done by the Viterbi [44] search algorithm. The new entropy measure becomes:

$$H_2 = -\sum_{k=1}^{K} \frac{p(\mathbf{y}^k|\mathbf{x})}{\sum_{k=1}^{K} p(\mathbf{y}^k|\mathbf{x})} \log \frac{p(\mathbf{y}^k|\mathbf{x})}{\sum_{k=1}^{K} p(\mathbf{y}^k|\mathbf{x})}; \tag{5.2}$$

where $\mathbf{y}^k$ represents the k^{th} best state subsequence.

In entropy-based active training of CRFs, the user is asked to label the training subsequence that has the highest entropy measure. The new labeled

Algorithm 3: Entropy-based Active Training of CRFs.

inputs :

- structure of CRF
- labeled training sequences $L = \{q_1^l \dots q_L^l\}$ containing $(\mathbf{x}_i, y_i)$, with $y_i \in \{0, 1\}$, $1 \leq i \leq N$
- unlabeled training sequences $U = \{q_1^u \dots q_U^u\}$ containing $(\mathbf{x}_i, y_i)$, $N + 1 \leq i \leq M$

output:

- CRF weights, θ

```
1 while system needs more improvement or |U| > 0 do
2     Training: Run BP on labeled training data; Compute
      likelihood p(y|L), given different configurations of y;
3     for j = 1, ..., J do
4         Obtain label for sequence q* for which H_2 is the highest.
5         Update: U = U − q*, L = L ∪ q*, N = N + |q*|.
6     end
7 end
```

training subsequence is then added to the labeled data and system trains the model again.

5.2 Active Virtual Evidence Boosting

In semi-supervised virtual evidence boosting, we have a set of labeled and a set of unlabeled data. Experimental results in chapter 4 show that sVEB outperforms VEB, meaning that incorporating unlabeled data into training improves the performance. However, we would like to achieve the best performance having the least amount of labeled data. In other words, we would like the system to ask for the label of the most informative unlabeled training instances. By informative instances we mean the ones that knowing their labels could change the value of model parameters the most. We could define a confidence score for VEB similar to the entropy-based con-

fidence score we described before. The entropy for the distribution over a subsequence, is as follows:

$$H_{veb} = -\sum_{i=b}^{e}\sum_{y_i'} p(y_i'|\mathbf{ve}_i)\log p(y_i'|\mathbf{ve}_i) \tag{5.3}$$

Here q is a subsequence containing $< (x_b, y_b), \ldots, (x_e, y_e) >$, where b denotes the index of the first instance in q and e denotes the last index of last instance. Similar to precious chapters $\mathbf{ve}_i$ denotes the virtual evidence of instance i. The difference between this measurement and what we described before is that here we use virtual evidence instead of neighbouring states and observations. Also since we are using this score within the VEB setting, number of features are much less than what we have in maximum likelihood setting. We would like to query the user for the label of the most informative subsequence,q^*. That is the one with the highest entropy.

$$q^* = \arg\min_{q\in U}[\sum_{i=b}^{e}\sum_{y_i'} p(y_i'|\mathbf{ve}_i)\log p(y_i'|\mathbf{ve}_i)] \tag{5.4}$$

In the active virtual evidence boosting (aVEB) algorithm, similar to semi-supervised virtual evidence boosting (see chapter 4) we have a set of labeled and unlabeled training data. Through boosting iterations and incorporating both labeled and unlabeled data, we compute virtual evidence and likelihoods. Then if the performance is not satisfactory we ask user to label few unlabeled training subsequences. We then move this newly labeled training data from set of unlabeled subsequences to labeled ones. Once again we train the CRF as we do in sVEB and test if we are content with the performance. This cycle continues until the user is satisfied or wants to stop labeling. Instead of setting a performance threshold, such as computing accuracy error, we could set the algorithm to query the user for a fixed number of time. Algorithm 3, shows the steps of aVEB algorithm. For more efficiency, instead of asking the user to label one subsequence each time, we ask for J sequences.

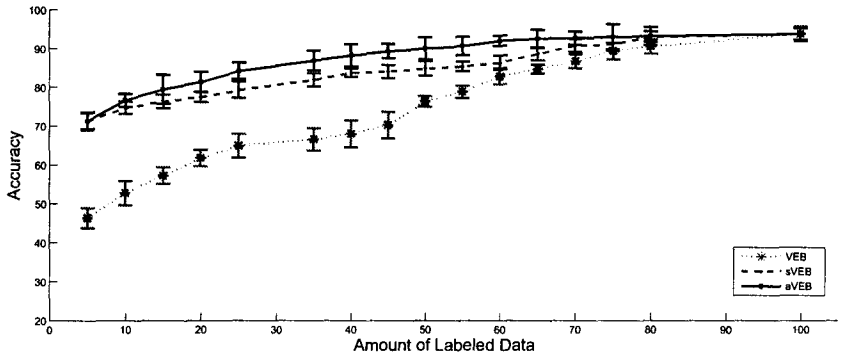

Figure 5.1: Comparison of active virtual evidence boosting (aVEB) algorithm with semi-supervised (sVEB) and supervised virtual evidence boosting(VEB) on dataset 1 (multi-person). The performance is evaluated as we label higher percentage of data and incorporate into training and is measured based on the average classification accuracy during testing. The plots shows that aVEB outperforms sVEB and aVEB. The curves for all approaches reach the same point when we label 100% of the training data.

5.3 Experiments

In the first set of experiments we would like to evaluate the effect of active training of CRFs. We compare aVEB with sVEB and VEB on two activity datasets. In the first experiment we start with a fixed amount of labeled data (5% of training data) and incorporate more labeled and less unlabeled data into training. For baseline methods, sVEB and VEB, we randomly select unlabeled sequences and add them to the labeled training data. But for aVEB we select the unlabeled data based on equation (5.4). Similar to the experiments of chapters 3 and 4, we perform leave-one-person-out cross validation for dataset 1 and leave-one-day-out cross validation for dataset 2. For dataset 1 we use subsequences of length 4 and we set J (the number of querying user each time) to 10. But for dataset 2 subsequences have length 40 and J is 1.

After training the model we do inference using standard MAP and BP. We average the accuracies and compute the 95% interval. The experimental

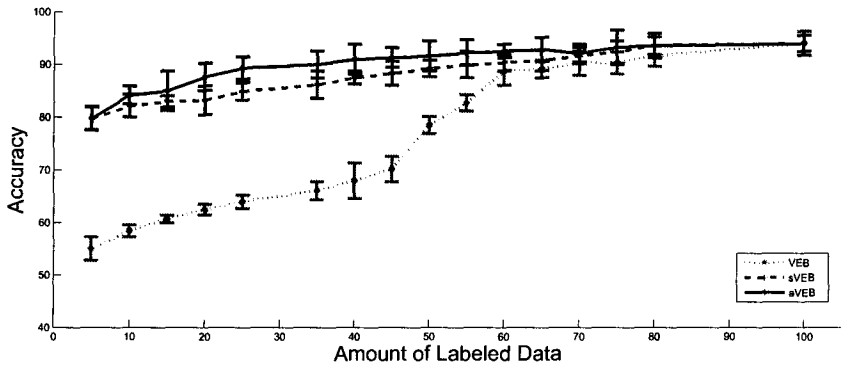

Figure 5.2: Comparison of active virtual evidence boosting (aVEB) algorithm with semi-supervised (sVEB) and supervised virtual evidence boosting(VEB) on dataset 2 (single-person multi-day). The performance is evaluated as we label higher percentage of data and incorporate into training and is measured based on the average classification accuracy during testing. The plots shows that aVEB outperforms sVEB and aVEB. The curves for all approaches reach the same point when we label 100% of the training data.

results are demonstrated in figures (5.1) and (5.2). aVEB and sVEB have the same performance in the beginning since they are basically identical at that point. Later aVEB outperforms sVEB since we randomly add labeled data to sVEB but choose to add the most informative instances to aVEB. The effect of using an entropy-based measure is evident form the empirical comparison. VEB on the other hand has lower accuracy compare to both sVEB and aVEB. The reason behind it is that we incorporate unlabeled data into sVEB and aVEB. At the end, when we are using 100% of the labeled and no unlabeled training data, all three algorithms result the same accuracy.

In the next set of experiments we would like to compare active virtual evidence boosting, aVEB, with entropy based CRF training algorithm described earlier in this chapter. These experiments are designed to observe the effect of using feature selection and virtual evidence boosting in the active training of CRFs. Similar to the previous experiments we start with 5% of training data labeled. Each time we query the label of selected subsequences and add them to the training. We again use subsequence length of 100 for dataset 1 and 40 for dataset2. For dataset 1 we set J to 10 where for the second dataset J is 1. Figures (5.3) and (5.4) demonstrate the experimental results. As VEB and sVEB outperform MAP and sMAP, here we also see that the active training of in the VEB setting performs better than the active training in the MAP setting.

5.4 Summary

In this chapter we gave an overview of the limited work on active training of CRFs, [12, 13]. In details we described the entropy-based approach which is similar to [12]. We introduced active virtual evidence boosting (aVEB) algorithm. aVEB iteratively calls sVEB but each time adds a set of informative labeled subsequences to the training data. We compared the performance of aVEB with sVEB and VEB to show the effectiveness of incorporating active learning into virtual evidence boosting training methods. Moreover we compared aVEB with an entropy-based MAP active learning algorithm. On two sets of activity recognition datasets, aVEB again demonstrates its superiority. There is much room to explore active learning for training CRFs. It will be valuable to investigate decision theoretic approaches [11] or value of information that is proposed for logistic regression [38].

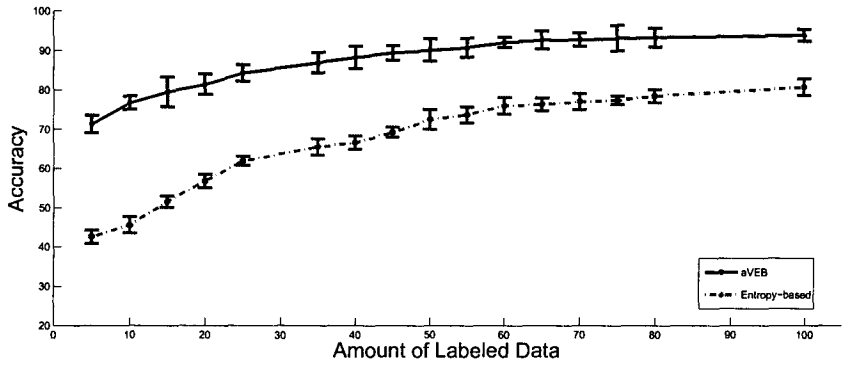

Figure 5.3: Comparison of active virtual evidence boosting (aVEB) algorithm with entropy-based algorithm, described in chapter 5, on dataset 1 (multi-person). The performance is evaluated as we label higher percentage of data and incorporate into training and is measured based on the average classification accuracy during testing. The plots shows that aVEB outperforms the entropy-based approach.

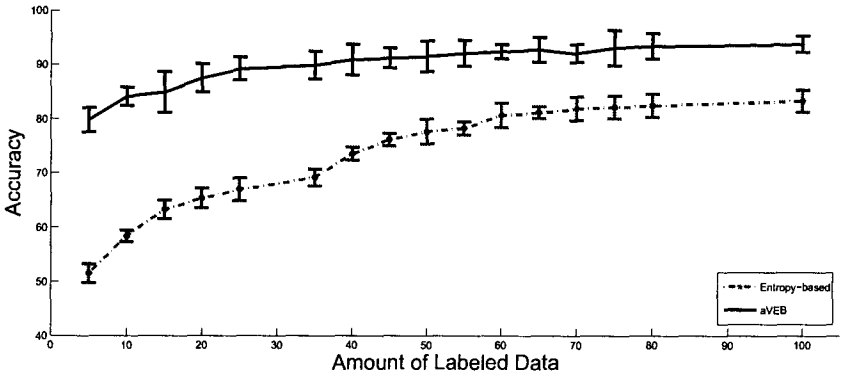

Figure 5.4: Comparison of active virtual evidence boosting (aVEB) algorithm with entropy-based algorithm, described in chapter 5, on dataset 2 (single-person, multi-day). The performance is evaluated as we label higher percentage of data and incorporate into training and is measured based on the average classification accuracy during testing. The plots shows that aVEB outperforms the entropy-based approach.

Algorithm 4: Active Training of CRFs using VEB

inputs :

- structure of CRF
- labeled training sequences $L = \{q_1^l \dots q_L^l\}$ containing $(\mathbf{x}_i, y_i)$, with $y_i \in \{0, 1\}$, $1 \leq i \leq N$
- unlabeled training sequences $U = \{q_1^u \dots q_U^u\}$ containing $(\mathbf{x}_i, y_i)$, $N + 1 \leq i \leq M$
- F_0

output:

- Learned F_T and their corresponding weights, θ

while *system needs more improvement or* $|U| > 0$ **do**
  for $t = 1, 2, \cdots, T$ **do**
    Run BP using F_t to get virtual evidence $\mathbf{ve_i}$;
    for $i = 1, 2, \cdots, N$ **do**
      Compute likelihood $p(y_i|\mathbf{ve_i})$;
      Compute w_i and z_i using equation (3.15)
    end
    for $i = N + 1, ..., M$ *and* $y_i = 0, 1$ **do**
      Compute likelihood $p(y_i|\mathbf{ve_i})$;
      Compute w_i and z_i using equation (4.7)
    end
    Obtain "best" weak learner f_t according to equation (4.6)
    Update: $F_t = F_{t-1} + f_t$;
  end
  for $j = 1, \cdots, J$ **do**
    Obtain label for sequence q^* such that
    $q^* = \arg\min_{q \in U} [\sum_{i=b}^{e} \sum_{y_i'} p(y_i'|\mathbf{ve}_i) \log p(y_i'|\mathbf{ve}_i)]$,equation (5.4);
    Update: $U = U - q^*$, $L = L \cup q^*$, $N = N + |q^*|$.
  end
end

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis we investigated an application driven problem. Activity recognition has attracted increasing attention in recent years mainly because of its numerous applications in health-care, human and computer interaction and surveillance. Inspired by the work of Liao *et al.* [4, 22], we studied application of Conditional Random Fields to this problem domain. Moreover, we extended virtual evidence boosting, the novel training algorithm of [22] to semi-supervised and active learning frame-works. To the best of our knowledge, this thesis for the first time proposes the use of semi-supervised and active learning for activity recognition. Besides providing a review of state-of-the-art CRF training algorithms and evaluating them on our activity datasets we carefully described the two novel algorithms of this thesis:

- We presented semi-supervised virtual evidence boosting (sVEB), a new training method for CRFs, that can simultaneously select discriminative features via modified LogitBoost and utilize unlabeled data via minimum-entropy regularization. Our experimental results demonstrate that sVEB significantly outperforms other training techniques in real-world activity recognition problems. The unified framework for feature selection and semi-supervised training presented in this paper reduces the overall computational cost and the human labeling cost, which are often the major bottlenecks in building large classification systems.

- We introduced a simple yet applicable approach for training CRFs in an active-learning framework. The proposed methodology could be employed in adaptive devices designed for multi-users. The goal of active learning is mainly to make training faster and more efficient by reducing labeling effort. Moreover, the proposed active training methods, similar to the virtual evidence boosting algorithm of Liao *et al.* [22], is computationally efficient. As a result it could be used in on-line adaptive systems such as adaptive mobile devices.

Our promising experimental results demonstrate that activity recognition and ubiquitous computing in general could largely benefit from employing semi-supervised and active learning methods. Although our work was motivated by activity recognition problems, it could be applicable in many other domains. Exploring and improving these new CRF training algorithms for other applications could be an immediate future work. In the next section we will discuss some of the possible future directions that the author has in mind.

6.2 Future Work

Employing machine learning in ubiquitous computing is a relatively new research area and there is much room left for exploring. Activity recognition in particular could be improved by data mining and machine learning methods. Machine learning, on the other hand, could take advantage of a new and interesting application domain. This thesis along with the work of Liao [21, 22], are considered as pioneers on the application of temporal learning in activity recognition. There are a number of theoretical challenges left unsolved. To name a few we could mention clustering users based on behaviourial patterns and hierarchical activity learning. On the other hand, on the application side there is much work to be done. Many applications could take advantage of the methods discussed in this thesis. For example, in the area of ubiquitous computing, they could be used in assistive technologies and adaptive mobile devices.

6.2.1 Hierarchical Activity Recognition

Many complex activities, such as cooking and attending a meeting, consist of few lower-level sub-activities (for example, cooking could contain standing, walking and stirring; meeting could involve sitting, talking and writing). Perhaps, we could approach activity recognition as a hierarchical structural learning problem for which we might have few labels for the higher level data and some labels for the lower level data. Incorporating the labeled and unlabeled data of the two activity levels and the model structure into training will result in a better performance. However, the theory behind semi-supervised hierarchical activity recognition is still needed to be established.

6.2.2 Clustering Users Based on Behavourial Patterns

Transferring activity knowledge between people is very attractive. In the experimental sections of this thesis, we explored the possibilities of training the model based on a group of people and testing it on a different person. However, this data is collected from a particular group of people, graduate students of University of Washington, and at the same environment, the campus of the University of Washington. The data they collected for a sequence of activities is perhaps very different from the data a handicapped person collects. Moreover, outdoor and indoor environments, humid and dry climate affect the collected data as well. One easy way to tackle this problem is to train a model for every new user. However, this is expensive and impractical for real-world applications. One promising solution as Liao also mentions [21] is to cluster users based on their activity patterns and train a model for each cluster. When a new user starts working with the system, she is assigned to one of the clusters and the corresponding activity model. However, the problem is that it is not clear what metric we should use for measuring similarity within activities. Also, since we do not have activity labels for her, it is not clear how we should select a cluster. This could be explored as a model-selection problem.

6.2.3 Adaptive Mobile Devices

In chapter 5 we talked about active-training of the model. Particularly when activity recognition is based on the wearable sensors data, user adaptation becomes necessary. Each user has different hand or shoulder movements or navigates different environments. As a result it is important to adapt the device to each user. The proposed methods of this thesis could be really applicable for adapting intelligent health care devices for new patients. User modeling and customized software applications are not new concepts. However, ubiquitous computing still lacks adaptive technologies. In this thesis we did not talk about design of such devices. However, it will be an interesting direction perhaps in the fields of ubiquitous computing.

Bibliography

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

[2] M. Bezzi and R. Groenevelt. Towards understanding and modeling individual behavior and group dynamics. In *Workshop in Pervasive Technology Applied Real-World Experiences with RFID and Sensor Networks*, 2005.

[3] O. Chapelle, B. Schölkopf, and A. Zien, editors. *Semi-Supervised Learning*. MIT Press, Cambridge, MA, 2006.

[4] T. Choudhury, J. Lester, K. 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.

[5] T. Dietterich, A. Ashenfelter, and Y. Bulatov. Training conditional random fields via gradient tree boosting. In *Proc. of the International Conference on Machine Learning (ICML)*, 2004.

[6] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1), 1997.

[7] J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting. *The Annals of Statistics*, 38(2), 2000.

[8] C. J. Geyer and E. A. Thompson. Constrained Monte Carlo Maximum Likelihood for dependent data. *Journal of Royal Statistical Society*, 1992.

[9] Y. Grandvalet and Y. Bengio. Semi-supervised learning by entropy minimization. In *Advances in Neural Information Processing Systems (NIPS)*, 2004.

[10] F. Jiao, W. Wang, C. H. Lee, R. Greiner, and D. Schuurmans. Semi-supervised conditional random fields for improved sequence segmentation and labeling. In *International Committee on Computational Linguistics and the Association for Computational Linguistics*, 2006.

[11] A. Kapoor, E. Horvitz, and S. Basu. Selective supervision: Guiding supervised learning with decision-theoretic active learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[12] S. Kim, Y. Song, K. Kim, J. Cha, and G. Lee. Mmr-based active machine learning for bio named entity recognition. In *Human Language Technologies (HLT)*, 2006.

[13] T. Kristjannson, A. Culotta, P. Viola, and A. McCallum. Interactive information extraction with constrained conditional random fields. In *Proceedings of American Association for Artificial Intelligence (AAAI)*, 2004.

[14] S. Kumar and M. Hebert. Discriminitive fields for modeling spatial dependencies in natural images. In *Proceedings of Neural Information Processing Systems*, 2004.

[15] 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.

[16] J. Lafferty, X. Zhu, and Y. Liu. Kernel conditional random fields: Representation and clique selection. In *Proceedings of the Twenty-First International Conference in Machine Learning*, 2004.

[17] C. Lee, S. Wang, F. Jiao, Schuurmans D., and R. Greiner. Learning to Model Spatial Dependency: Semi-Supervised Discriminative Random Fields. In *NIPS*, 2006.

[18] J. Lester, T. Choudhury, and G. Borriello. A practical approach to recognizing physical activities. In *Pervasive*, 2006.

[19] J. Lester, T. Choudhury, G. Borriello, S. Consolvo, J. Landay, K. Everitt, and I. Smith. Sensing and modeling activities to support physical fitness. In *Ubicomp Workshop: Monitoring, Measuring and Motivating Exercise: Ubiquitous Computing to Support Fitness*, 2005.

[20] W. Li and M. Andrew. Semi-supervised sequence modeling with syntactic topic models. In *Proc. of the National Conference on Artificial Intelligence (AAAI)*, 2005.

[21] L. Liao. Location-based activity recognition. *PhD Thesis, Department of Computer Science and Enginering, University of Washington*, 2006.

[22] L. Liao, T. Choudhury, D. Fox, and H. Kautz. Training conditional random fields using virtual evidence boosting. In *Proc. of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

[23] P. Lukowicz, H. Junker, T. Stäger, M., and G. Tröster. Wearnet: A distributed multi-sensor system for context aware wearables. In *Proceedings of the 4th international conference on Ubiquitous Computing. Springer-Verlag*, 2002.

[24] P. Lukowicz and J. Ward. Recognizing workshop activity using body worn microphones and accelerometer. In *Pervasive Computing: Proceedings of the 2nd International Conference*, 2004.

[25] M. Mahdaviani and T. Choudhury. Fast and scalable training of semi-supervised CRFs with application to activity recognition. In *Neural Information Processing Systems (NIPS)*, 2007.

[26] G. Mann and A. McCallum. Efficient computation of entropy gradient for semi-supervised conditional random fields. In *Proccedings of North American Chapter of the Association for Computational Linguistics and Humen Language Technologies*, 2007.

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

[28] R. McDonald and F. Pereira. Identifying gene and protein mentions in text using conditional random fields. In *BioCreative*, 2004.

[29] T. McKenna. Video surveillance and human activity recognition for anti-terrorism and force protection. In *Proceedings. IEEE Conference on Advanced Video and Signal Based Surveillance*, 2003.

[30] A. Ng. Feature selection, L_1 vs. L_2 regularization, and rotational invariance. In *Proceedings of the Twenty-first International Conference on Machine Learning*, 2004.

[31] K. Nigam, A. McCallum, A. Thrun, and T. Mitchell. Text classification from labeled and unlabeled documents using EM. *Machine learning*, 2000.

[32] S. Park, I. Locher, A. Savvides, M. Sirvastava, A. Chen, R. Muntz, and S. Yuen. Design of a wearable sensor badge for smart kindergarten. In *Proceedings of 6th International Symposium on Wearable Computers*, 2002.

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

[34] F. Peng and A. McCallum. Accurate information extraction from research papers using conditional random fields. In *Proceedings of Human Language Technology Conference and North American Chapter of the Association for Computational Linguistics*, 2004.

[35] D. Pinto, A. McCallum, X. Wei, and W. B. Croft. Table extraction using conditional random fields. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2003.

[36] A. Quattoni, M. Collins, and T. Darrel. Conditional random fields for object recognition. In *Proceedings of International Joint Conference on Artificial Intelligence*, 2004.

[37] L. Rabiner. A tutorial on Hidden Markov Models and selected applications in speach recognition. *Proceeding of the IEEE, 77(2), p. 257-286*, February 1989.

[38] A. Schein and L. Ungar. Active learning for logistic regression: An evaluation. *Machine Learning*, 68(3), October 2007.

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

[40] C. Sminchisescu, A. Kanaujia, Z. Li, and D. Metaxus. Conditional models for contextual human motion recognition. In *In Proceedings of the International Conference on Computer Vision, (ICCV)*, 2005.

[41] C. Sutton, M. Sindelar, and A. McCallum. CIIR technical report ir-402, university of massachusetts. In *Feature Bagging: Preventing Weight Undertraining in Structured Discriminative Learning*, 2005.

[42] A. Torralba, K. P. Murphy, and W. T. Freeman. Contextual models for object detection using boosted random fields. In *Advances in Neural Information Processing Systems (NIPS)*, 2004.

[43] D. L. Vail, J. D. Lafferty, and M. M. Veloso. Feature selection in conditional random fields for activity recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007.

[44] A. J. Viterbi. Error bounds for convolutional codes and asymptotically optimum decoding algorithm. In *IEEE Transaction*, 1967.

[45] Y. Wang and Q. Ji. A dynamic conditional random field model for object segmentation in image sequences. In *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005.

[46] Y. Weiss. Comparing the mean field method and belief propagation for approximate inference in MRFs. *Advanced Mean Field Methods*, 2001.

[47] D. H. Wilson and C. Atkeson. Simultaneous tracking and activity recognition (star) using many anonymous, binary sensors. In *Proceedings of PERVASIVE*, 2005.

[48] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Understanding belief propogation and its generalizations. *Exploring Artificial Intelligence in the New Millenium*, 2001.

[49] J.S. Yedidia, W.T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *IEEE Transactions on Information Theory*, 51(7), 2005.

[50] A. Zhu, Z. Ghahramani, and J. Lafferty. Semi-supervised learning using Gaussian fields and harmonic functions. In *Proc. of the International Conference on Machine Learning (ICML)*, 2003.

[51] X. Zhu, J. Lafferty, and Z. Ghahramani. In *Proc. of the ICML 2003 workshop on The Continuum from Labeled to Unlabeled Data in Machine Learning and Data Mining.*

Chapter 7

Appendix A: Derivation of sVEB

In this section, we show how we derived the equations for w_i and z_i (equation 4.7):

$$L_F \equiv L_{sVEB} = L_{VEB} - \alpha H_{emp} \tag{7.1}$$

$$= \sum_{i=1}^{N} \log p(y_i|\mathbf{ve_i}) + \alpha \sum_{i=N+1}^{M} \sum_{y_i'} p(y_i'|\mathbf{ve_i}) \log p(y_i'|\mathbf{ve_i}) \tag{7.2}$$

As in LogitBoost, the likelihood function L_F is maximized by learning an ensemble of weak learners. We start with an empty ensemble $F = 0$ and iteratively add the next best weak learner, f_t, by computing the Newton update $\frac{g}{H}$, where g and H are the first and second derivative respectively of L_F with respect to $f(\mathbf{ve_i}, y_i)$.

$$F(\mathbf{ve_i}, y_i)) \leftarrow F(\mathbf{ve_i}, y_i) - \frac{g}{H}, \tag{7.3}$$

$$where\ g = \frac{\partial L_{F+f}}{\partial f}|_{f=0} \tag{7.4}$$

$$and\ H = \frac{\partial^2 L_{F+f}}{\partial f^2}|_{f=0} \tag{7.5}$$

$$\Rightarrow g = \sum_{i=1}^{N} 2(2y_i - 1)(1 - p(y_i|\mathbf{ve_i})) + \tag{7.6}$$

$$\alpha \sum_{i=N+1}^{M} \sum_{y_i'} [2(2y_i' - 1)(1 - p(y_i'|\mathbf{ve_i}))p(y_i'|\mathbf{ve_i})(1 - \log p(y_i'|\mathbf{ve_i}))]$$

$$H = -\sum_{i=1}^{N} 4p(y_i|\mathbf{ve_i})(1 - p(y_i|\mathbf{ve_i}))(2y_i - 1)^2 + \tag{7.7}$$

$$\alpha^2 \sum_{i=N+1}^{M} \sum_{y_i'} 4(2y_i' - 1)^2(1 - p(y_i'|\mathbf{ve_i}))[p(y_i'|\mathbf{ve_i})(1 - p(y_i'|\mathbf{ve_i})) + \log p(y_i'|\mathbf{ve_i})]$$

$$\Rightarrow \quad F \leftarrow F + \frac{\sum_{i=1}^{N} z_i w_i + \sum_{i=N+1}^{M} \sum_{y_i'} z_i w_i}{\sum_{i=1}^{N} w_i + \sum_{i=N+1}^{M} \sum_{y_i'} w_i} \tag{7.8}$$

where

$$z_i = \begin{cases} \frac{y_i - 0.5}{p(y_i|\mathbf{ve_i})} & \text{if } 1 \le i \le N \quad \text{eq. (3.15)} \\ \\ \frac{(y_i' - 0.5)p(y_i'|\mathbf{ve_i})(1 - \log p(y_i'|\mathbf{ve_i}))}{\alpha[p(y_i'|\mathbf{ve_i})(1 - p(y_i'|\mathbf{ve_i})) + \log p(y_i'|\mathbf{ve_i})]} & \text{if } N < i \le M \quad \text{eq. (4.7)} \end{cases} \tag{7.9}$$

and

$$w_i = \begin{cases} p(y_i|\mathbf{ve_i})(1 - p(y_i|\mathbf{ve_i})) \;\; eq.(3.15) & \text{if } 1 \le i \le N \\ \\ \alpha^2(1 - p(y_i'|\mathbf{ve_i}))[p(y_i'|\mathbf{ve_i})(1 - p(y_i'|\mathbf{ve_i})) \\ + \log p(y_i'|\mathbf{ve_i})] \;\; eq.(4.7) & \text{if } N < i \le M \end{cases} \tag{7.10}$$

At iteration t we get the best weak learner, f_t, by solving the WLSE problem in eq. (4.6).

Chapter 8

Appendix B: Per-class Average Accuracies

This appendix demonstrates the extended classification results of supervised methods of chapter 3 and semi-supervised methods described in chapter 4.

Class	*Average Accuracy (%)*		
	MAP+all obs	MAP+Boost	VEB
Sitting	74.1 ± 5.2	84.8 ± 3.3	86.8 ± 5.1
Standing	68.7 ± 4.6	84.2 ± 3.1	89.6 ± 5.2
Walking	84.5 ± 3.1	89.7 ± 2.4	93.7 ± 3.8
Walking up stairs	74.2 ± 2.8	86.5 ± 1.9	95.3 ± 3.4
Walking-down-stairs	80.9 ± 2.5	85.3 ± 2.3	92.5 ± 2.4
Riding elevator down	76.3 ± 2.7	83.6 ± 1.5	95.1 ± 3.1
Riding elevator up	79.4 ± 2.2	84.7 ± 1.7	96.8 ± 1.9
Brushing teeth	84 ± 1.8	90.3 ± 1.3	98.2 ± 1.7

Table 8.1: Average accuracy per class and ± 95% confidence interval of supervised algorithms on activity datasets 1.

Class	*Average Accuracy (%)*		
	MAP+all obs	MAP+Boost	VEB
Computer usage	88.1± 3.6	88.6 ± 3.4	92.1 ± 4.4
Having meal	83.7 ± 2.5	89.2 ± 2.9	96.9 ± 4.9
Meeting	84.3 ± 4.2	85.1 ± 4.5	88.6 ± 5.6
Watching TV	85.4 ± 2.9	86 ± 2.7	92.7 ± 4
Sleeping	87.2 ± 2.1	87.9 ± 2.3	97.4 ± 3.9

Table 8.2: Average accuracy per class and ± 95% confidence interval of supervised algorithms on activity dataset 2.

Class	*Average Accuracy (%)*		
	sMAP+all obs	sMAP+Boost	sVEB
Sitting	75.9 ± 3.5	82.5 ± 3.5	82.5± 3.8
Standing	68.1 ± 4.2	77.4± 5.2	77.4 ± 5.7
Walking	75 ± 4.8	79.6 ± 4.6	79.6 ± 4.1
Walking up stairs	76.4 ± 2.1	84.2 ± 2.4	84.2± 2.9
Walking-down-stairs	78.6 ± 2.2	83.9 ± 2	83.9 ± 2.5
Riding elevator down	75.2 ± 2.4	84.4 ± 2.6	84.4 ± 3.1
Riding elevator up	78.7 ± 2.1	85.1 ± 2.9	85.1 ± 3.2
Brushing teeth	83.1 ± 1.5	89.8 ± 1.4	89.8 ± 2.3

Table 8.3: Average accuracy per class and ± 95% confidence interval of semi-supervised algorithms on activity dataset 1.

Class	*Average Accuracy (%)*		
	sMAP+all obs	sMAP+Boost	sVEB
Computer usage	75.4 ± 3.1	78.8 ± 2.5	81.6 ± 3.9
Having meal	72.9 ± 2.5	74.3 ± 3.2	90.9 ± 4.1
Meeting	71.3± 5.5	70.1 ± 4.8	84.5 ± 4.5
Watching TV	74.1 ± 4.3	73.9 ± 3.1	83.7 ± 5.1
Sleeping	80.6 ± 2.9	85.6 ± 2.4	94.9 ± 2.6

Table 8.4: Average accuracy per class and ± 95% confidence interval of semi-supervised algorithms on activity dataset 2.