2018

Machine Learning Methods for Activity Detection in Wearable Sensor Data Streams

Roy Adams

MACHINE LEARNING METHODS FOR ACTIVITY DETECTION IN WEARABLE SENSOR DATA STREAMS

A Dissertation Presented

by

ROY ADAMS

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2018

College of Information and Computer Science

MACHINE LEARNING METHODS FOR ACTIVITY DETECTION IN WEARABLE SENSOR DATA STREAMS

A Dissertation Presented

by

ROY ADAMS

Approved as to style and content by:

Benjamin Marlin, Chair

Andrew McCallum, Member

Deepak Ganesan, Member

Patrick Flaherty, Member

James Allan, Chair
College of Information and Computer Science

ACKNOWLEDGMENTS

Very little worth doing was ever done in a vacuum and this thesis is no exception. I have received the support of many people over the past six years and fortunately, I can completely repay every single one of them by including them in my acknowledgments. One cannot overstate the importance of the advisor/advisee relationship to one's success and sanity. Fortunately, my Ph.D. is complete and my sanity is intact due in large part to my advisor, Ben Marlin. Ben is among the most pragmatic and quietly intelligent people I know. He is dedicated to teaching, patient in his advising, and recognizes the diverse needs of his advisees. I am immensely fortunate to have had him as an advisor.

I am indebted to the many great professors I have come into contact with at UMass and as part of MD2K including my committee Deepak Ganesan, Andrew McCallum, and Patrick Flaherty, whose reading group was a highlight of my time here. I have also bene

tted from the conversations with and general proximity to all of the great students in the MLDS lab including Juston, Tao, Malai, Steve, and Garrett, as well as the professors who helped create the lab, Dan Sheldon and Hanna Wallach. I would like to extend a special thanks to Shannon, whose ability to handle nearly any task we threw at them is truly humbling.

I have made a number of truly great friends at UMass including Pinar, Kevin, David, Kristen, and the Propagators: James, Luis, Pat, and Aaron. I am especially lucky to have met this last Propagator. Without him, I would likely have gotten more sleep, but would be much the worse for it.

Is it a cliche to say that I could not have done this without my parents? Undoubtedly, but it is nevertheless true. If I can have anything resembling the scienti

c careers they have had, I will consider myself lucky. They have continuously pushed me to

nd what makes me happy and I will be forever grateful.

Finally, as of this writing, it is exactly one month and one day until I marry Paige Seegan. Three weeks after that, she will receive her own Ph.D. It is hard to express how much I admire this woman, and yet, custom dictates that I try. She is smart and kind and funny and has more empathy than I will ever possess. She cares deeply. Not just about her work or her friends, but about nearly everyone who enters her life. She puts me at my ease and for that, I am a lucky man.

This work was partially supported by the National Institutes of Health under award 1U54EB020404, and the National Science Foundation under award IIS-1350522.

ABSTRACT

MACHINE LEARNING METHODS FOR ACTIVITY DETECTION IN WEARABLE SENSOR DATA STREAMS

SEPTEMBER 2018

ROY ADAMS

B.S., UNIVERSITY OF CALIFORNIA DAVIS

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Benjamin Marlin

Wearable wireless sensors have the potential for transformative impact on the fields of health and behavioral science. Recent advances in wearable sensor technology have made it possible to simultaneously collect multiple streams of physiological and context data from individuals in natural environments; however, extracting reliable high-level inferences from these raw data streams remains a key data analysis challenge. In this dissertation, we address three challenges that arise when trying to perform activity detection from wearable sensor streams. First, we address the challenge of learning from small amounts of noisy data by proposing a class of conditional random field models for activity detection. We apply this model class to three different activity detection problems, improving performance in all three when compared with standard independent and structured models. Second, we address the challenge of inferring activities from long input sequences by evaluating strategies for pruning

the inference dynamic programs used in structured prediction models. We apply these strategies to the proposed structured activity detection models resulting in inference speedups ranging from 66x to 257x with little to no decrease in predictive performance. Finally, we address the challenge of learning from imprecise annotations by proposing a weak supervision framework for learning discrete-time detection models from imprecise continuous-time observations. We apply this framework to both independent and structured models and demonstrate improved performance over weak supervision baselines.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

A small number of behaviors including physical inactivity, poor diet, tobacco use, and alcohol consumption are key risk factors in a wide array of chronic conditions including obesity, cancer, diabetes and cardiovascular disease [37, 40, 11]. These behaviors have traditionally been studied using self-report data; however, self-report has well-known limitations including data sparsity, recall bias, and high burden on study subjects [62]. The emerging field of mobile health (mHealth) seeks to supplement and eventually replace the use of self-report data with continuously recorded physiological and activity-related data streams collected using wearable sensors. While mHealth technologies have the potential to yield novel insights into health and behavior, significant data analysis challenges must first be overcome [28].

In this dissertation, we identify machine learning and data analysis challenges that arise in many mHealth settings and propose new models and methods for overcoming them. These challenges include data scarcity and noise, the high cost of acquiring annotated data, and the need to process high volumes of densely sampled signals. A core component of many mHealth applications is the ability to infer from wearable sensor data when a subject is engaging in the behaviors we want to study or treat. We call this the **activity detection problem** and it is the focus of this dissertation.

Enabling accurate mHealth activity detection has the potential for major impact on behavioral science. A search of the NIH grant database[1] for current projects whose

[1] `https://projectreporter.nih.gov/reporter.cfm`

descriptions include the phrase "ecological momentary assessment" (a common self-report methodology [62]) returns over 200 projects. These projects include studies of smoking, drug use, stress, post-traumatic stress disorder, alcohol consumption, food choices, and medication adherence, among many other behaviors and conditions. The ability to accurately infer behaviors of interest as well as contextual information about when and where these behaviors occur will allow us to to study these behaviors at a level of detail that is not currently possible [28]. Further, such detection capabilities open the possibility for novel types of interventions that do not require direct interaction with a health care provider and thus can be delivered at scale [42, 50].

Our hope is that the techniques developed in this thesis can be incorporated by behavioral scientists into the standard suite of tools for studying behavioral health, improving current study designs and enabling new ones. The following are three examples of the ways activity detection may be used in downstream mHealth tasks:

1. **Monitoring:** For many conditions, it is valuable to simply monitor and record behaviors of interest. For example, in the treatment of chronic conditions such as obesity, tracking food consumption is an effective part a behavioral intervention plan [6]. In this case, wearable sensors may supplant or supplement traditional monitoring methods such as journaling.

2. **Causal inference:** Behavioral scientists are concerned with understanding the causal mechanisms underlying behaviors and conditions such as addiction. To identify these mechanisms it is necessary to detect behaviors of interest and place them in a broader context. Wearable sensors may be used to both identify the behavior of interest, as well as record relevant contextual information.

3. **Intervention:** An emerging technology for delivering behavioral interventions for chronic conditions is Just-In-Time adaptive interventions [42, 50]. Just-in-Time interventions use signals from wearable sensors to determine the optimal

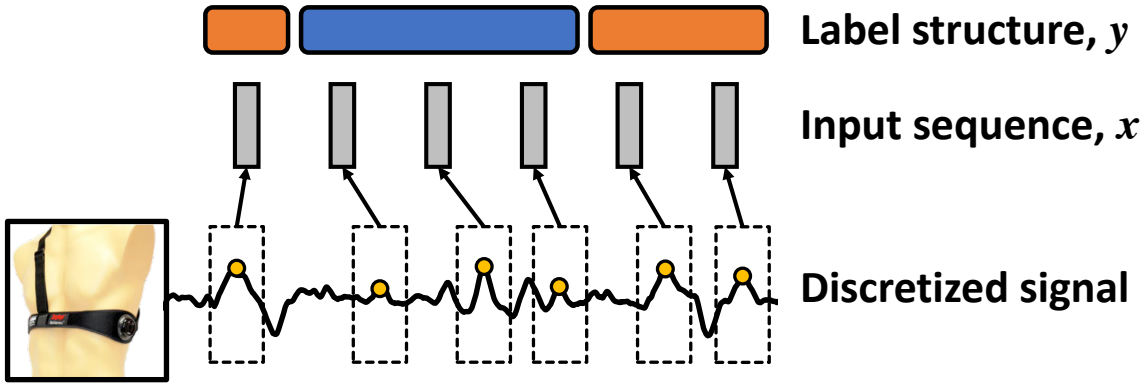


Figure 1.1: An illustration of the activity detection problem. On the bottom is a sample sensor signal that has been discretized using a peak detection method. Above that is the input sequence of feature vectors, $\mathbf{x}$. On top is a possible label structure for this input, $\mathbf{y}$. In this case, the label structure is a segmentation where blue and orange indicate positive and negative segment labels respectively.

time to deliver an intervention using a mobile device such as a smart phone. For example, a Just-In-Time intervention for insufficient exercise might use an accelerometer to detect physical activity and prompt the user to get up and move if they have been too sedentary. Accurate behavioral detection is necessary in this case to identify the right time to deliver an intervention.

In the remainder of this chapter, we define the detection problem as a supervised learning problem (Section 1.1) and outline the contributions of this dissertation (Section 1.2).

1.1 Problem Definition and Notation

We treat activity detection as a supervised learning problem. That is, we assume that we are given a set of signals where the occurrences of the activity of interest have been annotated and we would like to learn a function that takes a new signal and locates occurrences of the same activity. To formalize this problem, we must define the notation and describe the assumptions we will make about the input signals. We assume that the data is organized into a set of N **sessions** each corresponding to a

time series generated from one or more wearable sensors. Each time series may be multivariate and sessions may vary in length. Separate sessions may correspond to data from different subjects or to data from the same subject collected at different times. We assume that each session n has been discretized into a sequence of L_n discrete **instances** and that a feature vector $\mathbf{x}_{ni} \in \mathbb{R}^D$ has been extracted from a window around each instance i. Further, each instance i in session n is associated with a timestamp t_{ni} which may correspond to any point of interest associated with instance i. We refer to the complete sequence of feature vectors $\mathbf{x}_n = \{\mathbf{x}_{ni}\}_{i=1}^{L_n}$ as the **input sequence** and the complete sequence of timestamps $\mathbf{t}_n = \{t_{ni}\}_{i=1}^{L_n}$ as the **timestamp sequence**.

Finally, each session n is associated with a label structure $\mathbf{y}_n \in \mathcal{Y}_n$ that denotes the locations of activities in the sequence. The support set $\mathcal{Y}_n$ depends on the application at hand and may depend on the size of the session n. For example, some problems require only that each instance be labeled as positive or negative, in which case $\mathcal{Y}_n = \{0, 1\}^{L_n}$.

Alternatively, some applications require a segmentation of the input sequence. We define a **segmentation** as a sequence of segments $\mathbf{y}_n = \{y_{ns}\}_{s=1}^{S_n}$ where each segment $y_{ns} = (c_{ns}, j_{ns}, k_{ns})$ is a tuple containing a segment label $c_{ns} \in \mathcal{C}$, a start index $j_{ns} \in \{1, ..., L_n\}$, and an end index $k_{ns} \in \{1, ..., L_n\}$. In this case, $\mathcal{Y}_n$ will include all possible segmentations of a length L_n sequence. To ensure that a segmentation does not contain overlapping segments and that the entire input sequence is covered, we constrain the set of segmentations $\mathcal{Y}_n$ such that $k_{n1} = 1$, $k_{nS_n} = L_n$, and $k_{ns} = j_{ns+1}$ for all $1 \leq s \leq S_n - 1$.

With these definitions, we can formalize activity detection as a supervised learning problem where our goal is to learn a function $f : \mathbb{R}^{D \times L_n} \times \mathbb{R}^{L_n} \rightarrow \mathcal{Y}_n$ that maps a feature sequence and a timestamp sequence to a label structure representing the activity of interest. Figure 1.1 shows an example of this problem formulation where

peak detection is used to discretize the input signal and the label structure is a labeled segmentation of the input sequence.

1.2 Contributions

This dissertation makes three main contributions that address challenges faced when applying supervised learning methods to the problem of mHealth activity detection: *structured prediction models for learning from small amounts of noisy data*, *inference acceleration methods for performing structured prediction on long input sequences*, and *weak supervision frameworks for learning in the presence of temporal label imprecision.*

1. **Structured prediction for learning from small amounts of noisy data:** Data gathered from wearable sensors is subject to many confounding noise sources such as sensor movement and signal dropout. Further, gathering reliable labeled data may require expensive protocols which leads to scarcity of labeled data. Because of these issues, generic supervised learning algorithms can underperform in this domain. One way to improve over generic models when data is limited is to encode domain knowledge into the model.

   Traditionally, this has been done using feature engineering which has been successfully applied in the mHealth space a number of times [1, 56, 3, 74]; however, hand-engineered features are limited because they are typically only applicable to a single input modality. Further, it is still common practice to use generic instance labeling models on top of these features and to use ad-hoc methods to perform segmentation on top of predictions from these models. Instead, we use structured prediction to encode domain knowledge into the model itself and jointly perform segmentation and instance labeling.

In Chapter 3 we present a family of conditional random field-based models for the problem of *heterogeneous segmentation*, or segmentation where the instances within a segment need not share the same label. While similar models have been developed for specific applications [64], we present a general form models of this type. We apply versions of this model family to three different mHealth domains: conversation detection (Section 3.3), eating and smoking detection (Section 3.4), and electrocardiogram morphology extraction (Section 3.5). This represents the first application of structured prediction to many of these problems. We improve detection performance in all three domains compared with both independent instance classifiers and generic structured models.

2. **Inference acceleration for performing structured prediction on long input sequences:** The application of structured models to mHealth data requires that we perform inference on very long unlabeled sequences. In these cases, even inference algorithms with polynomial complexity in the length of the input sequence, such as those presented in Chapter 3, may be too slow in practice. One strategy for improving inference runtimes is to constrain the support set of label structures [78]. In Section 3.6, we apply this strategy to structured segmentation models. We present two such approaches including static constraints on the label set based on domain knowledge, and learned constraints that use an independent classifier to constrain the support set of a structured model. Such acceleration strategies have been heavily developed in the natural language processing literature. In this work we show how they can be applied to mHealth activity detection problems. In particular, we adapt the approach used by [5] for parsing to the problem of segmentation. We apply this approach to two mHealth segmentation problems resulting in up to 257x speedups in inference time with no significant drop in prediction accuracy.

3. **Weak supervision for learning in the presence of temporal label imprecision:** Gathering high-quality labeled data for supervised learning of activity detection models requires expensive, high-fidelity observation mechanisms and time-consuming annotation protocols. It is often possible to acquire annotations using lower cost methods, such as self-report, but these annotations may be imprecise. As we might expect, applying supervised learning methods directly to imprecise labels leads to lower performance as these methods have no way to distinguish errors in the training data. Techniques for learning from data with label imprecision have been developed and applied to domains such as image classification [39] and species distribution modeling [84], allowing for the estimation of models from higher volumes of lower-quality data.

   In Chapter 4, we present a new type of label imprecision that arrises when annotations for model estimation are provided as continuous timestamps. This problem is not naturally solved by existing approaches to weakly supervised learning, so we introduce weak supervision framework that allows estimation of discrete-time activity detection models from continuous-time, temporally imprecise annotations. We first develop this framework for independent detection models (Section 4.2) resulting in approximately a 0.06 improvement in F_1 over a model trained on ground truth labels. We further develop this framework for structured prediction models (Section 4.3) resulting in approximately a 0.02 loss in accuracy relative to a model trained on hand aligned labels. Finally, we show how this framework can be used to integrate imprecise observations with sensor data at test time (Section 4.4), allowing for inferences that combine both data sources.

1.3 Outline

The remainder of this dissertation is organized as follows: We begin by presenting background material on conditional random fields and weakly supervised learning in Chapter 2. In Section 2.3, we describe seven datasets that we use throughout this dissertation. In Chapter 3, we present a family of conditional random field models for joint sequence labeling and segmentation and apply this model family to three mHealth applications. We then present inference acceleration methods for improving inference times on long sequences. Finally, in Chapter 4, we present weak supervision frameworks for learning independent and structured activity detection models from temporally imprecise labels.

CHAPTER 2

BACKGROUND AND RELATED WORK

In this chapter, we present background material and related work used throughout the remainder of this dissertation. In Section 2.1 we discuss the conditional random field (CRF) model family and present a selection of standard CRF models. In Section 2.2 we discuss various weakly supervised learning frameworks for independent classification and structured prediction. Finally, in Section 2.3, we describe the datasets used throughout the remainder of this dissertation.

2.1 Structured Prediction

A typical first approach to the mHealth event detection problem is to treat each instance independently [1, 56, 47, 68, 38]. That is, a classifier is learned that maps each instance $\mathbf{x}_i$ to a discrete label y_i. For example, if our goal is to detect eating, we may learn an independent classifier to label each instance as an eating gesture or not. When it is necessary to detect more complete activities, such as a complete meal, it is common to use post-hoc segmentation rules. For example, we may decide that any two bites of food within five minutes of each other belong to the same eating activity.

This approach fails to leverage domain knowledge we may have about the application. For example, people tend to pause between bites of food. Encoding this knowledge into our model may allow us to generalize better from small amounts of noisy data, but it breaks the independence between instances. Further, by performing segmentation on top of fixed instance-level predictions, errors made by the independent instance classifier may propagate and cause errors in the post-hoc segmentation.

Information about what is a likely segmentation cannot flow back to the instance-level classifier. In this section, we present background on structured prediction, which can be used to model known structures in activities of interest and avoid propagating errors in prediction pipelines.

Structured prediction is a sub-area of supervised learning focused on the simultaneous prediction of multiple related label variables. Specifically, given a set of feature variables $\mathbf{X} = \{\mathbf{X}_i\}_{i=1}^{M} \in \mathcal{X}$ and a set of label variables $\mathbf{Y} = \{Y_i\}_{i=1}^{L} \in \mathcal{Y}$, the goal of structured prediction is to learn a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ that maps an assignment of $\mathbf{X}$ to an assignment of $\mathbf{Y}$. The set of label variables is often, though not always, used to represent a specific structure such as a label sequence, a segmentation, or a parse tree.

Structured prediction methods have been developed in a number of application areas, with computer vision and natural language processing playing a particularly pivotal role. We will draw on techniques from both fields in this work. In particular, we focus on the conditional random field (CRF) family of models [30]. In the following sections, we describe the CRF formalism and its application to mHealth activity detection problems.

2.1.1 Conditional Random Fields

CRFs are a sub-class of probabilistic graphical models [25] that generalize log-linear probabilistic classifiers like logistic regression [23] to the case of structured prediction. A CRF defines a conditional distribution over a set of L output variables $\mathbf{Y} = \{Y_i\}_{i=1}^{L}$ given a corresponding set of M feature variables $\mathbf{X} = \{\mathbf{X}_i\}_{j=1}^{M}$. We assume that each feature variable $\mathbf{X}_j \in \mathbb{R}^D$ is a D dimensional real vector and that each label variable Y_i takes a value in a set $\mathcal{Y}_i$; however, there may be additional constraints on the set of possible joint configurations, denoted by $\mathcal{Y}$. For example, if the structure we are trying to predict is a segmentation of a sequence, then it is

necessary to constrain $\mathcal{Y}$ to contain only segmentations that cover the entire sequence with no overlapping segments.

A general log-linear CRF is defined through a linear energy function that takes the form of a weighted sum of K feature functions $\mathbf{f}_k$ involving values of $\mathbf{Y}$ and $\mathbf{X}$:

$$\langle \theta, \mathbf{f}(\mathbf{x}, \mathbf{y}) \rangle = \sum_{k=1}^{K} \theta_k \mathbf{f}_k(\mathbf{y}, \mathbf{x}) \tag{2.1}$$

where $\langle u, v \rangle$ is the inner product of u and v. These feature functions are typically sparse in the sense that they involve few label and feature variables. The set of label and feature variables referenced in function $\mathbf{f}_k$ is referred to as its **scope** S_k. $\exp(\langle \theta_k, \mathbf{f}_k(\mathbf{y}, \mathbf{x}) \rangle)$ is commonly refered to as a **factor**.

Importantly, this energy function can be represented using an undirected graph $G = (V, E)$ where the set of vertices $V = \mathbf{Y} \cup \mathbf{X}$ is the set of all variables in the model. For any pair of variables u and v there is an edge in the graph if both u and v are in the scope of the same feature function: $E = \{(u, v) \mid u, v \in V, \exists\, k \text{ s.t. } u, v \in S_k\}$. Variables in the model obey a Markov property with respect to G. That is, a variable u is independent of all other variables in the graph given the variables in its neighborhood, otherwise known as the *Markov blanket* of variable u.

The probability $P_\theta(\mathbf{y}|\mathbf{x})$ of a configuration of all of the label variables $\mathbf{y} = \{y_i\}_{i=1}^{L}$ conditioned on the observed feature variables $\mathbf{x} = \{\mathbf{x}_j\}_{j=1}^{M}$ is given below. $Z_\theta(\mathbf{x})$ is referred to as the *partition function* and is the normalization term of the probability distribution.

$$P_\theta(\mathbf{y}|\mathbf{x}) = \frac{\exp(\langle \theta, \mathbf{f}(\mathbf{x}, \mathbf{y}) \rangle)}{Z_\theta(\mathbf{x})} \tag{2.2}$$

$$Z_\theta(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{Y}} \exp(\langle \theta, \mathbf{f}(\mathbf{x}, \mathbf{y}) \rangle) \tag{2.3}$$

We will refer to any log-linear conditional model as a CRF even if the model was not trained as a probabilistic model (i.e. using a likelihood). This includes

conditional graphical models trained to minimize various empirical risk functions such as structured support vector machines [67, 71] which are trained to minimize the multiclass hinge-loss and models trained to minimize expected negative marginal probabilities [12].

2.1.1.1 Inference in CRFs

Two common inference problems arise when using CRFs: maximum a posteriori (MAP) inference and marginal inference. **MAP inference** is the problem of finding the highest probability setting of $\mathbf{y}$ given a value for $\mathbf{x}$ and a setting of the parameters θ. MAP inference is commonly used at test time to predict unseen values of $\mathbf{y}$ and is a necessary sub-routine of maximum margin learning (discussed below). Formally, the MAP inference problem is defined as follows

$$\mathbf{y}^* \in \arg\max_{\mathbf{y}\in\mathcal{Y}} P_\theta(\mathbf{y}|\mathbf{x}) \tag{2.4}$$

$$= \arg\max_{\mathbf{y}\in\mathcal{Y}} \log P_\theta(\mathbf{y}|\mathbf{x}) \tag{2.5}$$

$$= \arg\max_{\mathbf{y}\in\mathcal{Y}} \langle\theta, \mathbf{f}(\mathbf{x},\mathbf{y})\rangle - \log Z_\theta(\mathbf{x}) \tag{2.6}$$

$$= \arg\max_{\mathbf{y}\in\mathcal{Y}} \langle\theta, \mathbf{f}(\mathbf{x},\mathbf{y})\rangle. \tag{2.7}$$

The last simplification can be made because the partition function does not depend on the value of $\mathbf{y}$. If $\mathcal{Y}_i$ is a discrete set for all i, then solving the MAP inference problem is equivalent to solving an integer linear program which is, in general, intractable. As a result, much of the work on CRFs has focused on models for which this problem can be solved tractably. In particular, for many models of interest, $\langle\theta, \mathbf{f}(\mathbf{x},\mathbf{y})\rangle$ factorizes in such a way that MAP inference can be performed using a dynamic program with polynomial complexity.

Alternatively, **marginal inference** is the problem of calculating the expected value of the feature functions for given values of $\mathbf{x}$ and θ. Formally, the the expected feature functions μ are defined as

$$\mu = \mathbb{E}_{P_\theta(\mathbf{y}|\mathbf{x})}[\mathbf{f}(\mathbf{x}, \mathbf{y})] \tag{2.8}$$

$$= \sum_{\mathbf{y} \in \mathcal{Y}} P_\theta(\mathbf{y}|\mathbf{x}) \mathbf{f}(\mathbf{x}, \mathbf{y}). \tag{2.9}$$

In many models, each feature function $\mathbf{f}_k$ depends on $\mathbf{Y}$ only through indicator functions on its scope S_k, $\mathbb{I}[\mathbf{y}_{S_k} = \mathbf{v}]$[1] where $\mathbf{y}_{S_k}$ is a setting of the variables in S_k. For example, the feature function for logistic regression is $f(\mathbf{x}, y) = \mathbf{x}\mathbb{I}[y = 1]$. In these cases, calculating the marginals is equivalent to calculating marginal probabilities because the expectation of an indicator function of a discrete variable is a probability. In the logistic regression case, $\mathbb{E}_{P_\theta(\mathbf{y}|\mathbf{x})}[\mathbf{f}(\mathbf{x}, \mathbf{y})] = \mathbf{x}P(Y = 1|\mathbf{x})$.

Because of the log-linear form of CRF models we have the following useful property:

$$\nabla_\theta \log Z_\theta(\mathbf{x}) = \sum_{\mathbf{y} \in \mathcal{Y}} \frac{\nabla_\theta \exp(\langle \theta, \mathbf{f}(\mathbf{x}, \mathbf{y}) \rangle)}{Z_\theta(\mathbf{x})} \tag{2.10}$$

$$= \sum_{\mathbf{y} \in \mathcal{Y}} P_\theta(\mathbf{y}|\mathbf{x}) \mathbf{f}(\mathbf{x}, \mathbf{y}) = \mu. \tag{2.11}$$

That is, the marginals are equal to the gradient of the log partition function with respect to the parameters θ. Assuming we can tractably calculate the log partition function, we can also tractably calculate the marginals using automatic differentiation [4]. Unfortunately, calculating the partition function involves a sum over a set of size $\mathcal{O}(\prod_i |\mathcal{Y}_i|)$ and so, like the MAP inference problem, this computation is intractable in the general case. As with the MAP inference problem, many models of interest allow

[1]The notation $\mathbb{I}[s]$ is the indicator function where $\mathbb{I}[s] = 1$ if the statement s is true and $\mathbb{I}[s] = 0$ if the statement s is false.

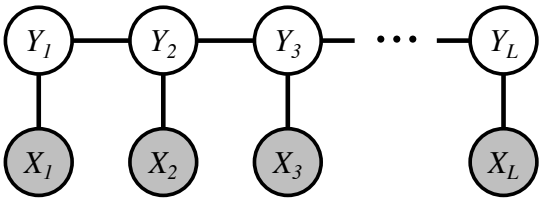


Figure 2.1: Graphical model for a linear chain conditional random field (CRF-LC). Grey nodes indicate variables that are always observed.

factorizations of this sum that result in tractable dynamic programming solutions. We discuss some important examples of this below.

2.1.2 Standard CRFs

In this section, we present three standard CRF models which are related to the models presented in Chapter 3.

2.1.2.1 Linear Chain CRFs

The linear-chain CRF gives a distribution over a label sequence given a feature sequence of the same length. The graphical model for the linear-chain CRF model is shown in Figure 2.1. The linear-chain CRF model defines a joint distribution over a sequence of discrete label variables $\mathbf{Y} = \{Y_i\}_{i=1}^{L}$ given a corresponding sequence of feature vectors $\mathbf{X} = \{\mathbf{X}_i\}_{i=1}^{L}$. Each label variable Y_i is assumed to take a value from the label set $\mathcal{C}$. This model incorporates the feature vectors via a set of pairwise feature functions $\mathbf{f}_{iv}^{(1)}(\mathbf{y}, \mathbf{x}) = \mathbb{I}[y_i = v]\mathbf{x}_i$. Consecutive label variables are tied together by a set of pairwise feature functions $\mathbf{f}_{ivu}^{(2)} = [y_i = v][y_{i+1} = u]$. Importantly, this model obeys a first order Markov property on the sequence, that is, $Y_i \perp Y_{i+k}|Y_{i+1},\ \forall\, k < 1$. MAP inference can be performed in this model using the Viterbi algorithm [79] and marginal inference can be performed using the closely related forward-backward algorithm [25]. Both algorithms are dynamic programming algorithms that have complexity $\mathcal{O}(|\mathcal{C}|^2 L)$.

2.1.2.2 Semi-Markov CRFs

For many applications, the Markov assumption made in the linear-chain CRF model ($Y_i \perp Y_j | Y_{i-1}, \forall j < i-1$) does not hold. Semi-Markov models generalize Markov models by defining a segmentation of the input sequence and allowing non-Markov dependencies within segments [57]. The semi-Markov CRF defines a distribution over such segmentations given an input sequence $\mathbf{x} = \{\mathbf{x}_i\}_{i=1}^{L}$. The semi-Markov CRF represents a segmentation as a sequence of segment variables $\mathbf{Y} = \{Y_s\}_{s=1}^{S}$ where a segment $Y_s = (C_s, J_s, K_s)$ is a tuple containing a label $C_s \in \mathcal{C}$, a start position $J_s \in \{1, ..., L\}$, and an end position $K_s \in \{1, ..., L\}$.

To ensure only valid segmentations, we constrain the set of segmentations $\mathcal{Y}$ such that $K_1 = 1$, $K_S = L$, and $K_s = J_{s+1}$ for all $1 \leq s \leq S-1$. The semi-Markov CRF is defined by a feature function $\mathbf{f}(Y_s, C_{s-1}, \mathbf{x})$ which is applied to each segment Y_s for all $s > 1$. This function may depend on the segment Y_s, the label of the previous segment C_{s-1}, and the complete input sequence $\mathbf{x}$. The function $\mathbf{f}$ maps these inputs to a length F feature vector. Given a parameter vector $\theta \in \mathbb{R}^F$, the distribution over segmentations is given by:

$$P_\theta(\mathbf{y}|\mathbf{x}) \propto \prod_s \exp\left(\langle \theta, \mathbf{f}(y_s, c_{s-1}, \mathbf{x}) \rangle\right). \tag{2.12}$$

Both MAP and marginal inference can be performed in the semi-Markov CRF using dynamic programs with complexity $\mathcal{O}(|\mathcal{C}|^2 L^2)$. We present some details of this algorithm here, as it is closely related to the inference algorithm presented in Section 3.2.1. The dynamic program for calculating the partition function $Z_\theta(\mathbf{x})$ is defined by the following recursion:

$$\alpha(k, c) = \sum_{j=1}^{k} \sum_{c' \in \mathcal{C}} \exp(\langle \theta, \mathbf{f}((c, j, k), c', \mathbf{x}) \rangle)\, \alpha(j-1, c') \tag{2.13}$$

with the base case $\alpha(0, c) = 1$ for all $c \in \mathcal{C}$. The dynamic programming table α has the interpretation that $\alpha(k, c)$ is the sum over all segmentations of the subsequence from position 0 through k constrained so that the last segment has label c. Given this interpretation, the full partition function is given by $Z_\theta(\mathbf{x}) = \sum_{c \in \mathcal{C}} \alpha(L, c)$.

2.1.2.3 Context Free Grammar CRFs

The context free grammar (CFG) CRF generalizes a number of other CRF models (including the linear-chain CRF and the semi-Markov CRF) and defines a distribution over the types of tree structures commonly used to describe language. Specifically, given a CFG and an input sequence, a CFG CRF defines a distribution over all possible parses of the sequence using the rules defined by the context free grammar [65]. We will first define context free grammars and then describe the CFG CRF model.

A CFG is a mathematical structure that compactly describes the set of valid strings that can occur in a set of strings known as a *language*. The set of valid strings is described using a set of production rules $\mathcal{R}$ that specify transformations from a set of internal (or non-terminal) symbols $\mathcal{I}$ to sequences consisting of both non-terminal and terminal symbols. We let $\mathcal{V}$ represent the set of terminal symbols. A string in the language is simply a sequence of terminal symbols.

In this work, we will consider CFGs where each rule $r \in \mathcal{R}$ is either a triple (A, B, C) where $A \in \mathcal{I}$, $B \in \mathcal{I} \cup \mathcal{V}$, and $C \in \mathcal{I} \cup \mathcal{V}$, or a tuple (A, a) where $A \in \mathcal{I}$ and $a \in \mathcal{V}$.[2] Such rules can be written in the form $A \to BC$ or $A \to a$.

To generate a string, the set of rules $\mathcal{R}$ is applied recursively starting from a special "start" symbol $\alpha \in \mathcal{I}$. The generation of a terminal symbol serves as the recursion base case. The sequence of recursive production rule applications thus generates a

[2]This is a slightly relaxed form equivalent to Chomsky normal form [10]. It allows for more compact sets of production rules.

binary tree with non-terminal symbols as the internal nodes and terminal symbols as the leaf nodes. The left-to-right sequence of terminal symbols in the tree gives the generated string.

A CFG is formally defined as a tuple $G = (\mathcal{I}, \mathcal{V}, \mathcal{R}, \alpha)$. The language defined by the grammar G consists of all strings that can be generated through the recursive application of production rules in $\mathcal{R}$ starting from α. As an example, consider a simple CFG with $\mathcal{I} = \{\alpha, A, B\}$, $\mathcal{V} = \{a, b\}$ and the production rules $\alpha \rightarrow AB$, $A \rightarrow aA$, $A \rightarrow a$, $B \rightarrow bB$, $B \rightarrow b$.[3] The recursive application of these rules produces strings containing any number of a's followed by any number of b's.

The problem of parsing a string is the problem of inverting the generative process defined by the grammar to infer the tree structure and production rules responsible for generating the string. In the simple example described above, every string in the language has a unique valid parse, but this is not the case in general. In such cases, weights can be attached to the productions to express their relative likelihoods, and the parsing problem can be converted into the problem of identifying the tree structure and production rules that result in the parse with the maximum total weight. The weighted CFG can equivalently be viewed as a probabilistic model over trees, and the maximum weighted parse can be interpreted as the maximum probability parse.

The CFG CRF model is a subclass of CRF models that specifies a probability distribution over parse trees given a feature sequence and a context free grammar $G = (\mathcal{I}, \mathcal{V}, \mathcal{R}, \alpha)$. Let $\mathbf{X} = \{\mathbf{X}_i\}_{i=1}^{L}$ be an input sequence consisting of L feature vectors $\mathbf{X}_i$. Let $\{Y_{A,BC,i,j,l} \mid A \rightarrow BC \in \mathcal{R}, 1 \leq i \leq j < l \leq L\}$ be a set of binary variables where $Y_{A,BC,i,j,l}$ takes the value 1 if the rule $A \rightarrow BC \in \mathcal{R}$ is used in the parse tree with the sub-tree rooted at B covering positions i to j in the input sequence

[3] As a notational convenience, the possible productions starting from each non-terminal are typically written together using "|" as a separator. This allows writing the last four rules as $A \rightarrow aA|A$ and $B \rightarrow bB|B$.

and the sub-tree rooted at C covering positions $j+1$ to l. The variable $Y_{A,BC,i,j,l}$ takes the value 0 otherwise.

Next, we define a set of $K^{A,BC}$ scalar feature functions for every production rule $A \rightarrow BC$ in $\mathcal{R}$: $\{\mathbf{f}_k^{A,BC}(y_{A,BC,i,j,l}, i, j, l, \mathbf{x}) \mid 1 \leq i \leq j < l \leq L\}$. Each feature function takes the form of a product of the binary indicator variable $y_{A,BC,i,j,l}$ and a function $g_k^{A,BC}(i, j, l, \mathbf{x})$ that computes a feature value from $\mathbf{x}$, as shown below:

$$\mathbf{f}_k^{A,BC}(y_{A,BC,i,j,l}, i, j, l, \mathbf{x}) = y_{A,BC,i,j,l} \cdot g_k^{A,BC}(i, j, l, \mathbf{x}). \tag{2.14}$$

While this model is substantially richer and more complex than either the linear-chain or semi-Markov CRFs, it has the important property that MAP and marginal inference can still be performed in polynomial time. Specifically, marginal inference can be performed in $\mathcal{O}(|\mathcal{R}|L^3)$ time using the inside-outside dynamic programming algorithm originally developed for the weighted CFG model [31]. MAP inference can be performed using an algorithm closely related to the inside portion of the inside outside algorithm [65]. In the next section, we discuss methods for estimating the parameters of CRF models.

2.1.3 Learning in CRFs

Given a data set $\mathcal{D} = \{(\mathbf{y}_n, \mathbf{x}_n)\}_{n=1}^N$ of fully labeled training examples, the unknown parameters θ must be learned from training data before the model can be used for prediction. Two commonly used learning methods for CRFs are **maximum likelihood learning** and **maximum margin learning**.

2.1.3.1 Maximum Likelihood Learning

In maximum likelihood learning, we try to find the parameters that maximize the likelihood of the observed data. More specifically, the parameters are estimated by maximizing the conditional log-likelihood shown below:

$$\mathcal{L}_{ML}(\theta|\mathcal{D}) = \sum_{n=1}^{N} \log P_\theta(\mathbf{y}_n|\mathbf{x}_n) \tag{2.15}$$

$$= \sum_{n=1}^{N} \langle \theta, \mathbf{f}(\mathbf{x}_n, \mathbf{y}_n) \rangle - \log Z_\theta(\mathbf{x}_n) \tag{2.16}$$

This objective can alternatively be viewed as minimizing the conditional KL-divergence from the empirical distribution to the distribution given by the model. For a log-linear model, this objective function is strongly convex, so gradient-based methods are guaranteed to find the unique optimal solution. In particular, due to the log-linear nature of the model, we have that the gradient is zero when $\frac{1}{N}\sum_n \mathbf{f}(\mathbf{x}_n, \mathbf{y}_n) = \frac{1}{N}\sum_n \mathbb{E}_{P_\theta(\mathbf{y}|\mathbf{x}_n)}[\mathbf{f}(\mathbf{x}_n, \mathbf{y})]$. That is, in maximum likelihood learning, we are trying to match the expected feature function under the model to the expected feature function under the empirical distribution.

The computational bottleneck to computing the gradient of the log-likelihood is computing the gradient of the log partition function, $\nabla_\theta \log Z_\theta(\mathbf{x})$; however, as shown above, this is equivalent to performing marginal inference. That is, $\mathbb{E}_{P_\theta(\mathbf{y}|\mathbf{x})}[\mathbf{f}(\mathbf{x}, \mathbf{y})] = \nabla_\theta \log Z_\theta(\mathbf{x})$. Therefore, any log-linear model that supports efficient marginal inference also supports efficient maximum likelihood learning.

2.1.3.2 Maximum Margin Learning

An alternative way to learn the parameters of a CRF is to treat the energy function as a scoring function and to minimize a loss on predictions made by this scoring function. A **scoring function** maps configurations of the label variables to a real number and predictions can be made by finding the configuration with the highest score. One noteworthy loss function that has successfully been used for this purpose is the multiclass hinge loss. Given a set of fully labeled training examples $\mathcal{D} = \{(\mathbf{y}_n, \mathbf{x}_n)\}_{n=1}^N$, we can learn the parameters by minimizing the multiclass hinge-loss defined below:

$$\mathcal{L}_{MM}(\theta|\mathcal{D}) = \sum_{n=1}^{N} \max_{\mathbf{y}'} \langle \theta, \mathbf{f}(\mathbf{x}_n, \mathbf{y}') - \mathbf{f}(\mathbf{x}_n, \mathbf{y}_n) \rangle \tag{2.17}$$

This objective can be efficiently minimized using a number of methods including sub-gradient descent, cutting-plane methods [71, 67], and the Frank-Wolfe algorithm [29]. All of these methods need only an efficient MAP inference algorithm to work.

One problem with this objective is that it does not distinguish between training examples for which only a single variable is predicted incorrectly and examples where all variables are predicted incorrectly. One way to correct for this is to modify the hinge loss with an error term. Given an error function $\Delta : \mathcal{Y}^2 \to \mathbb{R}_+$, the loss-augmented hinge loss is defined as:

$$\mathcal{L}_{MM-L}(\theta|\mathcal{D}) = \sum_{n=1}^{N} \max_{\mathbf{y}'} \langle \theta, \mathbf{f}(\mathbf{x}_n, \mathbf{y}') - \mathbf{f}(\mathbf{x}_n, \mathbf{y}_n) \rangle + \Delta(\mathbf{y}, \mathbf{y}') \tag{2.18}$$

Optimizing this objective requires that we be able to perform efficient loss-augmented MAP inference. That is, we must be able to solve the following optimization problem

$$\max_{\mathbf{y}' \in \mathcal{Y}} \langle \theta, \mathbf{f}(\mathbf{x}, \mathbf{y}') \rangle + \Delta(\mathbf{y}_n, \mathbf{y}') \tag{2.19}$$

For many common error functions (e.g. zero-one error) this problem is very difficult; however, if the error function Δ is chosen so that it decomposes over the graphical model G, then it is often possible to augment the feature vectors $\mathbf{x}$ and parameter vector θ such that performing regular MAP inference in the augmented model is equivalent to performing loss-augmented MAP inference [71]. An important case of this is Hamming loss, which can be decomposed as a sum over the label variables and incorporated by augmenting unary feature functions.

2.1.4 Structured Prediction in mHealth

Numerous structured prediction models have previously been applied to the activity detection problem including: linear chain models [20, 55], variable duration linear chain models [66], and semi-Markov models [32, 75, 61, 66]. Importantly, all of these applications of structured prediction to activity detection assume homogeneity of the activities of interest. That is, it is assumed that a subject is engaging in one activity at a time and activities do not compose. We highlight three notable exceptions. Liao et al. [34] recognize that full activities can be decomposed into heterogeneous parts and use a model akin to those used for fixed segmentation in images [54]. Koppula and Saxena [26] further recognize the benefits of a dynamic segmentation, so they propose placing a distribution over the model structure and sampling it. Finally, Sung et al. [64] propose a joint segmentation and sequence labeling model similar to those presented in Chapter 3 that allows for a segmentation with heterogeneous sequence labels beneath it and apply it to activity detection from image sequences. In Chapter 3 we generalize this approach and apply it to three new detection problems.

2.2 Weakly Supervised Learning

The learning methods described in the previous section both require a set of fully labeled examples; however, in many cases the cost of acquiring such data can be quite high. Reducing the cost of acquiring labeled data is a fundamental problem in supervised learning and has been addressed using a number of approaches. One way to decrease the cost of obtaining labeled data is to decrease the amount of data that needs to be labeled. *Active learning* aims to optimally select the instances that are labeled from an unlabeled pool or stream of instances with the goal of learning better models from lower volumes of labeled data [60]. *Semi-supervised learning* aims to learn from small volumes of labeled data by combining it with large amounts of unlabeled data [8]. *Positive unlabeled learning* is a special case of semi-supervised

learning where a small number of positive instances have labels, but a large pool of unlabeled instances are available [33].

An alternative to labeling less data is to lower the cost of obtaining each label, which is typically achieved by lowering the quality of labels in some way. This form of learning is often referred to as *weakly supervised learning*. In this section, we review some of the approaches to weakly supervised learning for both independent and structured models.

2.2.1 Weakly Supervised Learning in Independent Models

Weak supervision may come in a variety forms. Here we survey some of standard weak supervision paradigms following a categorization presented by Zhou [86].

2.2.1.0.1 Inexact Supervision In the *inexact supervision* paradigm, accurate supervision is provided, but not at the granularity needed to use traditional supervised learning techniques. For example, suppose we are interested in localizing objects within an image. To use fully supervised learning, we require a dataset containing images annotated with a list of objects and their locations within each image; however, it is faster to label images if all the annotator has to do is provide a list of objects that appear in each image but *not* the object locations.

One of the most studied weak supervision problems is the multiple instance (MI) learning problem. MI learning generalizes supervised learning by allowing for sets (or "bags") of instances to be labeled instead of single instances [36]. For a classification problem, it is assumed that a bag labeled c contains at least one instance with the label c. For an extensive review of MI learning problems and techniques, see Carbonneau et al. [7]. The learning from label proportions framework is closely related to MI learning in that instances are still grouped into bags, but instead of receiving only a single label for each bag, we are provided with the proportion of each label type within each bag [21, 51].

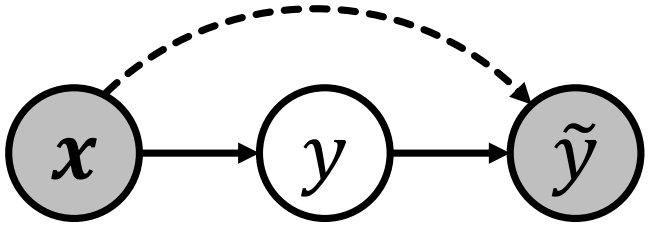


Figure 2.2: A typical graphical model used for inaccurate supervision problems. $\mathbf{x}$ is the input features, y is the true, unobserved label, $\tilde{y}$ is the observed, noisy version of the label. The dashed arrow from $\mathbf{x}$ to $\tilde{y}$ indicates that the noise model does not always depend on the features. Grey nodes indicate variables that are observed *during training*.

One version of inexact supervision that is particularly relevant to the problem addressed in Chapter 4 is when instances and labels occur in unaligned sequences with different lengths [18]. Graves et al. [18] address this problem by treating the alignment of the two sequences as a latent variable that is marginalized out; however, the label sequence is assumed to be accurate and the timestamps of positions in the sequences are not considered. The work presented in Section 4.2 can be view as an extension of this framework to allow for inaccurate supervision.

2.2.1.0.2 Inaccurate Supervision In the *inaccurate supervision* paradigm, supervision is provided at the appropriate granularity, but may contain errors. For example, this may occur when annotations requiring expertise are provided by non-experts [84]. While standard supervised learning models may be trained directly on the noisy labels, this generally leads to poor performance [45, 85]. There are three common approaches to this problem: detecting and correcting label errors, using a supervised method that is inherently robust to label noise, and explicitly modeling label noise. For a comprehensive survey of these methods, see Frénay and Verleysen [14].

Numerous approaches have been proposed for identifying and correcting label errors including model-based (e.g. [69]), neighborhood-based (e.g. [82]), and boosting-

based (e.g. [76]) among others. Alternatively, if we can identify methods that are naturally robust to label noise, then we can apply them directly without modifying the data. While most common loss functions are not robust to label noise [14], a few exceptions have been found [15, 16].

The approach we adopt later in this dissertation is to use a probabilistic model in which the true label for an instance is a latent variable and the model includes a distribution over noisy labels [83, 84, 53, 24, 39]. An example of this type of model is shown in Figure 2.2 where $\mathbf{x}$ is the feature vector, y is the unobserved true label, and $\tilde{y}$ is the observed noisy label. The learning algorithms and observation noise models depend on the specific applications, but learning generally involves maximizing the marginal likelihood of the observed noisy labels using either expectation maximization or gradient methods.

2.2.2 Weak Supervision in Structured Models

There has also been significant research in the area of weakly supervised structured prediction, particularly in computer vision applications. We briefly survey this work, again using the categorization presented in Zhou [86].

2.2.2.0.1 Inexact Supervision Many of the standard inexact supervision frameworks, such as MI learning, have been extended to structured prediction. Song et al. [63] and Vezhnevets et al. [77] extend the multiple instance support vector machine framework to structured support vector machines by treating an image as a bag of pixels or a bag of overlapping sub-windows. Guan et al. [20] extend multiple instance learning to an auto-regressive HMM using a similar approach. The core assumption in all three methods is that there are correlations between instances within a bag that can be modeled using structured prediction methods.

One inexact supervision problem unique to structured prediction is when only a subset of the label variables in the structured model are observed. This can be

handled in probabilistic models by marginalizing out the unobserved variables when marginal inference is tractable [52] and using approximate inference when it is not [59, 70]. In this way, supervision may propagate from the observed label variables to the unobserved label variables. A related problem is considered by Pathak et al. [49] who assume that supervision is provided as constraints on the set of possible structure $\mathcal{Y}$. Mann and McCallum [35] consider yet another inexact supervision framework where supervision is provided in the form of constraints on the marginal label distributions in structured models. These constraints can then be enforced on otherwise unlabeled data.

2.2.2.0.2 Inaccurate Supervision Inaccurate supervision for structure prediction models is a relatively understudied paradigm compared with inaccurate supervision for independent models; however, a few examples of work in this area do exist. Dredze et al. [13] consider the case where multiple, potentially conflicting label structures are given for each sample. Rather than include an explicit noise model, they estimate the probability that an observed label is correct and use that to weight observations in the loss function. Gross et al. [19] show that optimizing a loss that decomposes over the label structure (as opposed to, say, joint log likelihood) is more robust to noise in the individual label variables. Finally, Vahdat [73] make the assumption that errors occur at the level of individual variables in the label structure. They define a CRF over both the true label structure and the observed noisy label structure and estimate the parameters of this model by maximizing a lower bound on the marginal likelihood. In Chapter 4, we present an new weakly supervised learning problem in which the parameters of a structure prediction model over discrete variable must be learned from continuous annotations that correspond roughly to the discrete structure. Existing weakly supervised structured prediction methods do not cleanly apply to this problem as such methods generally assume that the labels used

Name	Citation	Activity	Setting	Sensors	# features	# subjects	# sessions
mPuff	Ali et al. [1]	smoking	lab	RIP band	17	10	13
puffMarker	Saleheen et al. [56]	smoking	lab, field	RIP band, actigraph	30	6	32
Extrasensory	Vaizman et al. [74]	various	field	phone, actigraph	175	28	80
rConverse	Bari et al. [3]	conversation	lab	RIP band	34	12	12
ECGmorph	Natarajan et al. [43]	cocaine use	lab, field	ECG	100	11	1704
eatingMoments	Thomaz et al. [68]	eating	lab	actigraph	15	20	20
RisQ	Parate et al. [47]	smoking	field	actigraph	34	15	18

Table 2.1: Basic information and statistics for the datasets used in this dissertation.

for weak supervision are also discrete. In the next section, we describe the datasets used to evaluate methods throughout this work.

2.3 Datasets

We use seven mHealth datasets to evaluate the methods proposed in the remainder of this dissertation. In this section, we summarize relevant information about each of these datasets. Basic information and statistics for these datasets is shown in table 2.1 and more details are provided in the following sections. For complete details on each dataset, see the original citations.

2.3.1 mPuff (Ali et al. [1])

The mPuff dataset contains smoking data gathered in a laboratory setting using a respiratory inductive plethysmograph (RIP) band. The RIP band is worn around the chest and measures the area inclosed within the band. As the subject inhales and exhales, this area grows and shrinks. While subjects smoked a cigarette, an observer recorded the occurrence of smoking puffs using a mobile phone. The RIP signal was discretized into non-overlapping respiration cycles using methods described in Ali et al. [1]. The recorded smoking puff timestamps were then visualized alongside the RIP signal and the recorded timestamps were manually aligned to individual respiration cycles to generate binary instance labels and a segmentation of the input sequence into periods of smoking and non-smoking. A total of 17 features were extracted from each respiration cycle. These features measure morphological structure

in each respiration cycle including magnitude and duration features. This dataset includes 13 smoking activities across 10 subjects. We consider each of these smoking activities a distinct session.

2.3.2 puffMarker (Saleheen et al. [56])

The puffMarker dataset contains smoking data gathered both in the lab and the field using a RIP band, a 3-axis wrist-worn accelerometer, and a 3-axis wrist-worn gyroscope. The sensors were worn at all times, but smoking was carried out in the presence of an observer. As in the mPuff dataset, the observer recorded the occurrence of smoking puffs using a mobile phone. The RIP signal was discretized into non-overlapping respiration cycles using methods described in Saleheen et al. [56]. As in the mPuff dataset, the observed smoking puff timestamps were visualized alongside the RIP signal and the recorded timestamps were manually aligned to individual respiration cycles to generate binary instance labels and a segmentation of the session into periods of smoking and non-smoking. 30 features were extracted from each respiration cycle. These features include those from the mPuff dataset as well as 13 new morphological features. This dataset includes 32 distinct smoking activities across 6 subjects. For each smoking activity, we sampled random amounts of non-smoking activities from either side to create a complete session.

2.3.3 Extrasensory (Vaizman et al. [74])

The extrasensory dataset[4] contains data on a wide variety of activities, but we use only the annotation of sleep. The data was gathered in the field using a 3-axis mobile phone accelerometer, a 3-axis mobile phone gyroscope, a mobile phone GPS, a mobile phone microphone, and a 3-axis wrist-worn accelerometer. Data was collected from all sensors for a twenty second period every minute. This leads to a natural discretization

[4]Available at `http://extrasensory.ucsd.edu/`

where each instance represents a minute. Subjects self-reported a wide range of activities using a specially designed mobile phone application. Subjects could freely log activities or respond to prompts asking about specific activities. If an instance overlapped with a reported activity, it was labeled with that activity. Researchers used ad hoc rules to correct inconsistencies between the self-report location or between self-reported activities. A total of 175 features were extracted from the sensors listed above including a wide variety of statistical and morphological features. The complete dataset contains data from 60 subjects and contains around 214 days of labeled data in total. We filtered the data using two criteria. First, we dropped any instance with missing features. Second, we partitioned the data into 24 hour sessions beginning and ending at 2:00pm, dropping any session with less than four hours of recorded data or less than one hour of reported sleep. After filtering, we were left with 80 sessions from 28 subjects.

2.3.4 rConverse (Bari et al. [3])

The rConverse dataset contains conversation and speaking data gathered in the lab using a RIP band. The RIP signal was discretized in non-overlapping respiration cycles using methods described in Bari et al. [3]. While wearing the RIP band, subjects were recorded using a microphone. An annotator later listened to the audio and marked each respiration cycle as containing speech by the subject or not, resulting in a binary label for each instance and a segmentation into periods of conversation and non-converation. A total of 34 morphological and statistical features were extracted from each respiration cycle including duration and magnitude-based features. Data was collected from 6 pairs of subjects (12 total). Each pair performed approximately 45 minutes of scripted and unscripted activities requiring different amounts of conversation and interaction. We treat each complete 45 minute signal as a session resulting in 12 sessions across 12 subjects.

2.3.5 ECGmorph (Natarajan et al. [43])

The ECGmorph dataset contains electrocardiogram (ECG) data gathered both in the lab and the field using a wireless ECG sensor. The ECG signal was discretized using an off-the-shelf peak detection method. Each peak was visualized and manually labeled as one of the characteristic ECG peaks, P, Q, R, S, or T, or a noise peak, N. Sparse coding was used to learn a length-100 sparse representation of a 204 millisecond window around each peak. Additionally, the peak height and squared peak height were included as features. Data was gathered from six subjects in the lab and five subjects in the field. Sessions containing two to four cardiac cycles were randomly sampled from the complete data. 1531 sessions were sampled from the lab data and 173 sessions were sampled from the field data.

2.3.6 eatingMoments (Thomaz et al. [68])

The eating moments dataset[5] contains eating data gathered in the lab using a 3-axis wrist-worn accelerometer. Instances were generated using six second sliding windows with three seconds of overlap. While eating, subjects were recorded on video and this video was later annotated with the beginning and end of hand-to-mouth eating gestures. An instance was given a positive label if at least one hand-to-mouth gesture fell within the corresponding six second sliding window. All eating within a session was considered a single eating activity. Five statistical features were extracted for each accelerometer channel for a total for 15 features. These features included the mean, variance, skewness, kurtosis, and uncentered second moment of each channel's signal within the sliding window. An average of approximately 31 minutes of data was collected for each of the 20 subjects as they performed scripted eating activities. Each subject's data was treated as a separate session.

---

[5]Available at `http://users.ece.utexas.edu/~ethomaz/`

2.3.7 RisQ (Parate et al. [47])

Finally, the RisQ dataset contains smoking data gathered in the field using a 9-axis inertial measurement unit (IMU) consisting of a 3-axis wrist-worn accelerometer, a 3-axis wrist-worm gyroscope, and a 3-axis wrist-worm magnetometer. Instances corresponding to gestures were extracted using a gesture detection method described in Parate et al. [47]. Subjects self-reported the beginning and end of smoking activities. For the purposes of annotating individual gestures, subjects wore a second 9-axis IMU on their upper arm. For each gesture in the proximity of a self-reported smoking activity, a 3D model of the subjects arm was visualized performing the gesture and the gesture was labeled as a smoking gesture or not. This resulted in a binary labeling of the input sequence and segmentation into periods of smoking and non-smoking. Duration, velocity, displacement, and angle features were extracted for each gesture resulting in 34 total features. Over 32 hours of data was collected from 15 subjects containing 17 complete smoking activities. The data from each subject was treated as a complete session with the exception of a single subject whose data contained a number of smoking activities. This subject's data was split randomly between each smoking activity. This resulted in a total of 18 sessions.

CHAPTER 3

STRUCTURED PREDICTION MODELS FOR HETEROGENEOUS MHEALTH SEGMENTATION

mHealth activity detection can often be framed as a segmentation problem. For example, given the signal from a wearable sensor and an activity of interest, we may want to segment the signal into periods where the subject is engaging in the activity of interest and periods where they are not. In traditional segmentation problems, all instances within a segment are assumed to share the same class (e.g. [32, 75, 61]). An example of such a problem is sleep detection since we can reasonably assume that a subject is not engaging in other activities during sleep. We call this type of segmentation problem homogenous segmentation. In such problems, the instance labels and segmentation are not typically treated as separate variables as they represent the same thing. This is in contrast to heterogeneous segmentation where instance labels may vary within a segment. For example, during eating, a person may engage in a wide variety of other activities between bites of food such as talking, drinking, or checking their phone. The difference between homogenous and heterogeneous segmentation is illustrated in Figure 3.1.

In the mHealth literature, this problem is commonly solved by applying post hoc segmentation rules to the predictions from an independent classifier [56, 38]. For example, if we are performing eating detection, we might reasonably assume that two bites of food that occur more than two minutes apart belong to separate eating activities [38]. Such approaches have two disadvantages. First, segmentation rules must be developed for every new detection application and detection performance will be heavily dependent on the quality of these ad hoc rules. Second, performing

segmentation in a post hoc manner allows information to flow in only one direction, from the instance labels to the segmentation. On the other hand, by considering the instance labeling and segmentation problems together, information about what a typical activity looks like can be used to "clean-up" the instance-level predictions. A discussion of previous applications of structured prediction to activity detection can be found in Section 2.1.4. While the heterogeneous segmentation has been addressed in at least one specific application [64], we significantly generalize this approach and apply it to three new detection problems.

Our first contribution in this chapter is the introduction a class of heterogeneous segmentation models and the application of this model class to three real mHealth detection problems. In Section 3.2, we introduce the proposed class of heterogeneous segmentation models. In Section 3.3, we consider the problem of conversation and speech detection where the goal is to segment the input sequence into periods of conversation and non-conversation and label each instance as containing speech or not. To solve this problem, we use a simple segmentation model where the instance labels are assumed to be independent given the segmentation. In Section 3.4, we consider the problems of eating and smoking detection. Like conversation, smoking and eating are heterogeneous activities; however, these activities exhibit structure in the distribution of positive instance labels within a segment, which we leverage to improve detection accuracy. Finally, in Section 3.5, we consider the problem of electrocardiogram morphology extraction where the problem is to identify the characteristic morphological structure of a heartbeat from an electrocardiogram signal. We frame this as a heterogeneous segmentation problem in order to overcome the limitations of the Markov assumption and to leverage the regular structure and timing of heart beats.

Our second contribution is a set of inference acceleration methods that allow inference in the proposed segmentation models to scale to long input sequences. In

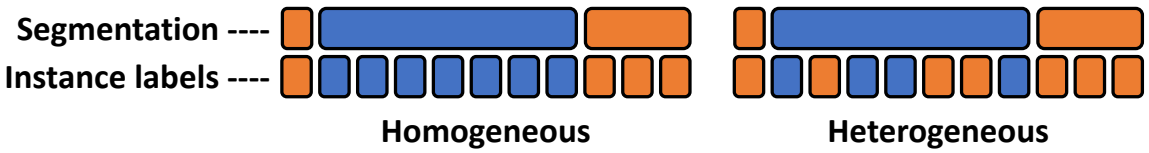


Figure 3.1: An illustration of homogeneous and heterogeneous segmentations. Colors indicate class label.

particular, we present two strategies based on pruning the dynamic programs used for inference in segmentation models. First, in Section 3.6.1, we present and evaluate two static, domain specific constraints on the set of possible segmentations. Second, in Section 3.6.2, we adapt the learning-to-prune method first presented for parsing by Bodenstab et al. [5] to the case of segmentation and evaluate this method on two activity detection problems.

3.1 Notation

As described in Section 1.1, we assume that the input data consists of N multivariate time series that we will call **sessions**. Each session contains a set of time-aligned signals gathered from one or more sensors. Separate sessions may correspond to data from different subjects, or to data from the same subject collected at different times. We assume that each session n has been discretized into a sequence of L_n **instances** and that a feature vector $\mathbf{x}_{ni} \in \mathbb{R}^D$ has been extracted for each instance i. Further, each instance i in session n is associated with a timestamp t_{ni} which may correspond to the start, end, or other point of interest associated with instance i. We refer to the complete sequence of feature vectors $\mathbf{x}_n = \{\mathbf{x}_{ni}\}_{i=1}^{L_n}$ as the **input sequence** and the complete sequence of timestamps $\mathbf{t}_n = \{t_{ni}\}_{i=1}^{L_n}$ as the **timestamp sequence**.

In this chapter, our goal is to map $\mathbf{x}$ and $\mathbf{t}$ to a length L sequence of instance labels and a labeled segmentation of the input sequence. We will use two sets of variables to describe this labeling and segmentation. First, let $\mathbf{y}^{(1)} = \{y_i^{(1)}\}_{i=1}^{L}$ be the length L sequence of **instance labels** where $y_i^{(1)} \in \mathcal{C}^{(1)}$ is the instance label for

instance i. Second, as described in Section 1.1, we represent a labeled segmentation as a sequence of segments $\mathbf{y}^{(2)} = \{y_s^{(2)}\}_{s=1}^{S}$ where each segment $y_s^{(2)} = (c_s, j_s, k_s)$ is a tuple containing a label $c_s \in \mathcal{C}^{(2)}$, a start position $j_s \in \{1, ..., L\}$, and an end position $k_s \in \{1, ..., L\}$. We denote a full labeling and segmentation as $\mathbf{y} = \{\mathbf{y}^{(1)}, \mathbf{y}^{(2)}\}$ and our goal is to learn a function that maps $(\mathbf{x}, \mathbf{t})$ to $\mathbf{y}$. In the next section, we present a class of CRF models for performing this type of labeling and segmentation.

3.2 Heterogeneous Segmentation

Our proposed model is a segmentation based CRF model. We include include both instance and segment-level factors. We incorporate feature information at the instance level using a simple log-linear pairwise factor $\psi^{(1)}(y_i^{(1)}, \mathbf{x}_i) = \exp(\langle \theta^{(1)}, \mathbf{f}^{(1)}(y_i^{(1)}, \mathbf{x}_i) \rangle)$ which is applied to each instance i. At the segment level, we include the standard semi-Markov factor $\psi^{(2)}(y_s^{(2)}, c_{s-1}, \mathbf{x}, \mathbf{t}) = \exp(\langle \theta^{(2)}, \mathbf{f}(y_s^{(2)}, c_{s-1}, \mathbf{x}) \rangle)$ which is applied to each segment in $\mathbf{y}$. Finally, we include a pairwise potential between each instance label and the segment containing it, $\psi^{(3)}(y_i^{(1)}, y_s^{(2)}) = \exp(\langle \theta^{(3)}, \mathbf{f}^{(3)}(y_i^{(1)}, y_s^{(2)}) \rangle)$. This factor can be used to model interactions between the instance labels and segmentations; however, it follows the assumption that instance labels are independent give the segmentation. The specific forms of these feature functions are application dependent and we give some examples in the following sections. The complete model can be written as

$$P_\theta(\mathbf{y}|\mathbf{x}) \propto \prod_{s=1}^{S} \psi^{(2)}(y_s^{(2)}, c_{s-1}, \mathbf{x}, \mathbf{t}) \prod_{i=j_s}^{k_s} \psi^{(3)}(y_i^{(1)}, y_s^{(2)}) \psi^{(1)}(y_i^{(1)}, \mathbf{x}_i) \tag{3.1}$$

where $\theta = \{\theta^{(1)}, \theta^{(2)}, \theta^{(3)}\}$ is the full set of parameters for the model. In the next section, we present a method for performing MAP inference in this model which we use to perform maximum margin learning, as discussed in Section 2.1.

3.2.1 Inference

We perform exact MAP inference in this model using an augmented version of the dynamic program used for inference in the semi-Markov CRF. Here, we present the dynamic program for MAP inference, however, the dynamic program for marginal inference can be obtained by swapping maximization for summation. The dynamic program for MAP inference has two parts: First, for each possible segment, we maximize over the instance labels of instances contained in that segment. Second, we maximize over the segmentation. We store the contributions of the instance-level factors in the table $\alpha^{(1)}$ which is defined by the following recursion:

$$\alpha^{(1)}(j,k,c) = \max_{y_k^{(1)} \in \mathcal{C}^{(1)}} \psi^{(1)}(y_k^{(1)}, \mathbf{x}_k) \cdot \psi^{(3)}(y_k^{(1)}, c) \cdot \alpha^{(1)}(j, k-1, c) \tag{3.2}$$

with the base case $\alpha^{(1)}(j,k,c) = 1$ for all $c \in \mathcal{C}^{(2)}$ and $j > k$. The entry $\alpha^{(1)}(j,k,c)$ is the contribution to the MAP score of the instance-level features for instances for j through k given that the MAP segmentation contains a segment $y = (c, j, k)$. Next, we maximize over all possible segmentations using a small modification to dynamic program for inference in the semi-Markov CRF (see 2.1.2.2). We denote the dynamic programming table as $\alpha^{(2)}$ which is defined by the following recursion:

$$\alpha^{(2)}(k,c) = \max_{j=1,\ldots,k} \max_{c' \in \mathcal{C}^{(2)}} \psi^{(2)}((j,k,c), c', \mathbf{x}) \cdot \alpha^{(1)}(j,k,c) \cdot \alpha^{(2)}(j-1, c') \tag{3.3}$$

with the base case $\alpha^{(2)}(0,c) = 1$ for all $c \in \mathcal{C}^{(2)}$. The score of the MAP labeling is given by $\max_{c \in \mathcal{C}^{(2)}} \alpha^{(2)}(L,c)$ and the MAP labeling can be found by backtracking through this dynamic program. Filling in $\alpha^{(1)}$ has complexity $\mathcal{O}(|\mathcal{C}^{(1)}||\mathcal{C}^{(2)}|L^2)$ and filling in $\alpha^{(2)}$ has complexity $\mathcal{O}(|\mathcal{C}^{(2)}|^2 L^2)$ so the complete inference algorithm has complexity $\mathcal{O}(|\mathcal{C}^{(2)}|(|\mathcal{C}^{(1)}| + |\mathcal{C}^{(2)}|)L^2)$. Equipped with this inference algorithm, we can now estimate the parameters of this model using maximum margin learning. In the following sections, we describe applications of this model to three real mHealth

problems: conversation detection, sleep and eating detection, and ECG morphology extraction.

3.3 Conversation Detection

In this section, we apply the heterogeneous segmentation model to the problem of conversation and speech detection from a wearable respiration monitor used to track inhalation and exhalation of a subject. Conversation is at the core of many social interactions and understanding the patterns of conversation, such as turn-taking behavior, can help us understand mental well-being and work productivity [46, 80]. In particular, understanding turn-taking behavior requires identifying both when a person is speaking, as well as the context in which they are speaking (e.g. a conversation). The goal in this problem is to segment the input sequence into periods of conversation and non-converstation and to label each instance (in this case, a respiration cycle) as containing speech by the subject or not. That is, the set of possible instance labels $\mathcal{C}^{(1)} = \{0, 1\}$ is binary with a positive instance label for instance i indicating that instance i contains speech by the subject. Likewise, the set of segment labels $\mathcal{C}^{(2)} = \{0, 1\}$ is also binary with a positive segment label $c_s = 1$ indicating that segment s corresponds to a complete conversation. In the rest of this section we describe the proposed model and evaluate the model on respiration data gathered in the lab.

3.3.1 Model

To model speech and conversation, we use the following factors. We incorporate instance level features using a pairwise factor $\psi^{(1)}(i, y_i^{(1)}, \mathbf{x}) = \exp(\langle \theta^{(1)}_{y_i^{(1)}}, \mathbf{x}_i \rangle)$. At the segment level, we incorporate two segment-level features which aggregate features from the instances contained within each segment. First, for a segment beginning at j and ending at k, we include the mean of instance features contained within it,

$\bar{\mathbf{x}}_{jk} = \frac{1}{k-j}\sum_{i=j}^{k}\mathbf{x}_i$. Second, we include a normalized histogram of feature values for instances contained within a segment. We do this by first discretizing each feature into one of five evenly spaced percentile bins where the bin edges are calculated using the complete training set. Next, we convert each feature to a five-bin, one-hot encoding and, finally, we average the values of each of the one-hot encoded feature vectors for all instances contained within a segment. Call this operation $hist(\mathbf{x}, j, k)$. We define the complete segment level feature function as $\mathbf{f}^{(2)}(y_s^{(2)}, \mathbf{x}) = [\bar{\mathbf{x}}_{j_s k_s} \;\; hist(\mathbf{x}, j_s, k_s)]$. Then, the segment-level factor can be written as:

$$\psi^{(2)}(y_s^{(2)}, \mathbf{x}) = \exp(\langle \theta_{c_s}^{(2)}, \mathbf{f}^{(2)}(y_s^{(2)}, \mathbf{x})\rangle). \tag{3.4}$$

Finally, we include a pairwise potential between each instance label and the label of the segment containing it, $\psi^{(3)}(y_i^{(1)}, c_s) = \exp\left(\theta^{(3)}_{y_{ic_s}^{(1)}}\right)$. This models the probability of speaking inside and outside of a conversation. The full model is shown in Equation 3.3 and MAP inference is described in Section 3.2.1. We used loss-augmented maximum margin learning to learn the parameters of this model with Hamming-loss on the instance label sequence as the loss. To perform learning, we used a cutting plane method [72] as implemented by PyStruct [41].

3.3.2 Experiments

In this section, we present a comparison of the proposed model against two baseline models on the speech detection (instance labeling) task. We use the rConverse dataset (Section 2.3), which consists of data gathered from subjects during conversation in a lab setting using a respiration monitor [3]. We used all instance features described in Bari et al. [3]. Additionally, we applied a non-linear transformation to these features by finding five equal-sized percentile bins for each feature and calculat-

ing the Euclidean distance from the center of each percentile bin to the input feature value.

3.3.2.1 Models

We compare the proposed model (SEG) against an independent model and a linear-chain model. The independent model (IND) is a linear model which includes only the instance-level features and makes independent speech/non-speech predictions for each instance. The linear-chain model (CHAIN) includes instance-level features as well as pair-wise potentials between adjacent instance labels (for full-details, see Section 2.1.2.1). We trained the SEG and CHAIN models using loss augmented max-margin learning with instance level Hamming-loss as the loss function and the IND model using hinge-loss. This makes IND equivalent to a linear SVM and CHAIN equivalent to a linear-chain structured SVM. All models were trained using ℓ_2 regularization.

3.3.2.2 Train and Test Procedures

We evaluated each model using a leave-one-subject-out cross-validation procedure. For all models, the strength of the ℓ_2 regularization was tuned over a logarithmic grid to maximize average accuracy using a further leave-one-subject-out cross-validation on the training set.

3.3.2.3 Experiment: Speech Detection

The average instance-level prediction accuracy for each model is show in Figure 3.2 (left) with error bars showing one standard-error calculated across subjects. We can see that the IND and CHAIN models perform nearly the same with accuracy around 0.8 whereas the SEG model gives an accuracy of around 0.85, a relative error reduction of approximately 25%. The improvement of SEG over IND and CHAIN is significant at the $p = 0.05$ level using a paired t-test with Bonferroni correction. Figure 3.2 (right)

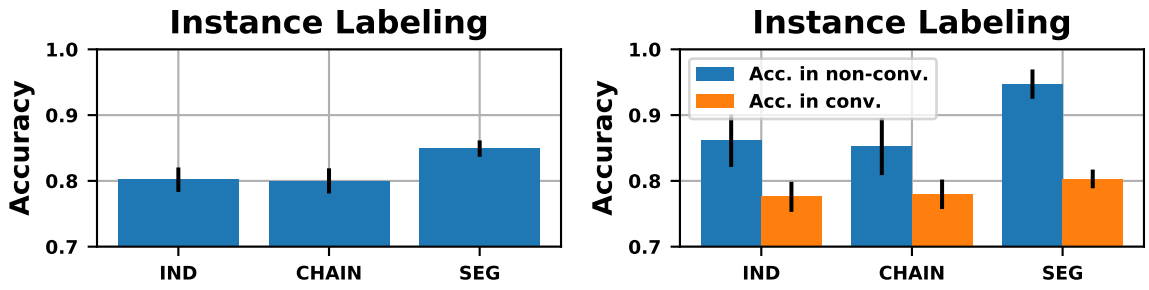


Figure 3.2: (Left) Average instance-level labeling accuracy for each model. Error bars represent one standard-error calculated across subjects. The y-axis is clipped at 0.7. (Right) Average instance-level labeling accuracy for each model in conversation and non-conversation activities.

shows average accuracy for each model within true conversation and non-conversation activities. While the SEG model performs best in both cases, the performance gap is much higher in the non-conversation activities. This is explained by observing that, in our data, subjects do very little speaking outside of conversations. The SEG model recognizes this and learns a negative weight for positive instances within a negative segment. As a result, the SEG model is able to suppress positive instances outside of conversations whereas, the IND and CHAIN models predict many false positives. This is a case where making use of top-down information substantially improves instance-level performance.

3.4 Eating and Smoking Detection

In some heterogeneous mHealth segmentation problems, the distribution of positive instances within a positive segment exhibits modelable structure. This is the case for semi-periodic behaviors such as eating, in which the activity of interest (eating a meal) consists of semi-regularly spaced bites. As in speech detection, our goal is to provide a label for each instance and a labeled segmentation of the instances. A positively labeled segment corresponds to the activity of interest.

In this section, we present and evaluate a heterogeneous segmentation model for detection of semi-periodic activities. This model encodes two structural quantities: the time between consecutive positive instances and the number of positive instances per activity of interest. For example, in the case of eating, we model the time between consecutive bites of food and the number of bites it takes to consume a meal. This domain structure generalizes to both eating and smoking detection problems and we evaluate it using four real mHealth datasets. In the next section, we describe the heterogeneous segmentation model used for eating and smoking detection.

3.4.1 Model

As in conversation detection, the set of possible instance labels $\mathcal{C}^{(1)} = \{0, 1\}$ is binary. We make two modifications to the segmentation structure described in Section 3.2 that allow us to model the time between positive instances and the number of positive instances that make up a positive activity. First, unlike in the speech detection problem, a positive segment will no longer represent a complete activity. Instead, let each segment $y_s^{(2)}$ represent the period between two positive instances. We enforce this interpretation by constraining the space of joint labelings such that *every segment $y_s^{(2)}$ must begin with a positive instance followed by any number of negative instances*. We call this **constraint (1)**.

By defining segments in this way and including segment duration as a feature (described in more detail below), we are able model the time between positive instances. The second modification we make is to change the set of segment labels to include all non-negative integers up to a positive integer C which represents the maximum possible number of positive instances in a positive activity. That is, $\mathcal{C}^{(2)} = \{0, 1, ..., C\}$. A segment with label $c_s > 0$ now represents the period between positive instances c_s and c_s+1 in a positive activity while a segment with label $c_s = 0$ represents a negative activity. In essence, the label values are counting the number of positive instances

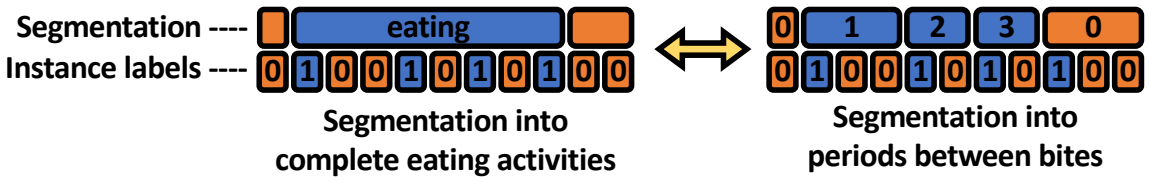


Figure 3.3: On the left is an example of a standard instance labeling and segmentation where a positive instance represents a bite of food and a positive segment represents a complete eating activity. On the right is the implied segmentation into periods between positive instances described in Section 3.4.1. At test time, predictions are converted back to the original observed format.

in each positive activity. We enforce this interpretation by further constraining the set of possible segmentations to only allow transitions from segments with label c_s to segments with label $c_s + 1$ or segments with label 0 if $c_s > 0$. We call this **constraint (2)**.

Importantly, this constrained label set allows transitions between segments with label $c_s > 0$ and segments with label 0. These transitions represent the end of positive activities and we can model the number of positive instances per positive activity by placing a weight on these types of transitions. We note that, data is not generally provided with this type of labeling. Instead, it is typically annotated with the locations of positive instances and beginning and end of positive activities; however, there is a one-to-one mapping between a standard activity segmentation and the segmentation described above. Figure 3.3 illustrates this mapping.

We include both instance and segment-level factors. At the instance level, we include a log-linear unary factor $\psi^{(1)}(i, y_i^{(1)}, \mathbf{x}) = \exp(\langle \theta^{(1)}_{y_i^{(1)}}, \mathbf{x}_i \rangle)$. At the segment level, we include a log-linear factor on the segment duration and segment duration squared. This induces the equivalent of a class-conditional normal distribution on the segment durations. This factor can be written as

$$\psi^{(2a)}(y_s^{(2a)}, \mathbf{t}) = \exp(\langle \theta^{(2a)}_{\mathbb{I}[c_s>0]}, [t_{k_s} - t_{j_s} \;\; (t_{k_s} - t_{j_s})^2] \rangle). \tag{3.5}$$

Also at the segment level, we include the following segment transition potential which models the number of positive instances in a positive activity and enforces constraint (2):

$$\psi^{(2b)}(c_s, c_{s-1}) = \begin{cases} \exp(\mathbb{I}[c_{s+1} = 0]\theta^{(2b)}_{c_s}) & : c_s = c_{s-1} + 1 \text{ or } (c_s = 0 \text{ and } c_{s-1} > 0) \\ 0 & : \text{else} \end{cases}$$

The complete segment-level factor can be written as $\psi^{(2)}(y^{(2)}_s, c_{s-1}, \mathbf{x}, \mathbf{t}) = \psi^{(2a)}(y^{(2)}_s, \mathbf{t}) \cdot \psi^{(2b)}(c_s, c_{s-1})$. Finally, we use $\psi^{(3)}$ to enforce constraint (1) as:

$$\psi^{(3)}(y^{(1)}_i, y^{(2)}_s) = \begin{cases} 1 & : i = j_s \text{ and } y^{(1)}_i = 1 \\ 1 & : i > j_s \text{ and } y^{(1)}_i = 0 \\ 0 & : \text{else} \end{cases}$$

The full model can be written as in equation 3.3. We make one important modification to the MAP inference described in Section 3.2.1. Because transitions are only allowed between certain types of segments, we can constrain the recursion for $\alpha^{(2)}$ in the following way:

$$\alpha^{(2)}(k, c) = \begin{cases} \max_j \ \psi^{(2)}((j, k, c), c-1, \mathbf{x}) \cdot \alpha^{(1)}(j, k, c) \cdot \alpha^{(2)}(j-1, c-1) & \text{if } c > 0 \\ \max_{j,c'} \psi^{(2)}((j, k, c), c', \mathbf{x}) \cdot \alpha^{(1)}(j, k, c) \cdot \alpha^{(2)}(j-1, c') & \text{if } c = 0 \end{cases} \tag{3.6}$$

In other words, we drop the maximization over the label for the previous segment when the current segment has label $c > 0$. This modification reduces the complexity of solving $\alpha^{(2)}$ from $\mathcal{O}(|\mathcal{C}^{(2)}|^2 L^2)$ to $\mathcal{O}(|\mathcal{C}^{(2)}| L^2)$. Since $|\mathcal{C}^{(2)}| = C + 1$ represents the maximum possible number of positive instances in a segmentation, this reduction in complexity is quite significant. Once again, we estimate the parameters of this model

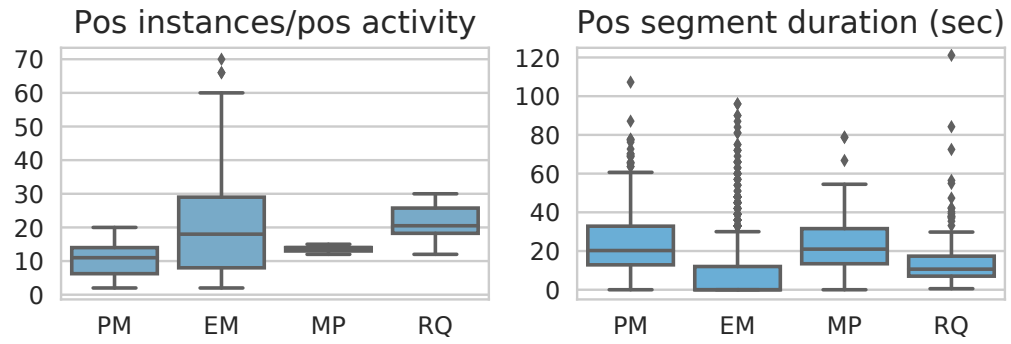


Figure 3.4: Distributions of the modeled quantities for the four datasets used. The left plot shows the distributions of the number of positive events in a complete positive activity. The right plot shows the distributions of the time in seconds between consecutive positive instances. For display purposes, three outliers were omitted from the eatingMoments dataset box-plot in right plot.

using loss-augmented maximum margin learning. We use the average of the Hamming loss on the instance labels and the Hamming loss of the segmentation as our loss.

3.4.2 Experiments

We evaluated the proposed model using synthetic data (described below) as well as four real mHealth datasets. The mPuff dataset (MP) was presented in Ali et al. [1] and contains respiration data from smokers recorded using a chest band sensor. The puffMarker dataset (PM) was presented in Saleheen et al. [56] and contains contains data from smokers recorded using a chest band sensor and a wrist-worn actigraphy device (accelerometer and gyroscope). The RisQ (RQ) dataset was presented in Parate et al. [47] and contains wrist-worn actigraphy data. Finally, we use the data presented in Thomaz et al. [68] (EM), which also contains wrist-worn actigraphy data for eating. For details on these datasets, see Section 2.3. Empirical distributions for the two structural quantities included in the segmentation model (positive instances per positive activity and time between consecutive positive instances) are shown in Figure 3.4. We can see that these distributions vary significantly across the datasets.

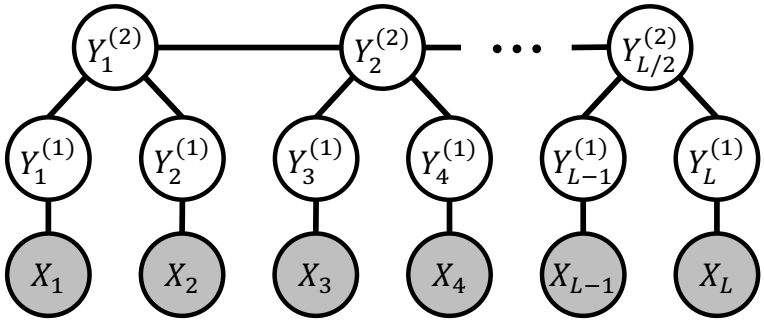


Figure 3.5: This figure shows the graphical model for the TREE baseline model with a window size of two. $Y_i^{(1)}$ is the i'th instance label and $Y_j^{(2)}$ is the j'th activity label.

For each of the datasets described above, we use the instance-level features originally published with the data with the exception of puffMarker, where we omit actigraphy features used in the original paper, which were not available for all instances. Additionally, we applied a non-linear transformation to these features by finding five equal sized percentile bins for each feature and calculating the Euclidean distance from the center of each percentile bin to the input feature value.

We evaluated performance on both the instance labeling and segmentation tasks. For smoking, a positive instance corresponds a puff on a cigarette or a cigarette-to-mouth gesture and for eating a positive instance corresponds to a food-to-mouth gesture. When using the heterogeneous segmentation model to predict segmentations, we treat all segments with label $c > 0$ as positive segments and a positive segment may represent smoking or eating depending on the dataset (for an example of this, see Figure 3.3).

3.4.2.1 Baseline Models

We compared the proposed model (SEG) against two baselines: a Logistic Regression (LR) model, and the tree structured pairwise CRF (TREE) shown in Figure 3.5. The TREE model includes two levels of labels: activity-level labels and instance-level labels. Each activity-level label corresponds to a fixed size window of instances in the

base sequence (Figure 3.5 shows a model with a window size of two). We treated the window size as a hyperparameter. The TREE model thus provides a strong segmentation baseline which allows for heterogeneous instance labels beneath homogenous activity labels, but is restricted to pairwise factors. We generated activity-level features for this model by averaging the instance-level features sitting beneath each activity-level label. The LR model was trained using ℓ_2 regularized maximum likelihood and the TREE and SEG models were trained using ℓ_2 regularized maximum margin methods. On the instance labeling task we compared against both the LR and TREE models and on the segmentation task we compared only against the TREE model since LR does not produce an explicit segmentation.

3.4.2.2 Train and Test Procedures

We conducted experiments using a stratified 10-fold cross-validation protocol. Specifically, we split the sessions into two groups, one for all sessions containing the activity of interest and one for the rest, and randomized within groups. Next, we created 10 test folds so that each test fold contained approximately the same number of sessions from each group. To select hyperparameter values, we performed a further stratified 10-fold cross-validation on the training samples. The LR hyperparameters were tuned to maximize instance-level F_1 score, while the TREE and SEG hyperparameters were tuned to maximize segmentation accuracy. ℓ_2 regularization hyperparameters for each model were tuned on a logarithmic grid and the window size hyperparameter for the TREE model was tuned on a linear grid.

We assessed the performance for instance labeling using precision, recall, and F_1 score, which adjusts for the major class imbalance we faced in this problem. We do not report accuracy due to strong class imbalance. For the segmentation task, we compared the predicted activity level segmentation to the true segmentation by projecting each segmentation onto the input sequence, and calculating the precision,

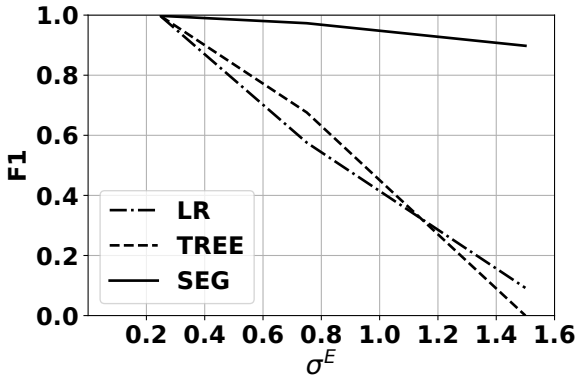


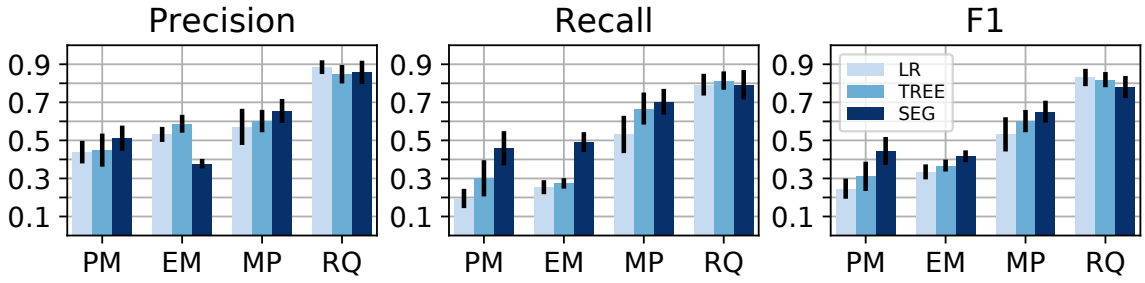
Figure 3.6: F_1 results for the LR, TREE, and SEG models on synthetic data with varied amounts of noise in the instance features as measured by σ^E.

recall, and F_1 score of the projected labels compared to the projected ground truth segmentation.

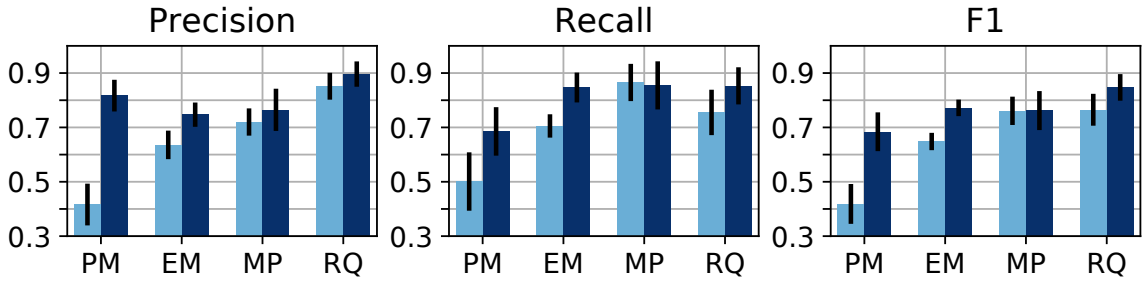
3.4.2.3 Experiment 1: Synthetic Data

To evaluate the performance of the SEG model under controlled noise conditions, we evaluated all models on a series of synthetic datasets. For each synthetic session, we sampled the length of each segment, the number of positive events per positive segment, and the number of negative instances between consecutive positive instances from discretized, truncated normal distributions with standard deviation $\sigma^S = 0.25$. Next, we sampled instance-level features from class conditional normal distributions with means separated by unit distance and a common standard deviation parameter σ^E, which we varied to simulate different amounts of discriminative information. That is, we sampled $\mathbf{x}_i$ from $\mathcal{N}(y_i^{(1)}, (\sigma^E)^2)$. We generated train, validation, and test sets containing 30, 50, and 50 sessions, each of length 100.

Figure 3.6 shows the instance-level F_1 score for each model versus the feature standard deviation (σ^E). When there is little noise in the features ($\sigma^E = 0.25$), all methods perform equally well; however, the SEG model substantially outperforms the other two models when there is there is less information in the instance features,



(a) Instance labeling task



(b) Segmentation task

Figure 3.7: The top row shows results on the instance labeling task and the second row shows results on the segmentation task. From left to right, the three panels in each row correspond to precision, recall, and F1. In each group of bars for the instance labeling task, the models are LR, TREE, and SEG. In each group of bars for the segmentation task, the models are TREE, and SEG.

indicating that the SEG model can more effectively leverage high-level structure in the data.

3.4.2.4 Experiment 2: Real Data

The results from our instance labeling experiments on each the mHealth benchmark datasets are shown in Figure 3.7a. The SEG model performs better in terms of average F_1 score on three of the four data sets. We ran a paired t-test on the combined results of all datasets and found that the SEG model achieves an improvement over both LR and TREE that is statistically significant at the $p = 0.05$ level using Bonferroni correction.

The results from the Task 2 segmentation experiments are shown in Figure 3.7b. The SEG model outperforms the TREE baseline in terms of F_1 score on three of the four datasets and has the same performance on the remaining dataset (mPuff). The improvements range from 0.082 to 0.266 F_1 When all datasets are considered together, the improvement in segmentation F_1 over the TREE model is significant at the $p = 0.05$ level. Finally, we note that we ran the same set of experiments using maximum likelihood learning for the SEG model; however, segmentation performance was uniformly worse across all datasets than using maximum margin learning.

One of the most common failure modes of the TREE model is a tendency to extend activities beyond the first or last positive instance in the activity. The SEG model avoids this by defining activities to begin and end on positive instances. This avoids slop in the segmentation and improves segmentation accuracy.

Unfortunately, our results on the segmentation task are not easily comparable to the original papers in which the data sets appeared as these papers do not consider the segmentation problem directly. On the instance labeling task, Ali et al. [1] evaluated performance on rebalanced data, Saleheen et al. [56] used ad-hoc pre-filtering methods to form train and test sets, and the exact data used for evaluation in Parate et al. [47] is not available. Our implementation of the random forest experiment from Thomaz et al. [68] achieves Task 1 F_1 score of 0.31[1]. This is very close to the performance of LR on the same task indicating that performance is not limited by the choice of a linear model, at least on the EM dataset.

3.5 Electrocardiogram Morphology Extraction

Until now, we have considered only binary segmentation problems, that is, our goal was to segment a sequence into positive and negative segments. In this section,

[1]This number differs somewhat from the performance reported in [68] due to differences in the train/test splits and the way results were averaged.

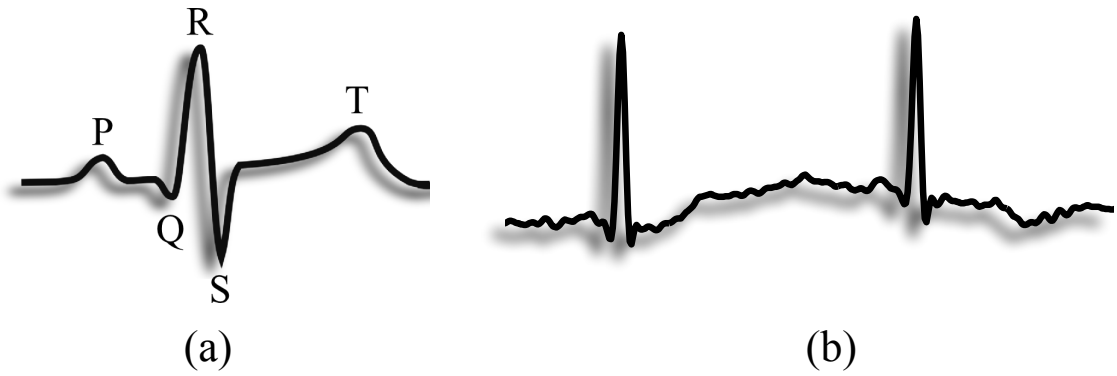


Figure 3.8: (a) Idealized ECG waveform (b) Sample data from the Zephyr BioHarness wearable chest band sensor

we consider electrocardiogram (ECG) morphology extraction, a setting with multiple segment types and strong transition structure.

An electrocardiogram (ECG) sensor produces a data stream corresponding to the electrical activity of the muscles of the heart as measured at the surface of the skin. Each normal cardiac cycle (corresponding to a single heart beat) produces a characteristic sequence of five waves (the P, Q, R, S, and T waves) as shown in Figure 3.8a. These waves are the result of atrial and ventricular depolarization and repolarization. Identifying each part of the heartbeat in an ECG signal, which we will call *morphology extraction*, entails labeling the ECG trace with the positions of each P, Q, R, S and T wave, when present.

We adopt the problem formulation in Natarajan et al. [43] which is consistent with our formulation of an activity detection problem. That is, we assume that the ECG signal has been discretized using a generic peak detection method. Features are calculated on windows around each peak to form an instance sequence $\mathbf{x} = \{\mathbf{x}_i\}_{i=1}^{L}$. The morphology extraction problem then simplifies to a sequence labeling problem where each candidate peak generated by the peak detection method must be labeled as a P, Q, R, S, or T wave or a noise peak, denoted by the label N. The peak detection

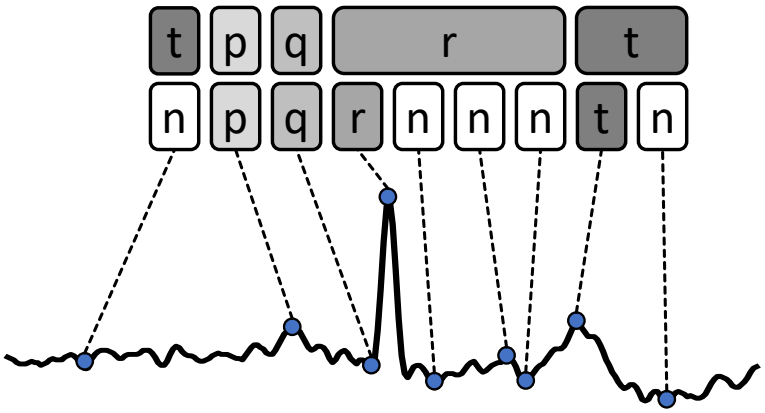


Figure 3.9: A sample ECG signal with peaks marked and an example of a labeling and segmentation of this sample.

method is tuned to over-produce peaks so that there is minimal probability of missing a true wave. In the remainder of this section, we describe a heterogeneous segmentation model for ECG morphology extraction and then evaluate the performance of this model on real mobile ECG data.

3.5.1 Model

In the ECG morphology extraction problem, the instance label set $\mathcal{C}^{(1)} = \{p, q, r, s, t, n\}$ includes the five ECG peak types, p, q, r, s, and, t, as well a label n indicating that a peak is none of the five ECG peak types. The set of possible segment labels is the set of ECG peak types $\mathcal{C}^{(2)} = \{p, q, r, s, t\}$. Similar to the model presented in section 3.4, we define a segment as representing the period between two consecutive ECG peaks. To achieve this, we include a constraint similar constraint (1) in section 3.4 that *the first instance in a segment must have take a label from the set* $\{p, q, r, s, t\}$ *and any remaining instances must have the label* n. Further, the label of the first instance in a segment must match the label of the segment. That is, for a segment $y_s^{(2)} = (c_s, j_s, k_s)$, $y_{j_s}^{(1)}$ must equal c_s and $y_i^{(1)}$ must equal n for all $i \in \{j_s + 1, ..., k_s\}$. An example of this type of segmentation is shown in Figure 3.9.

We include instance-level features $\mathbf{x}_i$ using a log-linear pairwise potential on the instance label $\psi^{(1)}(i, y_i^{(1)}, \mathbf{x}) = \exp(\langle \theta^{(1)}_{y_i^{(1)}}, \mathbf{x}_i \rangle)$. We model the time between peaks of each type using a segment-level log-linear factor on the segment durations:

$$\psi^{(2a)}(y_s^{(2)}, c_{s+1}, \mathbf{t}) = \exp(\langle \theta^{(2a)}_{c_s c_{s+1}}, [t_{k_s} - t_{j_s} \quad (t_{k_s} - t_{j_s})^2] \rangle) \tag{3.7}$$

This is equivalent to placing a normal distribution on the time between two peaks where the mean and variance of the distribution is learned separately for each pair of instance label types. In other words, the model can represent the characteristic period between p and q peaks, p and r peaks, p and s peaks, and so on. Further, we include a transition potential between consecutive segments

$$\psi^{(2b)}(c_s, c_{s+1}) = \exp(\theta^{(2b)}_{c_s c_{s+1}}) \tag{3.8}$$

The complete segment-level factor can be written as $\psi^{(2)}(y_s^{(2)}, c_{s-1}, \mathbf{x}, \mathbf{t}) = \psi^{(2a)}(y_s^{(2)}, c_{s+1}, \mathbf{t}) \cdot \psi^{(2b)}(c_s, c_{s-1})$. Finally, we use $\psi^{(3)}$ to enforce the labeling constraint defined above as

$$\psi^{(3)}(y_i^{(1)}, y_s^{(2)}) = \begin{cases} 1 & : i = j_s \text{ and } y_i^{(1)} = c_s \\ 1 & : i > j_s \text{ and } y_i^{(1)} = n \\ 0 & : \text{else} \end{cases}$$

The full model can be written as in equation 3.3 and the MAP inference algorithm is presented in section in section 3.2.1. Once again, we estimate the parameters of this model using loss-augmented maximum margin learning with Hamming loss on the instance label sequence as our augmentation loss. In the next section, we present an evaluation of this model on real mobile ECG data gathered in both the lab and the field.

3.5.2 Experiments

We evaluated the proposed model on the ECGmorph dataset, which consists of data from wearable a ECG monitor discretized using a peak detection method. Each peak in the dataset is labeled as p, q, r, s, t, or n. This dataset includes data gathered in the lab and in the field and we evaluated our model on both sets separately. For full details on the dataset, see Section 2.3. The task of interest in this dataset is instance labeling. While the proposed model also generates a segmentation, it is only used to propagate information between instance labels and is discarded at prediction time.

3.5.2.1 Models

We considered three different models for labeling the input sequence. We compared the performance of the segmentation model described above (SEG), the linear-chain CRF model (CHAIN) from Natarajan et al. [43], and a multinomial logistic regression model (MLR). The MLR model uses only instance features to independently predict each instance label. The linear-chain CRF model uses both instance features and transition information between peaks to predict instance labels. We trained the MLR and CHAIN models using maximum likelihood estimation while the SEG model was trained using max-margin learning. All models included tuned ℓ_2 regularization.

3.5.2.2 Train and Test Procedures

The ECG dataset contains both data gathered in the lab as well as data gathered in the field (see Section 2.3 for details). We performed two, slightly different, prediction experiments on the lab and field datasets. When evaluating on the lab data, we used a leave-one-subject-out protocol, holding one subject out as a test subject and training all models on the remaining subjects. We then averaged evaluation metrics across all subjects. For the field data, we trained all models on the complete set of lab data and

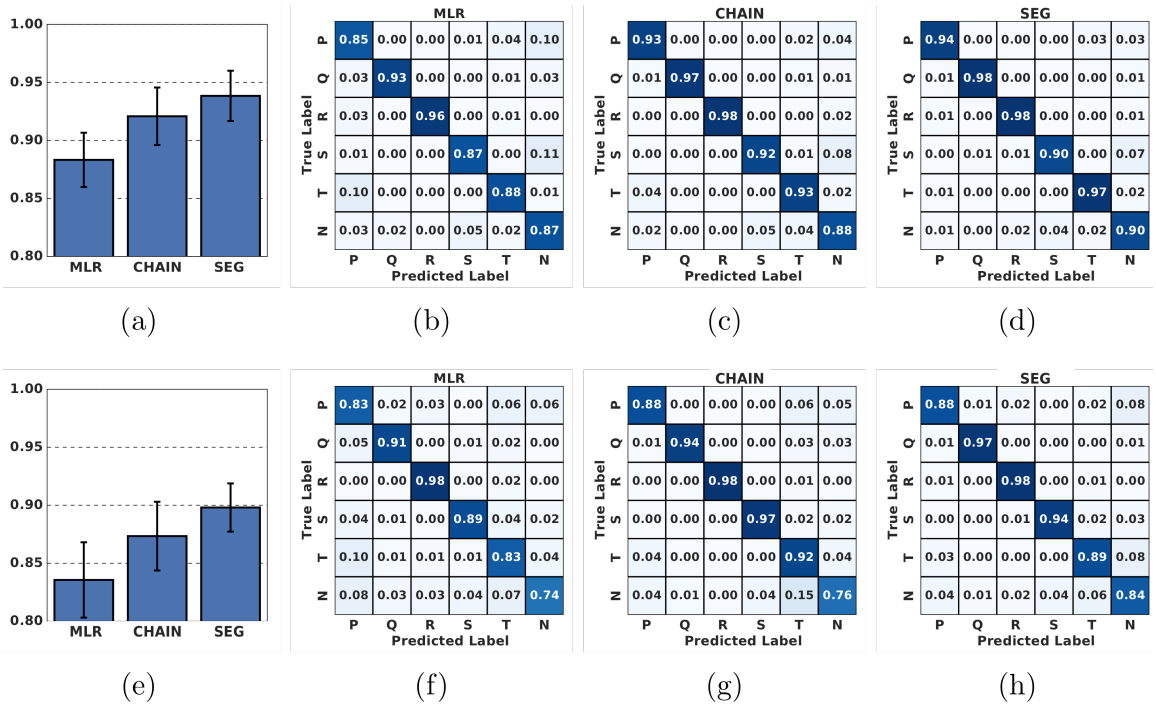


Figure 3.10: (a) shows the average accuracies across lab subjects, (b)-(d) show confusion matrices for the lab subjects, (e) shows the average accuracies across field subjects, and (f)-(h) show confusion matrices for the field subjects.

tested on the complete set of field data. In both cases, we tuned hyperparameters using further cross-validation on the training set. Specifically, we randomly generated three leave-one-subject-out train/validation splits from the training data and averaged performance across these three subjects. ℓ_2 regularization strength hyperparameters for each model were tuned on a logarithmic grid to maximize average label accuracy.

3.5.2.3 Experiment: Peak Labeling

The leave-one-subject-out prediction accuracy for each model averaged across subjects is shown in Figure 3.10(a) for the lab data. Error bars show one standard error calculated with respect to the subjects. These results show that the SEG model outperforms the CHAIN model by a margin of 1.8% for a relative error reduction of 22.2%. These results are significant at the $p = 0.05$ using a paired t-test with Bonferroni correction. Additionally, the confusion matrices for each method are shown

in Figures 3.10(b)-(d). The results show that the SEG method is at least as good as the linear-chain CRF model for most peak types, and provides significantly improved performance for T-waves.

The largest gap between the SEG and CHAIN models is on the accuracy of classifying t peaks. The CHAIN model has an accuracy of 0.93 whereas the SEG model has an accuracy of 0.97. In order to understand where the performance gains of the SEG model are coming from, we looked at t peaks that the SEG model correctly classified but the CHAIN model did not. There are 148 such instances. On approximately 88% of these instances, the chain model that predicted the previous instance in the session was a noise peak, n. This supports the hypothesis that the CHAIN model loses its place in the peak sequence when it encounters a noise peak.

The average classification accuracies on the field subjects are shown in Figure 3.10(e). The confusion matrices for the field setting are shown in Figures 3.10(f)-(h). We can see that while the SEG model does not have uniformly better performance for all peak types compared to the linear-chain CRF in the field setting, it does have superior overall performance obtaining a 2.2% improvement in accuracy for a relative error reduction of 19.5%. These results are significant at the $p = 0.1$ level using a paired t-test with Bonferroni correction. We attribute this drop in significance, at least in part, to the much smaller amount of labeled field data. We also note that while there is a drop in performance between the lab and field data sets due to the strict across-subjects protocol used, the overall performance gap between the SEG and CHAIN models is consistent in both the lab and field settings.

One trend of note is that the SEG model outperforms the linear-chain CRF at classifying T waves on the lab data, but not on the field data. This appears to be explained by a number of extremely noisy samples in the field data. In these cases, the linear-chain CRF tends to label many of the peaks as T waves resulting in high recall, but low precision. The SEG model, on the other hand, chooses a single peak

to call the T wave. While the choice is sometimes incorrect, the predicted structure makes much more sense than having a number of adjacent T waves. The result is that the SEG model has lower T wave performance, but higher overall accuracy.

3.6 Improving Inference Times in Segmentation Models

All of the models presented thus far admit polynomial time exact inference and until now, we have considered only offline processing of relatively short sequences, so polynomial inference complexity has been sufficient. In this section, we consider settings where quadratic or cubic inference complexities are unacceptable and evaluate a collection of methods for improving the run time of inference in the models discussed in this chapter.

One setting where polynomial time inference may be unacceptable is applications that generate long input sequences. For example, consider a conservative hypothetical case of the ECG morphology extraction problem. Assuming an average heart rate of 60 beats per minute and *perfect* candidate peak generation (i.e. no noise peaks), every hour of data will result in a sequence of approximately 18,000 instances. Further, if a subject is asked to wear a sensor all day, the raw input time series may be 8-16 hours long. For the purposes of morphology extraction, breaking the sequence into pieces may result in relatively few errors; however, if an activity detection task is layered on top, breaking the sequence may result in corruption of important quantities such as activity duration.

In this section, we present two strategies for accelerating the inference dynamic programs presented in this section. In particular, we focus on techniques for avoiding intermediate computations that do not have a large impact on the final output of the dynamic program. This form of inference accelleration is sometimes refered to as **pruning**. There are many approaches to pruning dynamic programs in graphical models. In Section 3.6.1 we present static pruning methods, which leverage domain

knowledge to constrain the set of possible segmentations. We present two such constraints on the heterogeous segmentation model from the previous section. In Section 3.6.2, we adapt the learning-to-prune method presented in Bodenstab et al. [5] to segmentation models and evaluate this method on the sleep and smoking detection problems.

3.6.1 Static Pruning

In many mHealth applications, we have domain knowledge that allows us to constrain the set of possible segmentations. Such constraints can be used to improve the runtime of inference. We refer to such pruning strategies a **static pruning**. The following are two examples static pruning that we apply to the heterogeneous segmentation models from this chapter. We analyze the effect on the complexity of inference for the heterogeneous segmentation model proposed for smoking and eating detection (Section 3.4). Recall that this model had an inference complexity of $\mathcal{O}(|\mathcal{C}^{(2)}|L^2)$:

- **Maximum segment length:** In many cases, the time between events or activities is not unbounded and we can upper-bound the length of individual segments. For example, in the case of smoking detection, we might say that two smoking puffs separated by five minutes (or approximately 50 respiration cycles) constitute two separate smoking activities. We can enforce this constraint by bounding the length all segments with non-zero labels. Adding this constraint reduces the complexity of inference to $\mathcal{O}(L^2 + |\mathcal{C}^{(2)}|LB)$ where B is the maximum segment length.

- **Maximum cardinality:** Similarly, we may reasonably bound the number of puffs that it takes to smoke a cigarette. This translates to a bound C on the size of the segment label set $\mathcal{C}^{(2)}$. Enforcing this bound reduces inference complexity to $\mathcal{O}(CL^2)$.

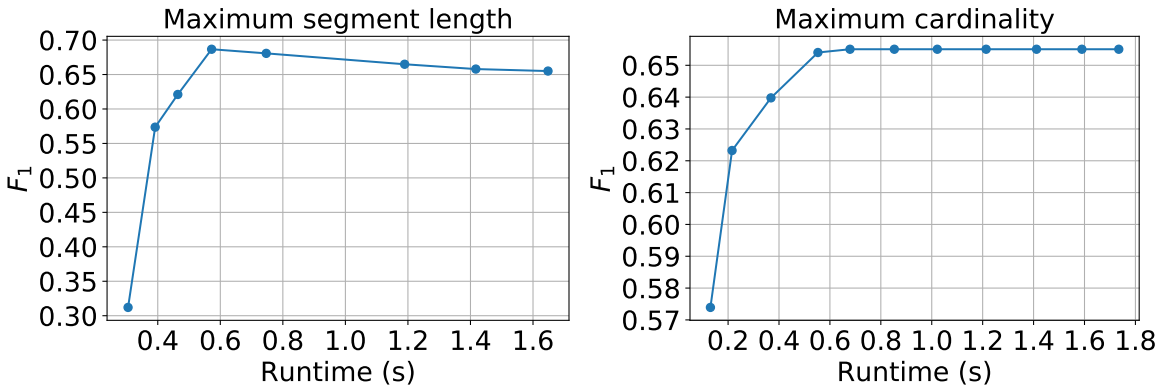


Figure 3.11: This figure shows the inference runtime against instance-level classification performance for different settings of each of the static pruning constraints described in this section.

3.6.1.0.1 Experiments We evaluated these static constraints on the puffMarker dataset using the heterogeneous segmentation model for smoking (Section 3.4). We used all respiration based-features. Further, we extracted features from the actigraphy data using the following procedure: Let t_i be the timestamp of the maximum peak in respiration cycle i. Extract a window beginning 8 seconds before t_i and ending 1 second after t_i and calculate as features the mean, max, min, standard deviation, median, and five bin histogram of each channel's signal within this window. The actigraphy channels included were accelerometer x, y, and z, accelerometer magnitude, gyroscope x, y, and z, gyroscope magnitude, and pitch and roll angles for a total of 100 actigraphy based features. Pitch and roll calculations using accelerometer data are only valid when the hand is stationary, so these signals were filtered using the procedure described in [56].

To evaluate the trade-off between inference runtime and accuracy, we first trained a model with $B = 100$ and $C = 50$. For comparison, the maximum session length L in the puffMarker dataset is 389 and the maximum number of positive instances in any session is 20. Next, we varied B and C to generate the Pareto curves shown in Figure 3.11. In both cases, static pruning achieves approximately a 3 times speedup

in inference compared to the original settings with little to no loss in performance. In fact, the performance appears to improve slightly as we constrain the maximum segment length. This is likely because the average distance between positive instances is not, in fact, very long, so constraining the model in this way is providing useful domain knowledge. In the next section, we consider learning constraints on the label set.

3.6.2 Learning to Prune

An alternative to static pruning is to learn pruning rules from data. This idea has been successfully applied to parsing [5, 78] and, more broadly, the idea of learning to filter a structured output space has been applied successfully in a number of settings (e.g. [81]). In this section, we review the learning-to-prune method presented in Bodenstab et al. [5] and adapt it to the semi-Markov CRF (the same ideas can be applied to heterogeneous segmentation models with little change). This presentation of learned pruning methods follows that in Vieira and Eisner [78], but is adapted to the case of segmentation.

3.6.2.1 Pruned MAP Inference

Bodenstab et al. [5] frame pruning as a decision making process carried out during the inference algorithm. Every time the inference algorithm fills in an entry in the dynamic programming table, a decision is made as to whether that entry is important for the final computation. If it is deemed unimportant, it is skipped, saving computation time. Every time an entry is skipped, it is equivalent to filtering the space of possible segmentations. The core learning problem is learning a decision function that can make these decisions while achieving an acceptable speed-accuracy trade-off.

Recall that the dynamic program for MAP inference in the semi-Markov CRF $\alpha(k, c)$ has two indices: k represents a position in the input sequence and c represents a segment label. Let $g_{\mathbf{w}} : \mathcal{X} \times \{1, ..., L\} \rightarrow [0, 1]$ be a pruning function, parameterized

Algorithm 1

$\alpha \leftarrow \mathbf{0}$
$\alpha(0, :) \leftarrow \mathbf{1}$
for $k = 1, ..., L$ **do**
    if $g_{\mathbf{w}}(\mathbf{x}, k) < \frac{1}{2}$ **then**
        continue
    for $j = 1, ..., k$ **do**
        if $g_{\mathbf{w}}(\mathbf{x}, j) < \frac{1}{2}$ **then**
            continue
        for $c \in \mathcal{C}$ **do**
            for $c' \in \mathcal{C}$ **do**
                $s \leftarrow \exp(\langle \theta, \mathbf{f}((c, j, k), c', \mathbf{x}) \rangle) \cdot \alpha(j - 1, c')$
                if $\alpha(k, c) < s$ **then**
                    $\alpha(k, c) \leftarrow s$
return α

by $\mathbf{w}$, that maps the feature sequence and a position in the input sequence to a number between zero and one. If $g_{\mathbf{w}}(\mathbf{x}, k) < \frac{1}{2}$, then we skip over the entries $\alpha(k, c)$ for all c while performing inference. In essence, $g_{\mathbf{w}}(\mathbf{x}, k)$ must decide whether or not k looks like a likely position for the beginning of a segment. The complete MAP inference algorithm with pruning for the semi-Markov CRF is shown in algorithm 1.

The complexity of inference with pruning $\mathcal{O}\left(|\mathcal{C}|^2 \left(\sum_i g_{\mathbf{w}}(\mathbf{x}, i)\right)^2\right)$ is now a function of the pruning decisions and scales quadratically in the number of *unfiltered* positions. In the remainder of this section, we consider the problem of learning $g_{\mathbf{w}}$ from data.

3.6.2.2 Learning the Pruning Function

Bodenstab et al. [5] train $g_{\mathbf{w}}$ to recreate, as closely as possible, the output from a fixed, pre-trained model. In particular, let $\mathbf{y}^* = \arg\max_{\mathbf{y}} \langle \theta, \mathbf{f}(\mathbf{x}, \mathbf{y}) \rangle$ be the output from MAP inference in a semi-Markov CRF with feature function $\mathbf{f}$ and parameters θ. Then, $g_{\mathbf{w}}$ makes an error if it prunes a position that is a split-point in $\mathbf{y}^*$ (false negative) or if it *does not* prune a position that is not a split-point in $\mathbf{y}^*$ (false positive). In order to control the balance between inference runtime and fidelity to the original model, Bodenstab et al. [5] place a weight on all false positive errors.

When this weight is low, the model prioritizes fidelity to the original model and when it is high, the model prioritizes efficiency. Bodenstab et al. [5] use weighted zero-one loss as their error function and train a linear pruning function using a perceptron method. To allow for non-linear pruning functions, we instead use weighted log-loss to train $g_{\mathbf{w}}$. Specifically, given a dataset $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^{N}$, we train $g_{\mathbf{w}}$ to minimize the following objective:

$$\mathcal{L}(\mathbf{w}|\mathcal{D}) = \sum_{n} \sum_{i=1}^{L_n} \omega(\mathbf{y}^*, i) g_{\mathbf{w}}(\mathbf{x}, i) + \lambda_1 (1 - \omega(\mathbf{y}^*, i))(1 - g_{\mathbf{w}}(\mathbf{x}, i)) + \frac{\lambda_2}{2} ||w||_2^2 \quad (3.9)$$

where $\omega(\mathbf{y}, i) = \sum_s \mathbb{I}[i = j_s]$ is the indicator function for whether position i is a split-point in $\mathbf{y}$, λ_1 controls the accuracy/runtime trade-off, and λ_2 controls the ℓ_2 regularization strength.

One advantage of the loss in equation 3.9 is that the ground truth labels $\mathbf{y}_n$ do not appear anywhere. This allows us to train the filter function on unlabeled data, which is often much easier to acquire. However, equation 3.9 also has two disadvantages. First, because $g_{\mathbf{w}}$ is trained to replicate the original model, it will also learn to replicate the mistakes made by the original model. Second, because $g_{\mathbf{w}}$ is trained on the outputs of a pre-trained segmentation model, it cannot be used to accelerate inference during the learning of the segmentation model. Both of these weaknesses can be addressed by training $g_{\mathbf{w}}$ directly on the training data as shown in the following loss:

$$\mathcal{L}(\mathbf{w}|\mathcal{D}) = \sum_{n} \sum_{i=1}^{L_n} \omega(\mathbf{y}_n, i) g_{\mathbf{w}}(\mathbf{x}, i) + \lambda_1 (1 - \omega(\mathbf{y}_n, i))(1 - g_{\mathbf{w}}(\mathbf{x}, i)) + \frac{\lambda_2}{2} ||w||_2^2. \quad (3.10)$$

In the following section, we present an evaluation of these methods on a standard semi-Markov CRF for sleep detection and the heterogeneous segmentation for smoking detection presented in Section 3.4.

3.6.2.3 Experiments

3.6.2.3.1 Sleep Detection We evaluated Bodenstab pruning on a standard semi-Markov CRF using the Extrasensory dataset [74]. For full details on this dataset, see Section 2.3. Subjects carried a variety of sensors during daily activities and self-reported a range of activities such as sleeping, eating, and exercising. We focus on the sleep detection problem, as this was one of the more abundantly reported activities. We note that there is no ground truth for this data, so we evaluated against the cleaned, self-reported sleep annotations provided in the data.

Our goal in the sleep detection problem is to segment the input sequence into periods of sleep and non-sleep. We used a binary semi-Markov CRF with a constraint that consecutive segments may not have the same label. We included as features the sum of all instance-level features within a segment $x_{jk} = \sum_{i=j}^{k} x_i$ as well as two duration-based features: $\mathbb{I}[c_s = 1](t_{k_s} - t_{j_s})$ and $\mathbb{I}[c_s = 1](t_{k_s} - t_{j_s})^2$ where $\mathbb{I}[\cdot]$ is the indicator function. The duration-based features are equivalent to placing a normal distribution on the duration of sleeping activities. We placed a zero-mean gaussian prior with tuned variance on the parameters of the semi-Markov CRF model (i.e. ℓ_2 regularization).

We used a neural network as our filter model $g_{\mathbf{w}}$. First, we applied a sigmoid layer to each instance followed by a one-dimensional convolutional layer with width 9 and 5 output units, and finally, a second sigmoid layer as the output layer.

We evaluated performance on this dataset using a 10-fold cross-validation procedure, where folds were formed at the session level. The strength of the ℓ_2 regularizer was tuned to maximize instance-level F_1 over a logarithmic grid using a further 9-fold cross-validation on the training set.

3.6.2.3.2 Smoking Detection We further evaluated Bodenstab pruning on the heterogeneous segmentation model for smoking detection (Section 3.4) using the puffMarker dataset. We used all respiration based-features as well as the actigraphy-based

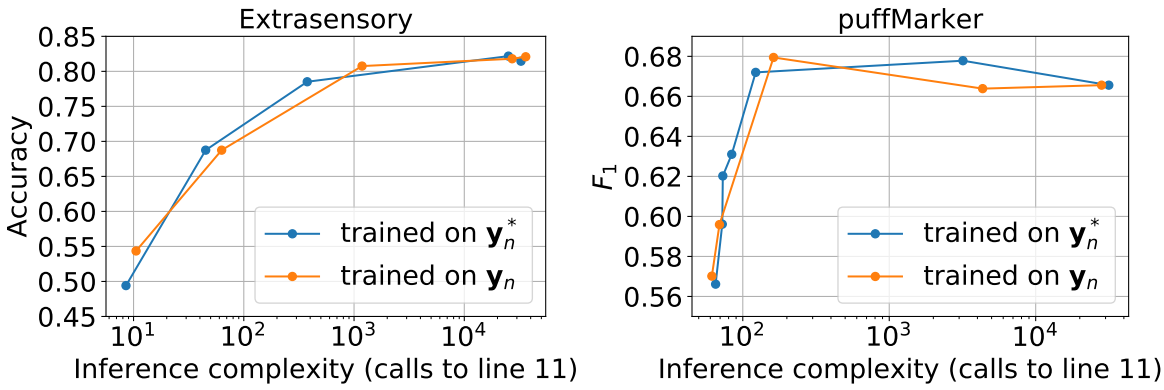


Figure 3.12: This figure shows the inference complexity against instance-level classification performance for different settings of the negative instance weight λ_1 the Bodenstab pruning objective. The left plot shows performance on the Extrasensory sleep detection dataset and the right plot shows performance on the puffMarker smoking dataset. Blue lines represent models trained using equation 3.9 and yellow lines represent models trained using equation 3.10.

features described in Section 3.6.1. We used logistic regression as our filter function $g_{\mathbf{w}}$, as the filter needs only detect a positive instance to know a segment transition should occur in this model.

3.6.2.3.3 Train and Test Procedures We evaluated performance on the Extrasensory and puffMarker datasets using 10 and 8-fold cross-validation procedures respectively, where folds were formed at the session level. The strength of the ℓ_2 regularizer was tuned to maximize instance-level F_1 over a logarithmic grid using further 9 and 7-fold cross-validation procedures on each training set.

3.6.2.3.4 Results For both datasets and both models, we varied the negative instance weighting parameter λ_1 over a logarithmic grid to evaluate the trade-off between runtime and test set performance. Figure 3.12 shows the Pareto frontier of inference complexity versus instance-level performance on the Extrasensory (left) and puffMarker (right) datasets. In both plots the x-axis represents the complexity of inference (calls to line 11 in algorithm 1) averaged across sessions and then across folds. The y-axis represents instance level performance averaged across folds. The

two lines in each plot represent training $g_{\mathbf{w}}$ using equation 3.9 and equation 3.10. On the Extrasensory dataset, Bodenstab filtering using equation 3.9 results in approximately a 66x speedup with only a 0.04 drop in average instance-level accuracy. On the puffMarker data, the results are even better, with Bodenstab filtering achieving approximately a 257x speedup with almost no change in instance-level F_1. Across both datasets, the difference between using equation 3.9 and equation 3.10 is minimal. These experiments demonstrate that we are able to achieve excellent speed-ups using trained filtering models.

3.7 Discussion

In this chapter, we presented a family of heterogeneous segmentation models that can be used for a variety of mHealth activity detection tasks. We derived quadratic time inference methods for this model family based on dynamic programming. We then applied instances of this model family to three real activity detection problems demonstrating improved performance on all three. Finally, we presented two inference pruning strategies which resulted in inference speedups of up to 257x.

Just like the linear-chain CRF and the semi-Markov CRF, heterogeneous segmentation models are a special case of context free grammar CRF models. For example, the ECG morphology extraction model can be written as a CRF-CFG model using the following grammar:

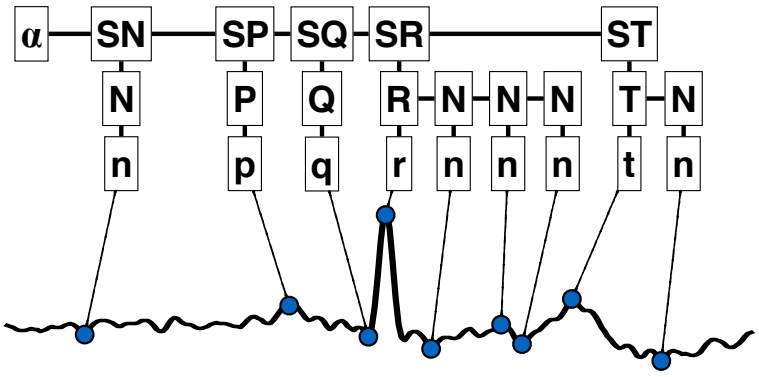


Figure 3.13: A sample ECG signal with peaks marked and a parse of this input sequence using the grammar from Section 3.7.

$$\alpha \rightarrow SP \mid SQ \mid SR \mid SS \mid ST \mid SN$$

$$SP \rightarrow P\ SQ \mid P\ SP \mid P\ SR \mid P\ SS \mid P\ ST$$

$$SQ \rightarrow Q\ SQ \mid Q\ SP \mid Q\ SR \mid Q\ SS \mid Q\ ST$$

$$SR \rightarrow R\ SQ \mid R\ SP \mid R\ SR \mid R\ SS \mid R\ ST$$

$$SS \rightarrow S\ SQ \mid S\ SP \mid S\ SR \mid S\ SS \mid S\ ST$$

$$ST \rightarrow T\ SQ \mid T\ SP \mid T\ SR \mid T\ SS \mid T\ ST$$

$$SN \rightarrow N\ SQ \mid N\ SP \mid N\ SR \mid N\ SS \mid N\ ST$$

$$P \rightarrow p \mid p\ N \qquad R \rightarrow r \mid r\ N \qquad T \rightarrow t \mid t\ N$$

$$Q \rightarrow q \mid q\ N \qquad S \rightarrow s \mid s\ N \qquad N \rightarrow n \mid n\ N$$

where α is the start symbol and $S\cdot$ symbols represent segment labels. An example of a parse tree from this grammar is shown in Figure 3.13. While this model does not admit a particularly compact CFG representation, this interpretation may still be useful in cases that do. While inference in CRF-CFG models generally has cubic complexity in the length of the input sequence, top-down parsing algorithms can achieve reduced complexity when the grammar allows it.

Heterogeneous segmentation models have a few notable limitations. First, they require the user to specify a discretization of the input sequence into instances. For some applications, this discretization is intuitive. For example, peak detection methods are a natural fit when the goal is to identify specific peaks as in ECG morphology extraction. In general, however, the choice of discretization method can have a large effect on activity detection performance. A related concern is the need for prespecified instance features. Such features must be specified for each newly encountered combination of target activity and sensing modality. One possible solution to this problem is to use feature learning methods such a neural networks to learn a representation of the raw sensor signal. One final limitation is that supervised learning of CRF models requires fully labeled data, which is expensive to gather. Further, if there are errors in the observed label structures, this may result in reduced detection performance. In the next chapter we address this problem using the framework of weakly supervised learning.

CHAPTER 4

LEARNING EVENT DETECTION MODELS FROM TEMPORALLY IMPRECISE LABELS

To this point, we have assumed that fully and accurately labeled data is available for model training; however, one of the core challenges facing mHealth researchers is the collection of precisely labeled data. In machine learning, this problem has been addressed through a number of alternative learning frameworks where labels can be obtained at lower cost, including frameworks for learning from lower volumes of labeled data (semi-supervised learning [8], positive unlabeled learning [33], active learning [60], etc.), and frameworks for learning from lower-quality labels (multiple instance learning [36], learning with label proportions [21] etc.). In this chapter, we consider a new low-quality label learning problem: learning discrete-time detection models from continuous-time temporally imprecise labels.

While many mHealth detection methods, including the ones discussed in this work, assume a temporally discretized input (e.g. [68, 1, 3]), annotations of events or activities are generally recorded in continuous time. Such annotations may come from a number of sources. In the lab, researchers may record the timestamps of events via live observation (e.g. [1]) or by annotating video or audio recordings of subjects (e.g. [68]). In the field, subjects may be asked to self-report activities or researchers may provide annotations based on a second, higher quality sensor such as a front-facing camera or microphone (e.g. [3]). In all cases, it is necessary to align the continuous-time annotations to the discrete input sequence in order to generate labels that can be used for training discrete time detection models. Specifically, let z be a potentially imprecise continuous time annotation of an activity of interest. Then, before using

it to learn a discrete time detection model, it is necessary to decide which instance i annotation z corresponds to.

This problem has resulted in a wide variety of (mostly ad hoc) alignment procedures including rule-based [38, 68] and manual [1, 56] alignment procedures. Rule-based alignment procedures map continuous time annotations to instances based on simple rules. For example, we may map a continuous time annotation to the temporally nearest instance in the discrete input sequence. This can be done rapidly, but may result in labeling errors when the continuous time annotations are imprecise. Manual alignment procedures, on the other hand, require an annotator to visualize the input signal and make decisions about how continuous time annotations map onto the discrete input sequence. Such procedures are time consuming and may also result in labeling errors when events are not apparent to the annotator in the input signal.

The primary intuition encoded in our approach is that observations are recorded "close" in time to the true label structures they correspond to (this intuition will be formalized in the following sections). One simple rule-based alignment procedure that encodes this intuition is to map each observation to the temporally closest time point in the discrete input sequence; however, this transformation will convert imprecision in the observation times into errors in the label values. We will refer to this as the *naive alignment* strategy. Our framework seeks to improve on the naive alignment strategy by explicitly modeling the observation process and allowing for temporal imprecision.

A further problem with applying supervised learning to discrete time aligned annotations is that supervised learning methods cannot account for missing or spurious annotations. In particular, even an alignment procedure that perfectly corrects for temporal imprecision cannot add or remove annotations. Our framework seeks to address this problem by explicitly modeling the probability of observing true activities.

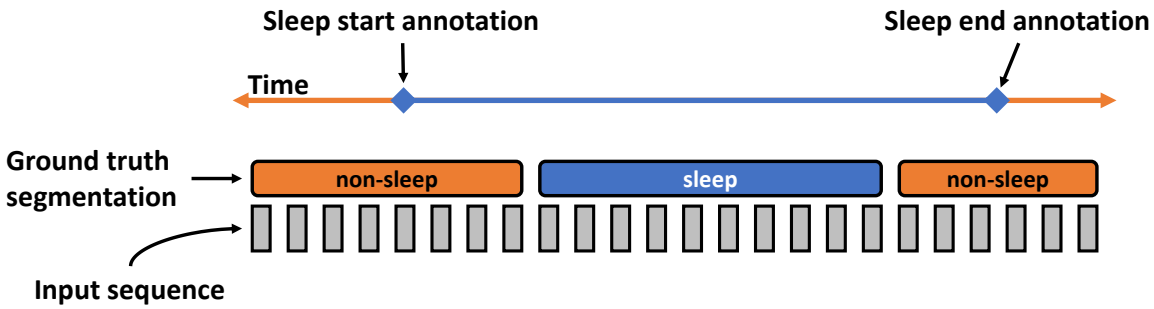


Figure 4.1: An example of an input sequence, ground truth labeled segmentation, and imprecise annotation for the beginning and end of sleep.

We take a weakly supervised learning approach to this problem. In particular, we consider the unaligned continuous annotations to be weak supervision for the detection problem of interest. Unfortunately, existing approaches to weakly supervised learning (discussed in Section 2.2) do not cleanly solve this problem. The closest set of approaches are based around multi-instance learning; however, applying multi-instance learning to this problem requires discretizing the continuous-time annotations in some way, resulting in a loss of information. We evaluate one such approach in Section 4.2.4.

The primary contribution of this chapter is a framework for estimating the parameters of discrete-time detection models from imprecise continuous-time annotations. As in Chapter 3, we begin with a multivariate time series and assume that it has been temporally discretized to produce a temporal input sequence. Unlike in Chapter 3, we assume that supervision for learning is provided in the form of timestamps corresponding approximately to the unobserved true label structure. Figure 4.1 shows an example where the annotations approximately mark the beginning and end of sleep.

Our framework augments a discriminative detection model with a probabilistic model of the annotation process, treating the true label structure as a latent variable. A graphical model for this augmentation is shown in Figure 4.2. In this model, $\mathbf{x}$ represents the input sequence, $\mathbf{y}$ represents the unobserved label structure, $\mathbf{z}$ rep-

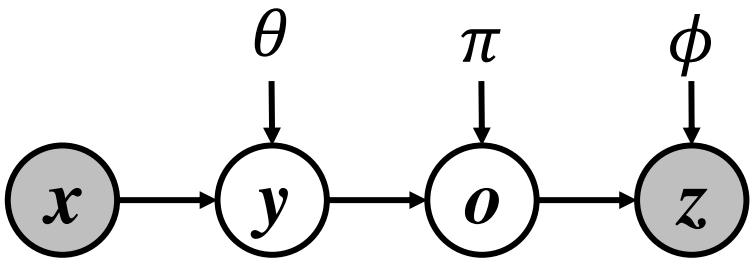


Figure 4.2: Graphical model for the proposed weak supervision framework. **z** represents the observed continuous-time annotations, and **o** represents the unobserved alignment between instances in **x** and annotations in **z**. Shaded variables are observed during model training.

resents the observed continuous-time annotations, and **o** represents the unobserved alignment between instances in **x** and annotations in **z**.

This framework allows us to marginalize out the proper alignment of annotations to discrete time instances from the data, obviating the need for ad hoc alignment methods. Further, by assuming an imprecise observation process, this framework allows us to more accurately estimate models from temporally imprecise annotations such as those generated by self-report.

The second contribution of this chapter is a method for integrating continuous-time annotations with signals from wearable sensors at test time. Self-report and wearable sensors are not mutually exclusive technologies and can potentially be used together in the same study. In such a study, predictions from a detection model can be combined with the self-reported activities hopefully resulting in more accurate predictions than using either separately. In this setup, one can think of the detection model as denoising the potentially imprecise self-reported annotations. This denoising can naturally be framed as posterior inference in the model shown in Figure 4.2.

In Section 4.2, we present a version of the proposed weak-supervision framework for independent instance labeling and evaluate it on a smoking detection problem. In Section 4.3, we extend this framework to the semi-Markov CRF and the segmentation models presented in Chapter 3. We evaluate this extension on sleep detection

and smoking detection problems. Also in Section 4.3, we apply inference pruning methods from Section 3.6 to the proposed weakly supervised structure prediction framework and evaluate them on smoking data. Finally, in Section 4.4, we show how this model can be used to integrate imprecise observations with sensor data at test time, improving on the accuracy of either one individually.

4.1 Notation

As in previous chapters, we assume that the input data consists of N multivariate time series that we will call **sessions**. Each session contains a set of time-aligned signals gathered from one or more sensors. Seperate sessions may correspond to data from different subjects data, or to data from the same subject collected at different times. We assume that each session n has been discretized into a sequence of L_n potentially overlapping sub-windows and that a feature vector $\mathbf{x}_{ni} \in \mathbb{R}^D$ has been extracted for each sub-window i. We refer to each sub-window i as an **instance**. Further, each instance i in session n is associated with a timestamp t_{ni} which may correspond to the start, end, or other point of interest associated with instance i. We refer to the complete sequence of feature vectors $\mathbf{x}_n = \{\mathbf{x}_{ni}\}_{i=1}^{L_n}$ as the **input sequence** and the complete sequence of timestamps $\mathbf{t}_n = \{t_{ni}\}_{i=1}^{L_n}$ as the **timestamp sequence**. Where it does not cause ambiguity, we will drop the session index n. We use the notation $\mathbf{x}_{j:k} = \{\mathbf{x}_i\}_{i=j}^{k}$ to refer to the subsequence of $\mathbf{x}$ beginning at j and ending at k (this applies to any sequence).

4.2 Independent Classification Models

We start by addressing the problem of learning independent instance labeling models from temporally imprecise observations. While Chapter 3 focused on structured prediction for mHealth detection problems, independent models are commonly used when the goal is to detect short duration events such as bites of food [1, 56, 68, 38]. Ac-

quiring accurate labels for short duration events is a particularly challenging problem in mHealth, often requiring video or live observation and time consuming annotation procedures. For example, in the mPuff [1] and puffMarker [56] smoking datasets, subjects smoked while under direct observation and smoking puffs were recorded by research staff using a mobile phone. Due to the short duration of smoking puffs, even minor imprecision in the annotations may result in instance label errors if a naive alignment procedure is used. As an alternative, we propose to learn an instance labeling model directly from timestamps corresponding to observations of positive instances, obviating the need for time consuming annotation or alignment processes and allowing for the correction of false positives and false negatives. In the remainder of this section, we introduce the proposed framework and evaluate it on real smoking data with temporally imprecise observations.

4.2.1 Weak Supervision Framework

In this section, our goal is to learn a function that maps instance features to binary labels. Let $y_{ni} \in \{0, 1\}$ be the binary label variable for instance i in session n. We assume the labels y_{ni} are not directly observed in the training data. Instead, we observe a length M_n sequence $\mathbf{z}_n = \{z_{nm}\}_{m=1}^{M}$ of **observations** where each observation z_{nm} is a timestamp corresponding roughly to the time at which a positive instance occurred. Our goal, then, is to learn a standard instance-level classification function $f : \mathbb{R}^D \rightarrow \{0, 1\}$ from a data set $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{t}_n, \mathbf{z}_n)\}_{n=1}^{N}$ consisting of the input sequences, the instance timestamps, and the observation timestamp sequences.

To map between the unobserved label sequence $\mathbf{y}$ and the observation sequence $\mathbf{z}$, let $\mathbf{o} = \{o_i\}_{i=1}^{L}$ be a sequence of latent binary variables where $o_i = 1$ if and only if instance i is associated with an observation. We call $\mathbf{o}$ the **observation indicator** sequence. We make two assumptions about $\mathbf{o}$. First we assume that $\sum_i o_i = M$, that is, the sum of the observation indicators equals the number of observations, which

is necessary for our interpretation of $\mathbf{o}$. Second, we make the following simplifying assumption:

Assumption 1. *If instance j occurs after instance i in the input sequence, then an observation timestamp associated with instance j must occur after an observation timestamp associated with instance i.*

Under these two assumptions, $\mathbf{o}$ defines a matching between instances in the input sequence and observations in the observation sequence.

Our proposed framework includes three components that can be chosen independently: a probabilistic base classifier $p_\theta(y_i|\mathbf{x})$, an observation indicator distribution $p_\pi(o_i|y_i)$, and an observation timestamp density $p_\phi(z_m|t_i)$.

The base classifier is the distribution we are primarily interested in estimating. For the base classifier, we assume a differentiable discriminative classifier of the form $p_\theta(y_i|\mathbf{x})$ with parameters θ. Any discriminative probabilistic classifier where the label variables y_i are probabilistically independent given the features $\mathbf{x}$ can be used. Such models include logistic regression [23] and kernel logistic regression [87], as well as multi-layer feedforward neural networks [22], convolutional neural networks [27], and recurrent neural networks [17] when used with logistic/softmax output layers. We focus on the case of logistic regression to demonstrate the framework.

The observation indicator distribution $p_\pi(o_i|y_i)$ models the probability that instance i is associated with an observation given its label y_i. This distribution allows for the occurrence of false positive observation (observations that do not correspond to a true positive instance) and missed observations (positive instances that are not associated with an observation). In settings where there are no such annotation errors, this can be set to a deterministic distribution. Finally, the observation timestamp density $p_\phi(z_m|t_i)$ models the observation timestamp z_m given the instance timestamp t_i with which it is associated. The choices for these distributions are domain specific. For example, in section 4.2.4 we evaluate a model where $p_\pi(o_i|y_i)$ is a Bernoulli dis-

tribution and $p_\phi(z_m|t_i) = \mathcal{N}(z; \mu + t, \sigma^2)$ is a normal distribution centered at t plus an offset μ. With these distributions, we can now write the observation generation process as shown below:

$M \leftarrow 0$
for $i = 1, ..., L$ **do**
    $y_i \sim p_\theta(y_i|\mathbf{x})$
    $o_i \sim p_\pi(o_i|y_i)$
    if $o_i = 1$ **then**
        $M \leftarrow M + 1$
        $z_M \sim p_\phi(z|t_i)$

This generative process asserts that each instance label y_i is first sampled from the base classifier $p_\theta(y_i|\mathbf{x})$. Then, each instance either generates an observation or not according to $p_\pi(o_i|y_i)$. Finally, if instance i does generate an observation, an observation timestamp is sampled from $p_\phi(z_m|t_i)$. The variable M counts the number of generated observations. We note that additional structure could be encoded into the label observation process at the potential cost of higher inference complexity.

We can now specify the individual joint distributions over the label sequence $\mathbf{y}$, the observation indicator sequence $\mathbf{o}$, and the observation sequence $\mathbf{z}$ as shown below. We define $i(m) = \min\{i | \sum_{j=1}^{i} o_j = m\}$ as the function mapping observation m to the instance that generated it.

$$p_\theta(\mathbf{y}|\mathbf{x}) = \prod_{i=1}^{L} p_\theta(y_i|x_i) \tag{4.1}$$

$$p_\pi(\mathbf{o}|\mathbf{y}) = \prod_{i=1}^{L} p_\pi(o_i|y_i) \tag{4.2}$$

$$p_\phi(\mathbf{z}|\mathbf{o}, \mathbf{t}) = \prod_{m=1}^{M} p_\phi(z_m|t_{i(m)}) \tag{4.3}$$

The complete joint distribution over $\mathbf{y}$, $\mathbf{o}$, and $\mathbf{z}$ is shown in Equation 4.4 where $\psi = \{\theta, \phi, \pi\}$ is the complete set of model parameters.

$$p_\psi(\mathbf{y},\mathbf{o},\mathbf{z}|\mathbf{x},\mathbf{t}) = p_\theta(\mathbf{y}|\mathbf{x})p_\pi(\mathbf{o}|\mathbf{y})p_\phi(\mathbf{z}|\mathbf{o},\mathbf{t}) \tag{4.4}$$

The label sequence and observation indicator sequence $\mathbf{y}$ and $\mathbf{o}$ are not observed during learning, but we can marginalize them out of the model as seen in in Equation 4.5. The marginalization operation is expressed in terms of a sum over the support sets $\mathcal{O} = \{o|o \in \{0,1\}^L, \sum_{i=1}^{L} o_i = M\}$ and $\mathcal{Y} = \{0,1\}^L$ of $\mathbf{o}$ and $\mathbf{y}$ respectively.

$$p_\psi(\mathbf{z}|\mathbf{x},\mathbf{t}) = \sum_{\mathbf{y}\in\mathcal{Y}} \sum_{\mathbf{o}\in\mathcal{O}} p_\psi(\mathbf{y},\mathbf{o},\mathbf{z}|\mathbf{x},\mathbf{t}) \tag{4.5}$$

In the next sections, we derive computationally efficient inference and learning algorithms for maximizing the log marginal likelihood of the data. We note that once the model is learned, the observation count distribution and the timestamp noise distribution can be discarded and instances can be classified based only on $p_\theta(y_i|\mathbf{x})$ as is typically done in independent classification.

4.2.2 Learning

To learn the proposed model, we will maximize the log marginal likelihood $\mathcal{L}(\theta,\phi,\pi|\mathcal{D})$ where θ are the base classification model parameters, ϕ are the observation noise model parameters, π are the observation indicator model parameters, and $\mathcal{D} = \{(\mathbf{x}_n,\mathbf{t}_n,\mathbf{z}_n)\}_{n=1}^N$ consists of the observed data for each session. This objective function is defined below:

$$\begin{aligned}\mathcal{L}(\theta,\phi,\pi|\mathcal{D}) &= \sum_{n=1}^{N} \log p_\psi(\mathbf{z}_n|\mathbf{x}_n,\mathbf{t}_n) \\ &= \sum_{n=1}^{N} \log \left(\sum_{\mathbf{y}_n\in\mathcal{Y}_n} \sum_{\mathbf{o}_n\in\mathcal{O}_n} p_\psi(\mathbf{y}_n,\mathbf{o}_n,\mathbf{z}_n|\mathbf{x}_n,\mathbf{t}_n) \right)\end{aligned} \tag{4.6}$$

We optimize this objective using standard gradient methods[1]. In this section, we consider the gradient equations for each of the three parameter groups: θ, π, and ϕ.

We first consider the gradient of $\log p_\psi(\mathbf{z}|\mathbf{x}, \mathbf{t})$ with respect to the base classifier parameters θ. For brevity, we drop the dependence on the session index n.

$$\nabla_\theta \log p_\psi(\mathbf{z}|\mathbf{x}, \mathbf{t}) = \sum_{i=1}^{L} \mathbb{E}_{p_\psi(y_i|\mathbf{z},\mathbf{x},\mathbf{t})}[\nabla_\theta \log p_\theta(y_i|\mathbf{x}_i)] \tag{4.7}$$

This gradient has the form of a sum of expected gradients for individual instances where the expectation is taken with respect to the marginal posterior distribution $p_\psi(y_i|\mathbf{z}, \mathbf{x}, \mathbf{t})$. Assuming that the gradient of the base classifier can be computed efficiently, the complexity of computing the gradient of the proposed likelihood with respect to θ depends only on the complexity of computing the posterior marginal $p_\psi(y_i|\mathbf{z}, \mathbf{x}, \mathbf{t})$. In the next section, we will show how this gradient can be computed efficiently.

Next, we give the gradient with respect to the noise model parameters ϕ.

$$\nabla_\phi \log p_\psi(\mathbf{z}|\mathbf{x}, \mathbf{t}) = \sum_{m=1}^{M} \mathbb{E}_{p_\psi(i(m)|\mathbf{z},\mathbf{x},\mathbf{t})}[\nabla_\phi \log p_\phi(z_m|t_{i(m)})] \tag{4.8}$$

where $i(m)$ (defined above) maps m to the instance it is associated with. We can again see that this gradient has the form of a sum of expectations for individual event time stamps. In this case, the sum is over each observed time stamp and the expectation is taken with respect to the posterior marginal distribution $p_\psi(i(m)|\mathbf{z}, \mathbf{x}, \mathbf{t})$, which gives the distribution over which instance corresponds to observation m. Assuming that the gradients of $\log p_\phi(z_m|t_i)$ can be computed efficiently, the complexity of the gradient computation depends on the complexity of computing the posterior marginal $p_\psi(i(m)|\mathbf{z}, \mathbf{x}, \mathbf{t})$.

[1]In practice, we use L-BFGS-B as implemented in SciPy

Finally, we give the gradient with respect to the observation indicator parameters, π.

$$\nabla_\pi \log p_\psi(\mathbf{z}|\mathbf{x}, \mathbf{t}) = \sum_{i=1}^{L} \mathbb{E}_{p_\psi(o_i, y_i|\mathbf{z}, \mathbf{x}, \mathbf{t})}[\nabla_\phi \log p_\pi(o_i|y_i)] \tag{4.9}$$

Once again, the gradient has the form of a sum of expected gradients. In this case the expectation is taken with respect to the posterior marginal $p_\psi(o_i, y_i|\mathbf{z}, \mathbf{x}, \mathbf{t})$ which gives the distribution over the label of instance i and whether or not it is associated with an observation.

The complete gradient system required to optimize all parameters of the complete joint distribution is shown below in Equations 4.10, 4.11, and 4.12.

$$\nabla_\theta \mathcal{L}(\theta, \phi, \pi|\mathcal{D}) = \sum_{n=1}^{N} \nabla_\theta \log p_\psi(\mathbf{z}_n|\mathbf{x}_n, \mathbf{t}_n) \tag{4.10}$$

$$\nabla_\phi \mathcal{L}(\theta, \phi, \pi|\mathcal{D}) = \sum_{n=1}^{N} \nabla_\phi \log p_\psi(\mathbf{z}_n|\mathbf{x}_n, \mathbf{t}_n) \tag{4.11}$$

$$\nabla_\pi \mathcal{L}(\theta, \phi, \pi|\mathcal{D}) = \sum_{n=1}^{N} \nabla_\pi \log p_\psi(\mathbf{z}_n|\mathbf{x}_n, \mathbf{t}_n) \tag{4.12}$$

One difficulty with learning the parameters of this model is that posterior distribution over the parameters $p(\theta, \pi, \phi|\mathcal{D})$ is multi-modal. In particular, analysis using the puffMarker data and model described in section 4.2.4 suggests that the posterior distribution over π is bimodal. Samples drawn from $p(\pi|\mathcal{D})$ using Hamiltonian Monte Carlo (HMC)[2] [44] are shown in Figure 4.3. In one of the modes, the model treats all observations as false positives and in the other it treats most of the observations a true positives. This makes training starting from a random location unstable as

[2]We used the pyhmc package (`https://pythonhosted.org/pyhmc/index.html` with 10 steps per sample and a step size of 0.001 resulting in an acceptance rate of approximately 0.87. We drew 5000 total samples, but dropped the first 2000 samples as burn-in and used only every 100th sample to avoid autocorrelation in the samples.

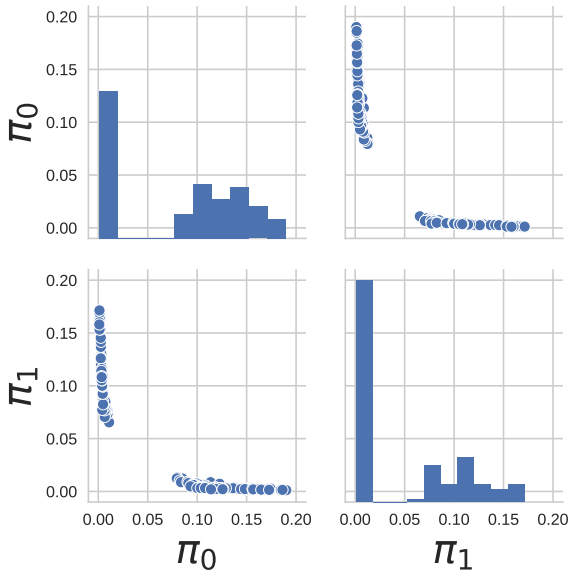


Figure 4.3: Samples of $\pi_0 = p_\pi(o_i = 1|y_i = 0)$ and $\pi_1 = p_\pi(o_i = 1|y_i = 1)$ from the posterior distribution over parameters $p(\pi|\mathbf{x}, \mathbf{t}, \mathbf{z})$.

Algorithm 2 Posterior Inference Dynamic Program for Temporally Imprecise Labels

Inputs: $\theta, \phi, \pi, \mathbf{x} \in \mathbb{R}^{LxD}$, $\mathbf{t} \in \mathbb{R}^{L}$, $\mathbf{z} \in \mathbb{R}^{M}$
Let $\boldsymbol{a} \in \mathbb{R}^{LxM}$, $\boldsymbol{b} \in \mathbb{R}^{LxM}$ $a(0,0) \leftarrow 1$
for $i = 1, ..., L$ **do**
    $a(i,0) \leftarrow \sum_{y \in \{0,1\}} p_\theta(y_i = y|\mathbf{x}) p_\pi(o_i = 0|y) a(i-1,0)$
    for $m = 1, ..., M$ **do**
        $a(i,l) \leftarrow \sum_{y \in \{0,1\}} \sum_{o \in \{0,1\}} p_\theta(y_i = y|\mathbf{x}) p_\pi(o_i = o|y) p_\phi(z_m|t_i)^o a(i-1, m-o)$
Return $\boldsymbol{a}$

it may fall into either mode. We avoid this by initially constraining the observation indicator distribution such that $p_\pi(o_i = 1|y_i = 1) = 1$ and $p_\pi(o_i = 0|y_i = 0) = 0$, training to convergence, and then relaxing the constraint on $p_\pi(o_i|y_i)$. This initialization procedure allows us to start learning within the desirable basin of attraction. We next turn to the problem of efficiently computing the log marginal likelihood and its gradients.

4.2.3 Inference

The primary computational challenge of this learning procedure is calculating the log marginal likelihood. This can be done exactly using the dynamic program shown

in Algorithm 2. An entry in the dynamic programming table α has the following interpretation: $\alpha(i, m)$ is the probability that the input subsequence $\mathbf{x}_{1:i}$ generated the observation subsequence $\mathbf{z}_{1:m}$. Or, written mathematically:

$$\alpha(i, m) = p_\psi(\mathbf{z}_{1:m}|\mathbf{x}_{1:i}, \mathbf{t}_{1:i}) \tag{4.13}$$

The marginal likelihood is given by $p_\psi(\mathbf{z}|\mathbf{x}, \mathbf{t}) = a(L, M)$. This algorithm has complexity $\mathcal{O}(LM)$ where L is the length of the input sequence and M is the length of the observation sequence. We use reverse-mode automatic differentiation [4] to compute the gradients of this dynamic program with respect to the parameters[3].

4.2.4 Experiments

In this section, we present an evaluation of the proposed framework under both synthetic and real temporal noise conditions.

4.2.4.1 Datasets

To evaluate the predictive performance of our model, we used two real mobile health datasets: mPuff [1] and puffMarker [56]. Details on these datasets and the feature preprocessing are given in Section 2.3. In both datasets, sessions were discretized into respiration cycles (i.e. a single inhalation/exhalation cycle) and the goal is to label each respiration cycle as being a smoking puff or not. Both data sets had the originally recorded positive event timestamps manually aligned to the instance sequence using a visualization procedure. We considered these manually aligned labels to be ground truth for both datasets.

We evaluated the proposed framework when trained on both synthetic (Experiment 1) and real (Experiment 2) observation timestamps. To test the behavior of our

[3]We performed automatic differentiation using Theano (`http://deeplearning.net/software/theano/`)

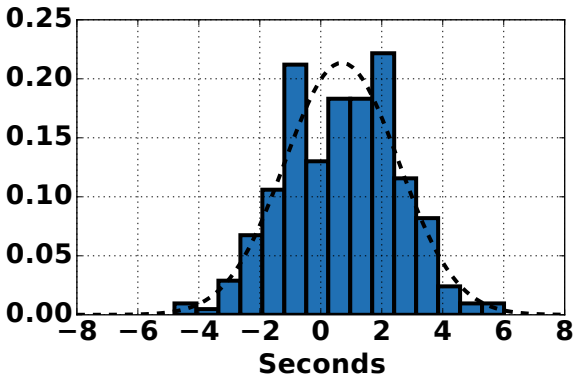


Figure 4.4: The marginal distribution of the difference between the true and observed time stamps for positive instance in the puffMarker data. The dashed lined shows a Normal distribution fit to this data.

model under varied amounts observation noise, we generated synthetic observation timestamps for both datasets. We used the following noise model, which conditions on the ground truth instance timestamps and labels:

$$p(o_i = 1|y_i = 1) = 1 - p(o_i = 1|y_i = 0) = \pi, \ \forall\, i \tag{4.14}$$

$$p(z_m|t_{i(m)}) = \mathcal{N}(z_j; t_{i(m)}, \sigma^2) \tag{4.15}$$

We varied $\pi \in (0, 1)$ and $\sigma \in \mathbb{R}^+$ to simulate different amounts of missed/spurious observations and timestamp noise respectively. We chose a normal noise distribution for the synthetic observation process because it approximately matches the noise distribution observed in the real puffMarker data, as shown in Figure 4.4.

The puffMarker dataset also includes the original imprecise smoking puff time stamps as recorded by the observer during data collection. This allowed us to conduct experiments where we trained on temporally imprecise smoking puff timestamps, but tested the learned classifier on held out sessions with ground truth labels provided by the manual alignment process.

4.2.4.2 Models

In the following experiments, we used logistic regression as the base classifier in our model (LR-WS). We placed a zero-mean Gaussian prior on θ (i.e. ℓ_2 regularization). As our observation count model, we used a simple Bernoulli distribution $p_\pi(o_i = 1|y_i = c) = 1 - p_\pi(o_i = 0|y_i = c) = \pi_c$ for $\pi_c \in (0, 1)$. This allows the model to account for both false positives and false negatives. We placed a beta prior on π. Finally, we used the following normal distribution as our observation noise model:

$$p_\phi(z_j|t_i) = \mathcal{N}(z_j; t_i + \mu_k, \sigma_k^2) \tag{4.16}$$

where $\phi = \{\mu, \sigma\}$. We placed a standard normal prior on μ and an inverse-Gamma prior with shape $\alpha = 1$ and scale $\beta = 1$ on σ^2. We found the LR-WS model to be relatively insensitive to the setting of these hyperparameters so weakly-informative default values were chosen.

We compared this instance of our proposed framework with logistic regression models trained using different transformations of the available observation sequences. In particular, we experimented with a logistic regression model learned with observation timestamps manually aligned to instances (LR-HA). If the manual alignment process results in *no* instance label errors, this model can be thought of as producing best-case results given the discretization and feature extraction methods. We also experimented with the naive alignment strategy where we assign a positive label to the instance that is closest in time to each observation timestamp (LR-NV). This approach treats the temporally imprecise observation timestamps as if they were correct, potentially resulting in label noise.

Finally, we compared our model to a traditional multi-instance learning strategy that produces an instance-level classifier (LR-MI). In particular, we adapted the MI-SVM training algorithm from Andrews et al. [2] to train logistic regression models.

This algorithm works by forming *bags* of instances which are labeled as either positive or negative. A negative bag label indicates that the bag contains only negative instances and a positive bag label indicates that at least one of the instances is positive. For a given bag size, B, we formed bags by segmenting the base sequence into non-overlapping segments of length B. We generated bag labels by applying the naive alignment strategy described above and labeling a bag as positive if at least one observation timestamp fell inside of the bag.

The MI-SVM algorithm alternates between picking a representative instance from each positive bag, called a *witness*, and training a classifier on the witnesses plus the negative instances. The witnesses are chosen to minimize the classifier's loss function. To our knowledge, this strategy has never been applied to logistic regression. In the case of logistic regression, this alternating algorithm results in a non-decreasing objective and therefore converges to a local optima. The hyperparameters for this model include the bag size B and the ℓ_2 regularization strength.

4.2.4.3 Train and Test Procedures

We evaluated all models using a 10-fold cross-validation procedure where the folds were generated across sessions. We tuned all hyper-parameters to maximize F_1 score using a further 10-fold cross-validation procedure on the training set. This is equivalent to assuming a small amount of labeled data is available for validation. ℓ_2 regularization parameters for each model were tuned across an logarithmic grid while the parameters for the beta prior on π and the bag size for the multi-instance models were tuned on a linear grid. Results for all models are presented in terms of F_1 score, which was chosen due to the heavy class imbalance and to highlight the performance on the class of interest, smoking puffs. Performance results were averaged across all folds. For all experiments involving synthetic noise, results were further averaged across data generated using two different seeds.

4.2.4.4 Experiment 1: Performance Under Varying Noise Conditions

We evaluated how robust the proposed model is to varied observation noise conditions by testing the predictive performance of all models on the mPuff and puffMarker datasets with synthetic observation timestamps generated by adding noise to the timestamps of true positive instances. Figures 4.5 (a) and (b) show the performance in terms of F_1 of these models on the mPuff and puffMarker datasets respectively with varied observation timestamp noise, but fixed observation indicator noise. For these plots, we varied the standard deviation (σ) of the synthetic noise from 0.25 to 5.0 with the observation count probability ($p(o_i = 1|y_i = 1) = \pi$) fixed at 1. While the performance of each model degrades as the amount of noise increases, the performance of the proposed model (LR-WS) degrades noticeably slower than the traditional multi-instance method (LR-MI), which in turn only slightly outperforms the naive alignment strategy (LR-NV) at this task. To understand why the performance of the naive and multi-instance methods degrade so much, Figure 4.5 (c) shows the proportion of true positive labels that are assigned positive labels by the naive alignment procedure on the mPuff and puffMarker datasets across the same range of noise standard deviations. Even at $\sigma = 2.0$, only 65% of true positives retain a positive label under the naive alignment strategy. As expected, the performance of the baseline methods track this plot very closely.

Figure 4.5 (d) and (e) show the performance of the logistic regression based models when σ is kept fixed at 1, but π is varied between 0.7 and 1. $\sigma = 1$ was chosen because it matches the standard deviation of the empirical observation noise distribution of the puffMarker dataset. Again, the performance of the multi-instance and naive alignment methods degrade much more quickly than the proposed method as noise is added.

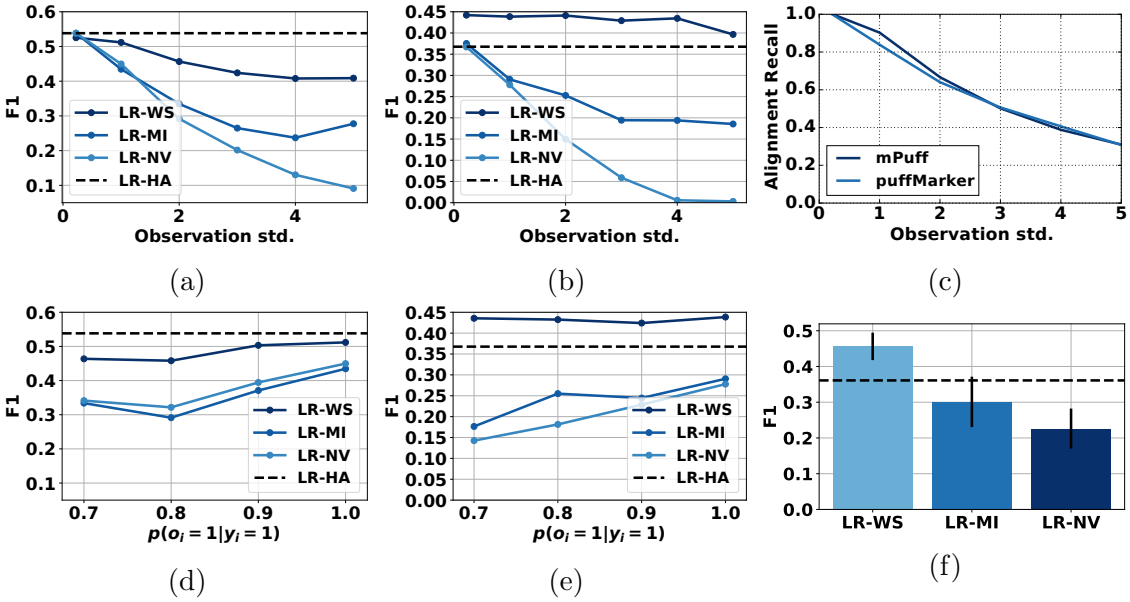


Figure 4.5: Figures (a) and (b) show the prediction performance for all models when varied amounts of synthetic noise is added to the hand aligned labels of the mPuff and puffMarker datasets respectively. Figure (c) shows the recall of the labels generated by the naive alignment strategy. Figures (d) and (e) show the predictive performance for all models when different proportions of observations are dropped from the observation sequence on the mPuff and puffMarker datasets. Figure (f) shows the F_1 performance of all models on the puffMarker dataset trained on the real unaligned observation sequence. The dashed line corresponds to LR-HA.

4.2.4.5 Experiment 2: Performance on Real Timestamps

We also evaluated the performance of all models on the puffMarker dataset using the original positive event timestamps during training. The predictive performance in terms of F_1 is shown in Figure 4.5 (f). The LR-WS model substantially outperforms both the multi-instance and naive methods. The improvement of LR-WS over LR-NV and LR-MI is statistically significant at the $p = 0.001$ level using a paired t-test with Bonferroni correction.

As in the synthetic experiments on the puffMarker dataset, the LR-WS model outperforms a logistic regression model trained on the hand aligned labels. One possible explanation for this behavior is that there is a non-trivial amount of annotation error that our model is correcting for. Another possible explanation for this result is that the there is a non-trivial amount of class overlap in the instance feature space. Our model allows for the possibility that there are false negatives in the labels and so it is able to get around this overlap by treating positive looking negative examples as positive, whereas the other models must treat them as negative. This is supported by the observation that the precision of our model is slightly lower than the two baselines, but the recall is much higher.

4.3 Segmentation Models

As demonstrated in Chapter 3, we can often achieve performance improvements by applying structured prediction methods to mHealth detection problems. In this section we extend the weak supervision framework presented in the previous section to the types of segmentation models presented in Chapter 3. In particular, we consider the problem of learning segmentation-based CRF models when supervision is provided in the form of timestamps roughly corresponding to the transitions between segments of different types. Observations of this type are particularly common in field data studies where either data volumes preclude finer-grained annotation (e.g.

[3, 68]) or subjects are asked to self-report activities (e.g [74]). For example, in Vaizman et al. [74], subjects were asked to report the times they started and finished activities of certain types, such as sleeping. While the activities of interest are generally much longer than the short duration events discussed in the previous section, the imprecision in annotations generated by self-report may be correspondingly larger. In the remainder of this section, we present the modified weak supervision framework, inference dynamic program, and evaluations of this framework on semi-Markov CRF and heterogeneous segmentation models (see Section 3.4) using both semi-synthetic and real datasets.

4.3.1 Model

In this section, our goal is to learn a model that produces a labeled segmentation of the input sequence $\mathbf{x}$. As in previous sections, we represent such a segmentation as a sequence of segments $\mathbf{y} = \{y_s\}_{s=1}^{S}$ where each segment $y_s = (c_s, j_s, k_s)$ is a tuple containing a label $c_s \in \mathcal{C}$, a start position $j_s \in \{1, ..., L\}$, and an end position $k_s \in \{1, ..., L\}$. To ensure only valid segmentations, we assume $j_1 = 1$, $k_S = L$, and $k_s = j_{s+1}$ for all $1 \leq s \leq S - 1$. Our goal, then, is to learn the distribution $p_\theta(\mathbf{y}|\mathbf{x}, \mathbf{t})$. For purposes of presentation, we will parameterize this distribution as a semi-Markov CRF, but we will show later how this framework can also be applied to the heterogeneous segmentation models from Chapter 3. The details of the semi-Markov CRF are presented in Section 2.1.

As before, we assume that the ground-truth segmentation $\mathbf{y}$ is not observed during training. Instead, we observe a sequence of **observations** $\mathbf{z} = \{z_m\}_{m=1}^{M}$ where each observation z_m is a timestamp corresponding to a particular transition between segments. For example, each z_m may be the time a subject reported going to sleep, marking the approximate start of a sleep segment. For ease of exposition, we will assume that there is only one type of observation and will later generalize to multiple

observation types. To map between the unobserved segmentation $\mathbf{y}$ and the observation sequence $\mathbf{z}$, let the **observation indicator sequence** $\mathbf{o} = \{o_i\}_{i=1}^{L}$ be a sequence of latent binary variables where $o_i = 1$ if and only if instance i is associated with an observation. Under the assumption that observations are recorded in the order they actually occurred (Assumption 1 in Section 4.2) and $\sum_i o_i = M$, $\mathbf{o}$ defines a matching between instances in the input sequence and observations in the observation sequence.

We model the observation sequence using three components. The base segmentation model $p_\theta(\mathbf{y}|\mathbf{x}, \mathbf{t})$ is the semi-Markov CRF model whose parameters we are interested in estimating. The observation indicator distribution $p_\pi(o_i|y_s, c_{s-1})$ models the probability that instance i is associated with an observation given the segment it is contained in and the label of the previous segment. Finally, the observation timestamp density $p_\phi(z_m|t_i)$ models the timestamp of an observation z_m given the timestamps t_i with which it is associated. For example, we may use a simple Bernoulli distribution for $p_\pi(o_i|y_s, c_{s-1})$ and a normal distribution centered at t_i for $p_\phi(z_m|t_i)$. With these distributions, we can now write the observation generation process as shown below:

$M \leftarrow 0$
$\mathbf{y} \sim p_\theta(\mathbf{y}|\mathbf{x})$
for $s = 1, ..., S$ **do**
    for $i = j_s, ..., k_s$ **do**
        $o_i \sim p_\pi(o_i|y_s, c_{s-1})$
        if $o_i = 1$ **then**
            $M \leftarrow M + 1$
            $z_M \sim p_\phi(z_m|t_i)$

This generative process asserts that a complete segmentation is first sampled according to the semi-Markov CRF model. Next, each instance either generates an observation or not according to $p_\pi(o_i|y_s, c_{s-1})$. Finally, if instance i does generate an observation, an observation timestamp is sampled from $p_\phi(z_m|t_i)$. The variable M counts the number of generated observations. The joint model implied by this generative process is given in Equation 4.17 where the set of all parameters in the model is

$\omega = \{\theta, \pi, \phi\}$. The distributions $p_\pi(\mathbf{o}|\mathbf{y})$ and $p_\phi(\mathbf{z}|\mathbf{o}, \mathbf{t})$ are defined in Equations 4.18 and 4.19 where $i(m) = \min\{i | \sum_{j=1}^{i} o_j = m\}$ is the function mapping observation m to the instance that generated it, as before.

$$p_\omega(\mathbf{z}, \mathbf{y}, \mathbf{o}|\mathbf{x}, \mathbf{t}) = p_\theta(\mathbf{y}|\mathbf{x}) p_\pi(\mathbf{o}|\mathbf{y}) p_\phi(\mathbf{z}|\mathbf{o}, \mathbf{t}) \tag{4.17}$$

$$p_\pi(\mathbf{o}|\mathbf{y}) = \prod_s \prod_{i=j_s}^{k_s} p_\pi(o_i|y_s, c_{s-1}, i) \tag{4.18}$$

$$p_\phi(\mathbf{z}|\mathbf{o}, \mathbf{t}) = \prod_{m=1}^{M} p_\phi(z_m|t_{i(m)}) \tag{4.19}$$

4.3.2 Learning

To learn the parameters of this model, we maximize the log marginal likelihood $\mathcal{L}(\omega|\mathcal{D})$:

$$\mathcal{L}(\omega|\mathcal{D}) = \sum_{n=1}^{N} \log p_\omega(\mathbf{z}_n|\mathbf{x}_n, \mathbf{t}_n) \tag{4.20}$$

$$p_\omega(\mathbf{z}|\mathbf{x}, \mathbf{t}) = \sum_{\mathbf{y} \in \mathcal{Y}} \sum_{\mathbf{o} \in \mathcal{O}} p_\omega(\mathbf{z}, \mathbf{y}, \mathbf{o}|\mathbf{x}, \mathbf{t}) \tag{4.21}$$

where $\mathcal{D} = \{(\mathbf{x}_n, \mathbf{t}_n, \mathbf{z}_n)\}_{n=1}^{N}$ consists of the observed data for all sessions. We perform this optimization using standard gradient methods. Here, we consider the gradient equation for each of the three parameter groups: θ, π, and ϕ. The gradient equations for π and ϕ are shown below.

$$\nabla_\phi \log p_\omega(\mathbf{z}|\mathbf{x}, \mathbf{t}) = \sum_{m=1}^{M} \mathbb{E}_{p_\omega(i(m)|\mathbf{z}, \mathbf{x}, \mathbf{t})} \left[\nabla_\phi \log p_\phi(z_m|t_{i(m)})\right] \tag{4.22}$$

$$\nabla_\pi \log p_\omega(\mathbf{z}|\mathbf{x}, \mathbf{t}) = \sum_{i=1}^{L} \mathbb{E}_{p_\omega(o_i, \mathbf{y}|\mathbf{z}, \mathbf{x}, \mathbf{t})} \left[\nabla_\pi \log p_\pi(o_i|\mathbf{y})\right] \tag{4.23}$$

Both gradient equations take the form of a posterior expectation of the log gradient of the relevant distribution. The gradient with respect to the base classifier parameters

1: **for** $k = 1, ..., L$ **do**
2: **for** $c \in \mathcal{C}$ **do**
3: **for** $m = 0, ..., M$ **do**
4: **for** $c' \in \mathcal{C}$ **do**
5: $\beta(j, k, c, c', m) \leftarrow \sum_o \alpha(k-1, c', m-o)\, p_\pi(o|(c, k, k), c', k)\, p_\phi(z_m|t_k)^o$
6: **for** $j = 1, ..., k-1$ **do**
7: $\beta(j, k, c, c', m) \leftarrow \sum_o \beta(j, k-1, c', m-o)\, p_\pi(o|(c, j, k), c', k)\, p_\phi(z_m|t_k)^o$
8: $\alpha(k, c, m) \leftarrow \sum_j \sum_{c'} \exp(\langle \theta, \mathbf{f}((c, j, k), c', \mathbf{x})\rangle)\beta(j, k, c', m)$
9: Return α

Figure 4.6: The complete dynamic program for calculating the marginal likelihood of the observation sequence $p_\omega(\mathbf{z}|\mathbf{x}, \mathbf{t})$ in the proposed framework.

also takes the form of an expected gradient of a log density and is shown below.

$$\begin{aligned} \nabla_\theta \log\, p_\omega(\mathbf{z}|\mathbf{x}, \mathbf{t}) &= \mathbb{E}_{p_\omega(\mathbf{y}|\mathbf{z},\mathbf{x},\mathbf{t})}\left[\nabla_\theta \log p_\theta(\mathbf{y}|\mathbf{x})\right] \\ &= \mathbb{E}_{p_\omega(\mathbf{y}|\mathbf{z},\mathbf{x},\mathbf{t})}\left[\nabla_\theta \langle \theta, \mathbf{f}(\mathbf{x}, \mathbf{t}, \mathbf{y})\rangle\right] - \nabla_\theta Z_\theta(\mathbf{x}) \\ &= \mathbb{E}_{p_\omega(\mathbf{y}|\mathbf{z},\mathbf{x},\mathbf{t})}\left[\mathbf{f}(\mathbf{x}, \mathbf{t}, \mathbf{y})\right] - \mathbb{E}_{p_\theta(\mathbf{y}|\mathbf{x})}\left[\mathbf{f}(\mathbf{x}, \mathbf{t}, \mathbf{y})\right] \end{aligned} \tag{4.24}$$

where $\mathbf{f}(\mathbf{x}, \mathbf{t}, \mathbf{y})$ denotes the complete feature function for the semi-Markov CRF model. In this case, the log-linear form of the semi-Markov CRF model gives us the further interpretation that the learning algorithm is trying to match the expected feature function under the base semi-Markov CRF model to the posterior expected feature function given by the observation model. This is in contrast to typical maximum likelihood estimation for a log-linear model, which would match the expected feature function under the model to the expected feature function under the empirical distribution. We perform optimization of this objective using L-BFGS and the same warm-start procedure described in Section 4.2.2.

4.3.3 Inference

The primary computational challenge of this learning procedure is calculating the log marginal likelihood. This can be done exactly using a dynamic program for calculating $p_\omega(\mathbf{z}|\mathbf{x}, \mathbf{t})$. The complete dynamic program is shown in Figure 4.6. The primary dynamic programming table is α which has the following interpretation: $\alpha(k, c, m)$ is the unnormalized probability that the input subsequence $\mathbf{x}_{1:k}$ generated the observation subsequence $\mathbf{z}_{1:m}$ given that the last segment in $\mathbf{y}$ has label c where here $\mathbf{y}$ is a segmentation of the input *subsequence* $\mathbf{x}_{1:k}$. Or, written mathematically:

$$\alpha(k, c, m) \propto p_\omega(\mathbf{z}_{1:m}|\mathbf{x}_{1:k}, \mathbf{t}_{1:k}, c_{|\mathbf{y}|} = c) \tag{4.25}$$

$$= \sum_{\substack{\mathbf{y} \in \mathcal{Y}(\mathbf{x}_{1:k}): \\ c_{|\mathbf{y}|} = c}} \sum_{\mathbf{o}_{1:k}} p_\omega(\mathbf{z}, \mathbf{y}, \mathbf{o}|\mathbf{x}, \mathbf{t}) \tag{4.26}$$

Once this algorithm is complete we can calculate the unnormalized marginal likelihood for the complete model as

$$p_\omega(\mathbf{z}|\mathbf{x}, \mathbf{t}) \propto \sum_c \alpha(L, c, M). \tag{4.27}$$

Then, all that remains is to normalize the unnormalized marginal likelihood. Since the observation model is locally normalized, we need only calculate the normalizer for the base semi-Markov CRF model $Z_\theta(\mathbf{x}, \mathbf{t})$ which can be done using a dynamic program with complexity $\mathcal{O}(|\mathcal{C}|^2 L^2)$ [57]. In this algorithm, line 5 has complexity $\mathcal{O}(1)$ and is executed $\mathcal{O}(|\mathcal{C}|^2 LM)$ times, line 7 has complexity $\mathcal{O}(1)$ and is executed $\mathcal{O}(|\mathcal{C}|^2 L^2 M)$ times, and line 8 has complexity $\mathcal{O}(|\mathcal{C}|L)$ and is executed $\mathcal{O}(|\mathcal{C}|LM)$ times. Thus, the whole algorithm has complexity $\mathcal{O}(|\mathcal{C}|^2 L^2 M)$ where L is the length of the input sequence, $\mathcal{C}$ is the set of possible segment labels, and M is the length of the observation sequence.

We use reverse-mode automatic differentiation [4] to derive a dynamic program with the same complexity to calculate the necessary gradients for learning. We do not

use automatic differentiation software as most such packages cannot efficiently handle highly dynamic computation graphs such as the one shown in Figure 4.6. Instead, we manually derive the adjoints for each entry in the dynamic programming table and use these to compute the parameter gradients. As with all computation graphs, the backwards pass has the same complexity as the forwards pass [4], so calculating the gradients has the same complexity as calculating the marginal likelihood, namely $\mathcal{O}(|\mathcal{C}|^2 L^2 M)$.

4.3.4 Multiple Observation Types

In some settings, it may be desirable to allow for multiple types of observations. For example, we may want to include observations of both the beginning and end of sleep. This can be handled by including multiple observation sequences $\mathbf{z}^{(l)}$ each with length $M^{(l)}$ and observation indicator sequences $\mathbf{o}^{(l)}$ where l indicates the observation type. Observation sequences of each type are assumed to be independent conditioned on the segmentation $\mathbf{y}$ and the ordering assumption need not hold between types. The complexity of inference in this setup is $\mathcal{O}(|\mathcal{C}|^2 L^2 \prod_l M^{(l)})$.

4.3.5 Experiments

We evaluated the proposed framework's ability to accommodate the temporal imprecision in the label structure that arises in both the lab and field settings on two mHealth detection problems: sleep detection and smoking detection. In this section we describe the specific models used and the results of these evaluations.

4.3.5.1 Sleep detection

We evaluated our framework's performance on data from the field using the Extrasensory dataset [74]. For full details on this dataset, see Section 2.3. This dataset contains signals from a variety of sensors including the accelerometer, gyroscope, GPS, and microphone on a mobile device as well as a wrist-worn accelerometer. Subjects

carried these sensors during daily activities and self-reported a range of activities such as sleeping, eating, and exercising. We focus on the sleep detection problem, as this was one of the more abundantly reported activities. We note that there is no ground truth for this data, so we evaluated against the cleaned, self-reported sleep annotations provided in the data. To simulate extra imprecision in the observation process, we added further synthetic noise (described below) to the observation timestamps.

4.3.5.1.1 Model Our goal in the sleep detection problem is to segment the input sequence into periods of sleep and non-sleep. We used a binary semi-Markov CRF with a constraint that consecutive segments may not have the same label. We included as features the sum of all instance-level features within a segment $x_{jk} = \sum_{i=j}^{k} x_i$ as well as two duration-based features: $\mathbb{I}[c_s = 1](t_{k_s} - t_{j_s})$ and $\mathbb{I}[c_s = 1](t_{k_s} - t_{j_s})^2$. The duration-based features are equivalent to placing a normal distribution on the duration of sleeping activities[4]. We placed a zero-mean gaussian prior with tuned variance on the parameters of the semi-Markov CRF model (i.e. ℓ_2 regularization).

In our observation model, we included two types of observations: the beginning of sleep $\mathbf{z}^{(1)}$ and the end of sleep $\mathbf{z}^{(2)}$. Because sleep was observed in all sessions, we used a fixed, deterministic observation indicator distribution. That is, if instance i is the beginning of a sleep segment, it must generate an observation $z_m^{(1)}$ and likewise for the end of a sleep segment. No other instances may generate observations in this model.

To model the procedure of self-reporting when you go to sleep and when you wake up, we used a one-sided distribution to model the observation timestamp noise. We used the following exponential distributions to model observation timestamp noise:

[4] $\mathbb{I}[\cdot]$ is the indicator function

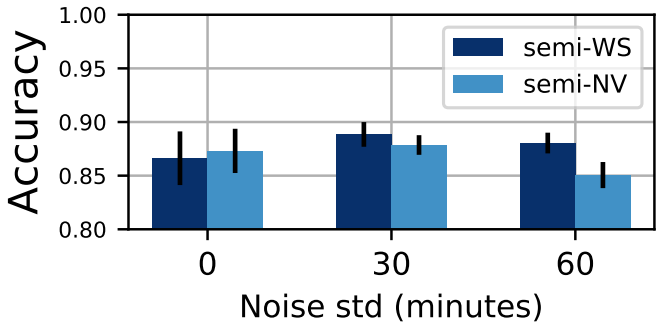


Figure 4.7: Performance for the semi-WS and semi-NV models on the sleep detection problem when trained on data with Exp(λ) distributed noise (measured in minutes) added to the observation timestamps.

$$p_\phi(z_m^{(1)}|t_{i(m)}) = \text{Exp}(t_{i(m)} - z_m^{(1)}; \lambda)$$
$$p_\phi(z_m^{(2)}|t_{i(m)}) = \text{Exp}(z_m^{(2)} - t_{i(m)}; \lambda)$$

We placed an inverse-Gamma prior with shape $\alpha = 1$ and scale $\beta = 1$ on λ. We found parameter estimation to be fairly insensitive to changes in the settings of this prior distribution and so we used weakly-informative default values for α and β.

4.3.5.1.2 Train and Test Protocols We evaluated performance using a 10-fold cross-validation procedure, where folds were formed at the session level. The strength of the ℓ_2 regularizer was tuned to maximize instance-level F_1 over a logarithmic grid using a further 9-fold cross-validation on the training set. This procedure is equivalent to assuming that some of the data has been labeled for tuning purposes. Predictions were evaluated against the self-reported labels.

4.3.5.1.3 Experiments We compared semi-Markov CRF models trained in two ways. First, we trained a semi-Markov CRF model based on a naive alignment defined by mapping each augmented observation to the nearest instance (semi-NV). Second, we trained a semi-Markov CRF model using the proposed weak supervision framework applied to the augmented observations (semi-WS). To test these models under a

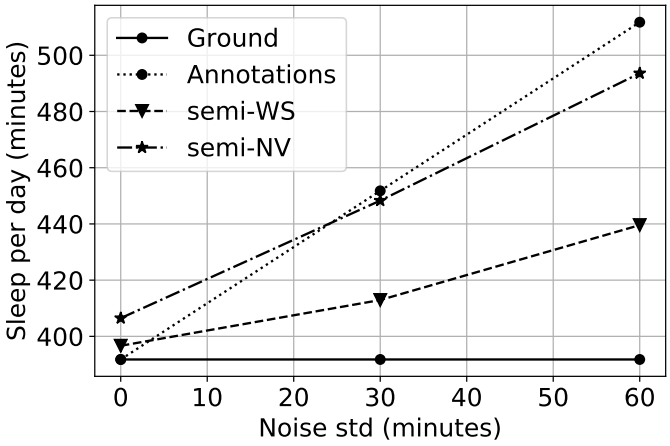


Figure 4.8: This plot shows the average sleep per day predicted by both the semi-WS and semi-NV models. Also shown is the average sleep per day in the true labels (Ground) and the expected sleep per day in the noisy annotations (Annotations).

variety of noise conditions, we added different amounts of independent, exponentially distributed noise to the observation timestamps and trained both models using these augmented observations. The results from these experiments are shown in figure 4.7. The plot shows how both models performed when trained and tested on observations augmented with standard deviation $\lambda = 0, 30, 60$ minutes of temporal noise. While small, the performance gap grows as the standard deviation of the observation noise increases, indicating that semi-WS is better able to learn from temporally imprecise labels.

This performance gap can be at least partially explained by looking at Figure 4.8. This plot shows the average predicted sleep per day for the semi-WS and semi-NV models. Also plotted is average sleep per day in the raw labels, approximately 392 minutes, and the average sleep per day with noise added. A model unaffected by the added noise should predict around 392 minutes of sleep per day whereas a model heavily effected by the added noise should predict an extra λ minutes of sleep per day. While both models fall between these extremes, the semi-WS is much closer to predicting the correct amount of sleep per day. This result supports the hypothesis

that the semi-NV model is learning to identify the periods around true sleep and incorrectly predicts that they are also sleep.

4.3.5.2 Smoking detection

We evaluated the proposed framework's ability to handle the types of imprecision that arise in a laboratory setting using the puffMarker smoking dataset [56]. For complete details on this dataset, see Section 2.3. Subjects were fitted with chest-worn respiration monitors and wrist-worn actigraphy sensors and asked to smoke a cigarette while an observer marked the occurrence of smoking puffs using a mobile phone app. The respiration signal was discretized into a sequence of non-overlapping respiration cycles (a single inhalation and exhalation) and the goal is to label each respiration cycle as a smoking puff or not and segment the respiration cycles into periods of smoking and non-smoking activities. The researchers visualized the respiration signal and manually aligned the observation timestamps to the visualized signal. We treat these manually aligned labels as ground truth for the purposes of evaluation, though we acknowledge that there may be errors in the alignment process. All experiments in this section used the real observation timestamps recorded during data collection for weakly-supervised learning. We all respiration based-features, augmented using the method described below.

4.3.5.2.1 Features We used all respiration based-features, augmented using the method described below. Further, we extracted features from the actigraphy data using the following procedure: Let t_i be the timestamp of the maximum peak in respiration cycle i. Extract a window beginning 8 seconds before t_i and ending 1 second after t_i and calculate as features the mean, max, min, standard deviation, median, and five bin histogram of each channel's signal within this window. The actigraphy channels included were accelerometer x, y, and z, accelerometer magnitude, gyroscope x, y, and z, gyroscope magnitude, and pitch and roll angles for a total of

100 actigraphy based features. Pitch and roll calculations using accelerometer data are only valid when the hand is stationary, so these signals were filtered using the procedure described in [56].

Respiration and actigraphy based features have very different properties as a function of time. Due in large part to the method we used to extract actigraphy based features (described in Section 2.3), these features tend to be very smooth through time, particularly as compared to the respiration features which are extracted from non-overlapping windows. One effect of this differential in smoothness is that the smooth noise model we propose in this section tends to over-emphasize temporally smooth features at the expense of less smooth features when the two feature sets are simply concatenated together into one long feature vector. To combat this effect, we use the actigraphy features to augment the respiration features in a manner similar to the filtering approach used in Saleheen et al. [56].

In particular, we used predictions $\hat{y}_{act}$ from a logistic regression model trained using only actigraphy features on a subset of instances with hand aligned labels to augment the respiration features. The form of the resulting augmented feature vectors is $\mathbf{x}_{aug} = [\hat{y}_{act}\mathbf{x}_{resp} \quad (1 - \hat{y}_{act})\mathbf{x}_{resp}]$ where $\mathbf{x}_{resp}$ is the vector containing only respiration features. This augmentation can roughly be thought of as a hierarchical model. A similar effect could be achieved by only including interaction effects between the actigraphy features and the respiration features; however, this would result in more than 10,000 features. The filtering approach can therefore also be thought of as first doing a supervised compression of the actigraphy features and then doing a polynomial basis expansion.

As stated, we assume that some number of hand-aligned labels are available for training the the feature augmentation model. Here we consider the effect of the amount of hand-aligned data on the end-to-end prediction performance of a model learned using augmented features $\mathbf{x}_{aug}$. The experimental protocol varies the number

of sessions of fully labeled instances used to train the feature augmentation model. For each number of sessions, the feature augmentation model is trained, and used to produce the augmented feature vectors $\mathbf{x}_{aug}$. For the purpose of this evaluation, a second-stage logistic regression model is then trained using the augmented features $\mathbf{x}_{aug}$.

Three second-stage models are considered: (1) logistic regression trained using hand-aligned labels (LR-HA), (2) logistic regression trained using a naive alignment strategy where positive instance observations are mapped to the nearest instance (LR-NV), and (3) the weakly supervised logistic regression model presented in 4.2 trained using the unaligned observation timestamps (LR-WS). In all cases, the results shown are for a leave-one-session-out experimental protocol using hand-aligned labels for testing. The results were averaged over three random seeds to account for the random sampling of the sessions used to train the feature augmentation model.

The end-to-end performance of these models is shown in Figure 4.9. We found that the relative performance of these models remains relatively stable as the subset size changes. In particular, there is a difference of 0.03 in the F1 score when doubling the number of sessions used to train the feature augmentation model from 10 to 20. For all experiments in the remainder of this section, we used augmented features $\mathbf{x}_{aug}$ derived from a fully-labeled subset of the data consisting of 10 sessions as our instance features for all models.

4.3.5.2.2 Model Our goal in the smoking detection problem is to label each respiration cycle as smoking or non-smoking and to segment the input sequence into periods of smoking and non-smoking; however, smoking detection differs from typical segmentation problems in that a complete smoking activity contains a mix of smoking puffs and non-smoking respiration cycles. Accordingly, we use the heterogeneous segmentation (SEG) model described in Section 3.4 as our base model $p_\theta(\mathbf{y}|\mathbf{x})$. We

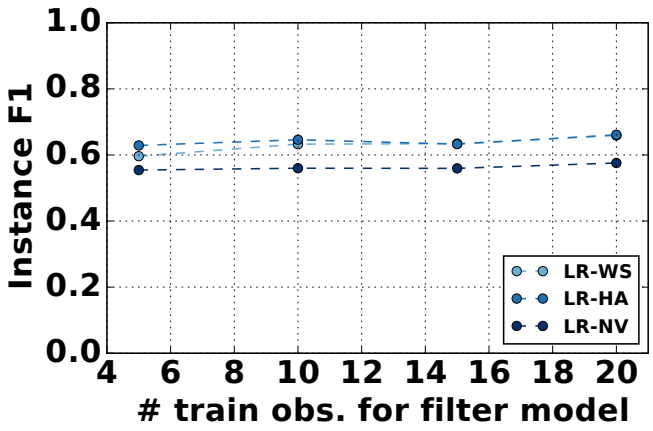


Figure 4.9: The instance labeling performance of logistic regression based models as a function of the number of fully-labeled sessions used to train the feature augmentation model.

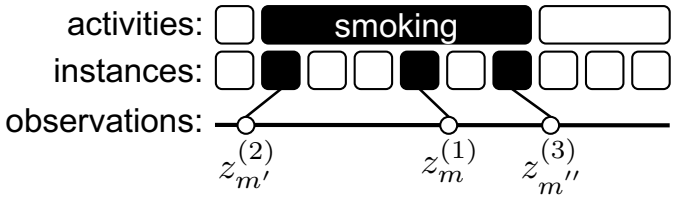


Figure 4.10: An illustration of the observation types used in the SEG-WS model. $\mathbf{z}^{(2)}$ and $\mathbf{z}^{(3)}$ contain activity start and end observations respectively while $\mathbf{z}^{(1)}$ contains observations of smoking puffs within a smoking activity.

placed a zero-mean gaussian prior on the parameters of the SEG model (i.e. ℓ_2 regularization).

As seen in Figure 4.10, we included three types of observations. $\mathbf{z}^{(1)}$ contains observations associated with smoking puffs that are neither the first nor last in a smoking activity. That is, $\mathbf{z}_m^{(1)}$ marks the start of a segment with label $c > 1$. $\mathbf{z}^{(2)}$ contains observations associated with the start of a smoking activity, or, in other words, the start of a segment with label $c = 1$. Finally, $\mathbf{z}^{(3)}$ contains observations associated with the last smoking puff in a smoking activity, or, in other words, with the start of a segment with label $c = 0$. We used the following Bernoulli distributions for our observation indicator model $p_\pi(o_i^{(l)}|\mathbf{y})$:

$$p_\pi(o_i^{(1)} = 1|\ i \text{ is the start of an inter-event span }) = \pi_1^{(1)}$$
$$p_\pi(o_i^{(2)} = 1|\ i \text{ is the start of a smoking activity}) = \pi_1^{(2)}$$
$$p_\pi(o_i^{(3)} = 1|\ i \text{ is the end of a smoking activity}) = \pi_1^{(2)}$$

where $\pi_1^{(1)}, \pi_1^{(2)} \in [0, 1]$. The distributions over $o_i^{(2)}$ and $o_i^{(3)}$ share a parameter $\pi^{(2)}$, which reflects the assumption that it is equally likely to miss an activity start observation as an activity end observation. For the observation timestamp density, we used the following normal distribution:

$$p_\phi(z_m^{(l)}|t_{i(m)}) = \mathcal{N}(z_m^{(l)}; t_{i(m)} + \mu_l, \sigma_l^2)$$

for $l \in \{1, 2, 3\}$ where $\phi = \{\mu, \sigma\}$. We placed a Uniform$(0, 1)$ prior on each $\pi^{(l)}$, a standard normal prior on each μ_l, and an inverse-Gamma prior with shape $\alpha = 1$ and scale $\beta = 1$ on each σ_l^2. As in the sleep detection model, we found parameter estimation to be insensitive to changes in the settings of these prior distributions, so we chose default weakly-informative values. The only hyperparameter for this model is the ℓ_2 regularization strength.

4.3.5.2.3 Train and Test Protocols We evaluated performance using a leave-one-session-out cross-validation procedure. We tuned all ℓ_2 regularization strength hyperparameters to maximize instance level F_1 over a logarithmic grid using a further nested leave-one-session-out evaluation on the training set. We evaluated predictions against the hand-aligned labels.

4.3.5.2.4 Experiment 1 - Inference Pruning While the inference algorithm described in Section 4.3.3 is at most quadratic in the size of each input, the overall run time can be quite high, particularly for long sequences or models with a large segment label set $\mathcal{C}$, such as the SEG model for smoking detection. In order to improve infer-

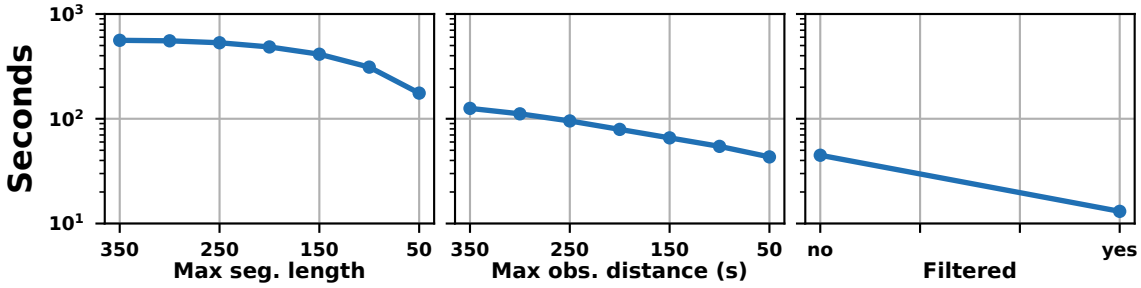


Figure 4.11: This figure shows the effect of changing the maximum segment length with no observation depth pruning or filtering (left), the effect of changing the maximum observation distance with no filtering (center), and the further marginal effect of filtering approximately 85% of instances (right). The maximum pruning configuration results in a 40x speedup.

ence run times, we consider three strategies to prune the inference dynamic program. First, we bound the maximum segment length, as described in section 3.6.1.

Second, we place a constraint on the maximum time between a true event and an associated timestamp. This corresponds to using a truncated distribution for $p_\phi(z_m|t_i)$. The effect this has on the complexity of inference is more complex than the effect of bounding the maximum inter-event segment length as the effect depends on the timestamps of the input and observation sequences; however, given a maximum observation distance of r, we can upper bound the inference complexity by $\mathcal{O}(|\mathcal{C}|^2 LB\tilde{M})$ where $\tilde{M}$ is the maximum number of observations that could be associated with a single instance or $\tilde{M} = \max_i \sum_m \mathbb{I}\left[t_i - r \leq z_m \leq t_i + r\right]$. In practice, the average improvement in runtime is better than this because many instances are so far from an observed timestamp that they could not have generated any observations.

Finally, we use a version of Bodenstab filtering (see Section 3.6.2). As our filter function, we use the same logistic regression that we used to augment the respiration features. Because this filter function is trained on ground truth labels, we can train it prior to training the detection model and use it to improve inference speeds. Because, this augmentation model is trained on temporally smooth features, it naturally tends to give high-recall predictions, making it well suited for use as a filter model.

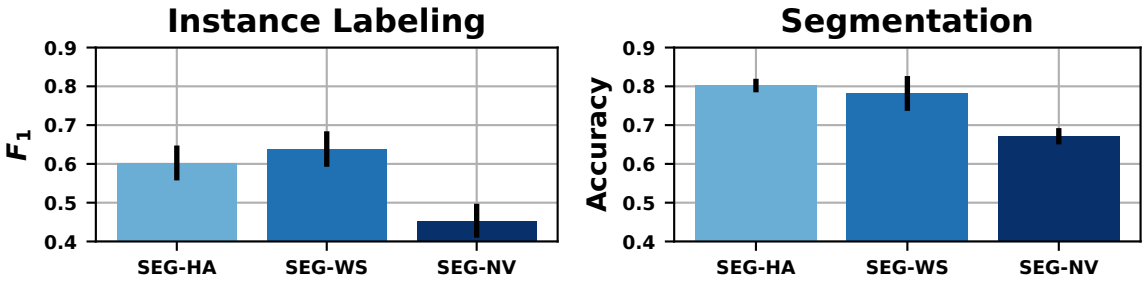


Figure 4.12: The left plot shows F_1 score for all three models on the instance labeling task. The right plot shows the accuracy for all three models on the segmentation task. Error bars show one standard error.

To test the effect of these pruning strategies, we ran an ablation experiment to assess the time required to run marginal inference in the SEG-WS model using different combinations of pruning techniques. First, we varied the maximum segment length from 350 to 50. Next, with the maximum segment length fixed at 50, we varied the maximum observation distance from 350 to 50. Finally with the maximum segment length and maximum observation distance fixed at 50, we ran inference with and without negative instance filtering. Figure 4.11 shows the run time in seconds for each of these settings[5]. Using all pruning strategies, the runtime of marginal inference is decreased from approximately 600 seconds to approximately 15 seconds, a 40 times speedup. We use the maximum pruning settings for all remaining experiments in this chapter.

4.3.5.2.5 Experiment 2 - Detection Performance We next evaluated the ability of the proposed framework to learn the parameters of the base classifier from imprecise lab data by comparing the SEG model trained in three different ways. First, we trained the SEG model directly on the hand-aligned labels (SEG-HA). This represents the gold standard performance that we would like to achieve. Second, we

[5]Runtime experiments were performed on a 2.8 GHz Intel Core i7 processor with 8GB of RAM and the inference algorithm was coded in Cython.

trained the SEG model on labels generated by associating each observation timestamp with the closest respiration cycle (SEG-NV). This represents the naive baseline and we would expect our procedure to fall somewhere between SEG-HA and SEG-NV. Third, we trained the SEG model using the weak supervision framework proposed above (SEG-WS). Figure 4.12 shows the performance of all three models on the instance labeling and segmentation tasks. The SEG-WS model performs approximately as well as the SEG-HA model at both the instance labeling and segmentation tasks while the SEG-NV model performs worse than either. A paired t-test indicates that the improvement in SEG-WS results over SEG-NV results is statistically significant in terms of both instance labeling and segmentation ($p \leq 0.05$). These results indicate that the proposed weak supervision framework is able to effectively learn from imprecise observations that occur in the lab setting.

One interesting characteristic of the SEG-WS model is that it tends to predict one or two contiguous segments, whereas the SEG-HA and SEG-NV models tend to predict a more fragmented segmentation. One possible explanation for this behavior is that while imprecision in the instance-level annotations means that the posterior expectations of the instance-level feature functions will be a mixture of the features from multiple nearby instances, the posterior expectations of the segment-level feature functions should be nearly the same as the ground truth values for these feature functions. For example, the exact location of the positive instances does not effect the number of such instances. Therefore, if the observation indicator distribution is peaked and there are not many missing or spurious observations, then the posterior expectation of the number of positives in an activity should be very close to the true value. This results in a stronger learning signal for the segment-level parameters than for instance-level parameters. This suggests that including segmentation-level features in the model may improve performance in settings with imprecise observations *beyond* the improvements we get in settings with ground truth observations.

4.4 Combining Imprecise Annotations and Wearable Sensors

Mobile sensors and self-report are not mutually exclusive study techniques. A combination of the two techniques has the potential to outperform either technique alone. While some studies have combined the two, to the best of our knowledge, current methods for synthesizing these two types of observations are ad hoc and domain specific (e.g. [48]). In this section, we address this problem by performing posterior inference in the weak supervision framework presented in the previous section, obviating the need for ad hoc solutions. We show that explicitly modeling the observation process leads to improved performance over treating test-time observations as ground truth.

4.4.1 MAP Inference

Our goal in this section is to combine continuous time observations, such as self-reported activities, with wearable sensor input to infer behaviors. That is, we would like to infer the most likely label structure $\mathbf{y}$ given $\mathbf{x}$, $\mathbf{t}$, and $\mathbf{z}$. To do this, we perform full maximum a posteriori (MAP) inference over both $\mathbf{y}$ and $\mathbf{o}$

$$\mathbf{y}^*, \mathbf{o}^* = \arg\max_{\mathbf{y},\mathbf{o}} p_\omega(\mathbf{y}, \mathbf{o} | \mathbf{z}, \mathbf{x}, \mathbf{t}) \tag{4.28}$$

$$= \arg\max_{\mathbf{y},\mathbf{o}} p_\omega(\mathbf{z}, \mathbf{y}, \mathbf{o} | \mathbf{x}, \mathbf{t}) \tag{4.29}$$

The dynamic program presented in Section 4.3.3 to calculate the marginal likelihood can be used to perform MAP inference by swapping summation over $\mathbf{y}$ and $\mathbf{o}$ for maximization and using backtracking to recover $\mathbf{y}^*$ and $\mathbf{o}^*$ with no change in the computational complexity. This modified dynamic program is shown in Figure 4.13. The main dynamic programming table α has the interpretation that an entry in this table $\alpha(k, c, m)$ is the unnormalized posterior probability of the MAP segmentation and observation indicator sequence given the input and timestamp subsequences $\mathbf{x}_{1:k}$ and $\mathbf{t}_{1:k}$ and the observation timestamp subsequence $\mathbf{z}_{1:m}$ given that the final

for $k = 1, ..., L$ **do**
  for $c \in \mathcal{C}$ **do**
    for $m = 0, ..., M$ **do**
      for $c' \in \mathcal{C}$ **do**
        $\beta(j, k, c, c', m) \leftarrow \max_o \alpha(k-1, c', m-o)\, p_\pi(o|(c,k,k), c', k)\, p_\phi(z_m|t_k)^o$
        for $j = 1, ..., k-1$ **do**
          $\beta(j, k, c, c', m) \leftarrow \max_o \beta(j, k-1, c', m-o) p_\pi(o|(c,j,k), c', k) p_\phi(z_m|t_k)^o$
      $\alpha(k, c, m) \leftarrow \max_j \max_{c'} \exp(\langle \theta, \mathbf{f}((c,j,k), c', \mathbf{x}) \rangle) \beta(j, k, c', m)$
Return α

Figure 4.13: The dynamic program for calculating the unnormalized probability of MAP assignment to **o** and **y** in the proposed framework.

segment in the MAP segmentation of this subsequence has the label c. Given this interpretation, the unnormalized posterior probability $\mathbf{y}^*$ and $\mathbf{o}^*$ can be calculated as $p(\mathbf{y}^*, \mathbf{o}^*|\mathbf{z}, \mathbf{x}, \mathbf{t}) \propto \max_c \alpha(L, c, M)$

4.4.2 Experiments

We evaluated this method on the sleep detection and smoking detection problems using the extrasensory and puffMarker datasets. We used the features, models, and train/test protocols presented in Sections 4.3.5.1 and 4.3.5.2. The key difference between the experiments in this section and those in Section 4.3.5 is that for each test session n, all methods are given access to the observations for that session, $\mathbf{z}_n$. In this context, the naive alignment strategy maps segment transition observations to the nearest instance and treats them as ground truth, finding the MAP segmentation that agrees with the naively aligned observations. We tested all models when given either all activity start observations (Start), all activity end observations (End), neither (None), or both (Start+End) at test time. This simulates different plausible self-report scenarios where subjects only give partial information.

4.4.2.0.1 Experiment 1 - Sleep Detection As in 4.3.5.1, we trained semi-NV and semi-WS using observations with different amounts of added exponential noise.

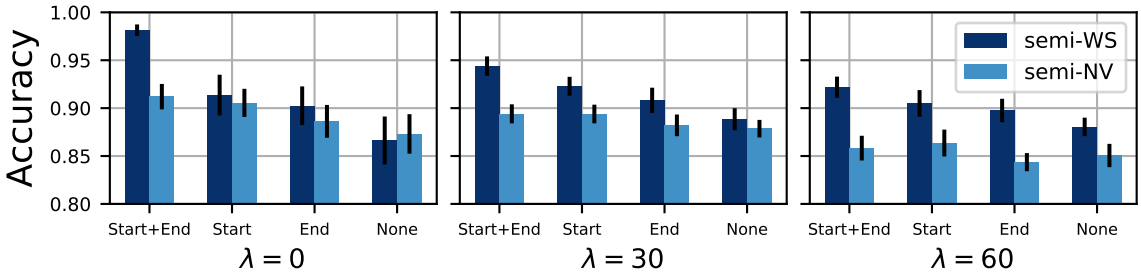


Figure 4.14: Performance for the semi-WS and semi-NV models on the sleep detection problem when trained on data with Exp(λ) distributed noise (measured in minutes) added to the observation timestamps. Each plot shows the performance of both models when conditioned on all segment start observations (Start), all segment end observations (End), neither (None), or both (Start+End) at test time.

In this experiment, observations for the test sessions were modified with the same amount of noise. The results from these experiments are shown in figure 4.14. The plot shows how both models performed when trained and tested on observations augmented with standard deviation $\lambda = 0, 30, 60$ minutes of temporal noise. Within each plot, the performance for each model when conditioned on different amounts of information is shown. In all but one case, semi-WS outperforms semi-NV. The performance gap grows as the standard deviation of the observation noise increases and as the amount of information conditioned on grows indicating that using an explicit observation model is useful when incorporating imprecise observations. In many cases semi-NV model is able to explain the test-time observations by inserting a short positive or negative segment and leaving the rest of the predicted segmentation unchanged. The semi-WS model, on the other hand, tends to explain observations by shortening or lengthening segments in the unconditioned prediction resulting in more substantive incorporation of the observations and higher prediction accuracy.

4.4.2.0.2 Experiment 2 - Smoking Detection We evaluated the ability of the SEG-WS model to combine sensor data with timestamp observations at test time. In these experiments, we are conditioning *only* on the activity start and end observations

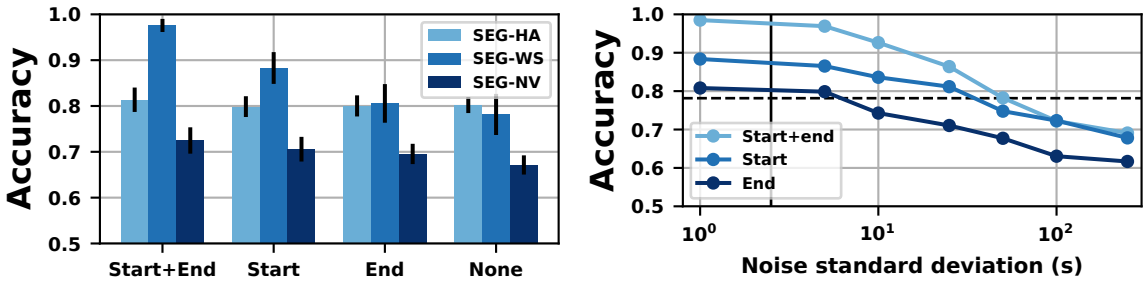


Figure 4.15: The left plot shows the segmentation accuracy when all three SEG models are conditioned on combinations of observations (segment start, segment end or both). The right plot shows the performance of the SEG-WS model when conditioned on segment observations with different amounts of synthetic noise added to the observation sequence. The dashed line shows the segmentation accuracy of the SEG-WS model when conditioned on no observations (None) and the solid black line shows the empirical standard deviation of the timestamp noise in the data, which reflects what SEG-WS was trained on.

$\mathbf{z}^{(2)}$ and $\mathbf{z}^{(3)}$ at test time and not the internal smoking puff observations $\mathbf{z}^{(1)}$. The results are shown in Figure 4.15 (left). Unlike in the sleep detection experiments, all imprecision present in these observations was real and all evaluations were made against carefully hand aligned labels. While conditioning on segment observations results in improvements for all three models, these gains are much larger for the SEG-WS model. In particular, conditioning on both the segment start and end timestamps results in a 6% error reduction for the SEG-HA model and a 16% error reduction for the SEG-NV model. Conditioning on the same information results in an 89% error reduction for the SEG-WS model.

The reason for this gap in performance when conditioning on observations becomes clear when we recall from Section 4.3.5.2 that the SEG-WS model tends to predict long, contiguous smoking activities whereas the SEG-HA and SEG-NV tend to predict fragmented short fragmented activities. When conditioning on a particular split-point, the SEG-HA and SEG-NV models can simply adjust one of the many small segments to match this split point and leave the rest unchanged. The SEG-WS

model, on the other hand, must adjust the single predicted activity to explain the observations resulting in near perfect predictions.

In general, we cannot expect the noise we observe in the field to look like the noise we observe in the lab, therefore it is valuable to know how sensitive the SEG-WS model is to the correctness of the observation timestamp model. To test this, we trained the SEG-WS model on the real imprecise observations, but tested using synthetic observation timestamps drawn from a normal distribution centered at the true activity start or end. We varied the standard deviation of the synthetic noise distribution to see how performance degrades as the test time noise distribution grows further from the train time noise distribution. The results of this experiment are shown in 4.15 (right) where the x-axis is the standard deviation of the synthetic noise distribution. The results show that the SEG-WS model can successfully incorporate observations with up to an order of magnitude more noise than was observed at train time. As expected, adding sufficient noise to the observations eventually causes performance to degrade; however, even with large amounts of noise, posterior segmentation accuracy plateaus between 0.6 and 0.7 compared to an accuracy of approximately 0.8 when not conditioning on any observations.

4.5 Discussion

In this chapter, we presented a framework for learning independent and structured detection models from temporally imprecise annotations. To motivate this framework, we focused on the case where all training data comes from a single source, however one interesting use case for the proposed framework is to allow models to be trained on data from multiple sources. For example, suppose we had a limited set of data gathered in the lab with precise annotations and a larger set of data gathered in the field with self-reported annotations. The proposed framework allows us to learn a shared base classifier by instantiating a separate observation model for each data

source. It is similarly trivial to incorporate data for which the true label structure is observed. In this case, the learning objective looks like standard semi-supervised learning objectives where for some instances we are maximizing $p_\theta(\mathbf{y}|\mathbf{x}, \mathbf{t})$ and for others we are maximizing $p_\theta(\mathbf{z}|\mathbf{x}, \mathbf{t})$. Combining data in this way, we can use field data to ensure external validity while still leveraging the high-quality data gathered in the lab.

The proposed framework has a few of important limitations that suggest future research directions. The first limitation is the need to specify and validate an observation model. While specifying an observation model may seem no more difficult than specifying any other piece of a model, validating such a model requires knowing the ground true label structure for some subset of sessions. In the self-report setting, for example, this may be achieved by having some subjects wear a second, higher-quality sensor, but at very least, this complicates the study design. This is a general problem for weakly supervised learning frameworks that treat the true label variable as a latent variable.

A second limitation is multimodality of the marginal likelihood. Analysis suggests that the posterior distribution over the parameters in this model is bimodal. In one of the modes, the model treats all observations as false positives and in the other it treats most of the observations a true positives. We were able to avoid the undesirable mode by pre-training the model with a deterministic observation indicator distribution; however, two questions remain for future work. First, while analysis of simple versions of the model suggests that using a deterministic observation indicator distribution, a log-linear base classifier, and a normal observation timestamp distribution results in a unimodal likelihood, this remains to be proved in general. Second, it remains to be shown whether the likelihood with a non-deterministic observation indicator distribution has only two modes, as it appears.

A final limitation is that the complexity of inference scales exponentially in the number of *types* of observations (see section 4.3.4). One possible alternative to this approach is to associate each individual observation with a type such that $z_m = (t_m, l_m)$ where t_m is the observation timestamp and l_m is the observation type. $\mathbf{o_i}$ is now a categorical variable indicating not only whether an instance is associated with an observation, but also which type of observation. In the stochastic processes literature, this is referred to as a marked point process (e.g. [58]). The complexity of inference in this case is $\mathcal{O}(|\mathcal{C}|L^2 \sum_l M^{(l)})$ where $M^{(l)}$ is the number of observations of type l; however, observations of different types must now obey the ordering assumption (assumption 1). When applied to multiple observation types, this assumption implies, for example, that a subject will never report two different types of activities out of order. Relaxing this assumption in cases with multiple observation types remains a problem for future work.

CHAPTER 5

CONCLUSIONS

In this thesis, we addressed a number of practically motivated problems in mHealth activity detection using machine learning methods. First, we presented a class of conditional random field models for heterogeneous segmentation. We applied this model class to three different mHealth detection problems and showed across-the-board improvements in prediction performance compared to the types of models that are typically used in mHealth settings. Second, we explored two strategies for pruning the dynamic programs used for inference in segmentation models. We showed that static pruning strategies can be applied to achieve linear improvements in inference runtime. Further, we used a learned pruning strategy, originally developed for parsing, to achieve a two orders of magnitude improvement in inference complexity on a smoking detection problem. Finally, we introduced a new weakly supervised learning problem in which supervision for discrete-time detection models is provided in the form of imprecise continuous-time annotations. We proposed a weakly supervised learning framework to address this problem and applied it to independent classification and segmentation models, demonstrating improvements over automatic alignment strategies.

While specific limitations and directions for future work were discussed in Sections 3.7 and 4.5, there are a few broader research directions that deserve further discussion. One important set of techniques that we did not discuss in this thesis is neural networks. Our primary goal in Chapter 3 was to model activity-level features. We did this using CRF-based structured prediction, however, an alternative would

have been to use a recurrent neural network to learn long-range dependencies between instances. In other fields, such as natural language processing, we have seen a progression from structured prediction based on graphical models to neural networks (e.g. [9]). Neural networks have the flexibility to learn structure directly from data; however, learning that structure typically requires more data than fine tuning the parameters in a heavily constrained CRF model. In many mHealth detection problems, we must learn to identify activities with fewer than 20 examples, which is generally not enough to learn long-range dependencies.

An alternative application of neural networks to the activity detection problem is to use them to learn instance-level features. Such networks could be used in place of hand-derived features, which may fail to generalize across target activities, discretization methods, or sensing modalities. While we may only have 20 complete activities to use for model training, these activities are typically comprised of hundreds to thousands of individual instances, making the prospect of learning a flexible neural model much better. We view this application of neural networks as orthogonal to the work in this thesis and as a promising direction for future research. For example, one could integrate neural networks into the models described in Chapter 3 by replacing the instance-level feature functions with neural networks. Similarly, the weak supervision framework presented in Section 4.2 can be used to learn any classier with probabilistic outputs which includes both feed-forward and recurrent neural networks with sigmoid output layers. A particularly interesting direction along these lines is to use the raw sensor signal as input to these instance-level networks. This approach may obviate the need for domain specific features, but may also necessitate novel network architectures to efficiently work with high-resolution signals.

An important limitation of the work presented in this thesis is that we focus almost entirely on global models of behavior. The characteristics of behavior vary widely among people and capturing these differences in personalized models may

prove crucial to improving detection performance. Limited data quantities make evaluating such techniques difficult; however, data from large scale studies, such as the *All of Us*[1] study, and techniques to learn from data gathered in the field, such as the methods presented in Chapter 4, can help us to overcome this obstacle. There are a number of possible approaches to this problem including adaptation of methods from multi-task and online learning. With sufficient data, models may be even further personalized to include contextual and temporal information. For example, in the ECG morphology extraction problem, we modeled the duration between peaks of different types. These durations vary between subjects and within a single subject's data depending on what activities the subject is performing. One can imagine a model that uses an auxiliary sensor to recognize when a person is sitting and adjusts the ECG morphology model accordingly.

[1] `https://allofus.nih.gov/`

BIBLIOGRAPHY

[1] Amin Ahsan Ali, Syed Monowar Hossain, Karen Hovsepian, Md Mahbubur Rahman, Kurt Plarre, and Santosh Kumar. mPuff: automated detection of cigarette smoking puffs from respiration measurements. In *Proceedings of the 11th international conference on Information Processing in Sensor Networks*, pages 269–280. ACM, 2012.

[2] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. Support vector machines for multiple-instance learning. In *Advances in neural information processing systems*, pages 561–568, 2002.

[3] Rummana Bari, Roy J Adams, Md Mahbubur Rahman, Megan Battles Parsons, Eugene H Buder, and Santosh Kumar. rconverse: Moment by moment conversation detection using a mobile respiration sensor. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(1):2, 2018.

[4] Atilim Gunes Baydin, Barak A Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *arXiv preprint arXiv:1502.05767*, 2015.

[5] Nathan Bodenstab, Aaron Dunlop, Keith Hall, and Brian Roark. Beam-width prediction for efficient context-free parsing. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 440–449. Association for Computational Linguistics, 2011.

[6] Lora E Burke, Jing Wang, and Mary Ann Sevick. Self-monitoring in weight loss: a systematic review of the literature. *Journal of the American Dietetic Association*, 111(1):92–102, 2011.

[7] Marc-André Carbonneau, Veronika Cheplygina, Eric Granger, and Ghyslain Gagnon. Multiple instance learning: A survey of problem characteristics and applications. *Pattern Recognition*, 2017.

[8] Olivier Chapelle, Bernhard Schölkopf, Alexander Zien, et al. Semi-supervised learning. 2006.

[9] Danqi Chen and Christopher Manning. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750, 2014.

[10] Noam Chomsky. On certain formal properties of grammars. *Information and control*, 2(2):137–167, 1959.

[11] Ross DeVol, Armen Bedroussian, Anita Charuworn, Anusuya Chatterjee, In Kyu Kim, Soojung Kim, and Kevin Klowden. An unhealthy america: The economic burden of chronic disease. 2007.

[12] Justin Domke. Learning graphical model parameters with approximate marginal inference. *IEEE transactions on pattern analysis and machine intelligence*, 35 (10):2454–2467, 2013.

[13] Mark Dredze, Partha Pratim Talukdar, and Koby Crammer. Sequence learning from data with multiple labels. In *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases Workshop on Learning from Multi-Label Data*, page 39, 2009.

[14] Benoît Frénay and Michel Verleysen. Classification in the presence of label noise: a survey. *IEEE transactions on neural networks and learning systems*, 25(5):845–869, 2014.

[15] Aritra Ghosh, Naresh Manwani, and PS Sastry. Making risk minimization tolerant to label noise. *Neurocomputing*, 160:93–107, 2015.

[16] Aritra Ghosh, Himanshu Kumar, and PS Sastry. Robust loss functions under label noise for deep neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1919–1925, 2017.

[17] Alan Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE, 2013.

[18] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 369–376. ACM, 2006.

[19] Samuel S Gross, Olga Russakovsky, Chuong B Do, and Serafim Batzoglou. Training conditional random fields for maximum labelwise accuracy. In *Advances in Neural Information Processing Systems*, pages 529–536, 2007.

[20] Xinze Guan, Raviv Raich, and Weng-Keen Wong. Efficient multi-instance learning for activity recognition from time series data using an auto-regressive hidden markov model. In *Proceedings of the International Conference on Machine Learning*, pages 2330–2339, 2016.

[21] Jerónimo Hernández-González, Iñaki Inza, and Jose A Lozano. Learning bayesian network classifiers from label proportions. *Pattern Recognition*, 46(12):3425–3440, 2013.

[22] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[23] David W Hosmer Jr and Stanley Lemeshow. *Applied logistic regression*. John Wiley & Sons, 2004.

[24] Rong Jin and Zoubin Ghahramani. Learning with multiple labels. In *Advances in neural information processing systems*, pages 897–904, 2002.

[25] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[26] Hema Koppula and Ashutosh Saxena. Learning spatio-temporal structure from rgb-d videos for human activity detection and anticipation. In *Proceedings of the International Conference on Machine Learning*, pages 792–800, 2013.

[27] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[28] Santosh Kumar, Wendy Nilsen, Misha Pavel, and Mani Srivastava. Mobile health: Revolutionizing healthcare through transdisciplinary research. *Computer*, (1):28–35, 2013.

[29] Simon Lacoste-Julien, Martin Jaggi, Mark Schmidt, and Patrick Pletscher. Block-coordinate frank-wolfe optimization for structural svms. *arXiv preprint arXiv:1207.4747*, 2012.

[30] John Lafferty, Andrew McCallum, and Fernando C N Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. 2001.

[31] Karim Lari and Steve J Young. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 4(1): 35–56, 1990.

[32] Sungyoung Lee, Hung Xuan Le, Hung Quoc Ngo, Hyoung Il Kim, Manhyung Han, Young-Koo Lee, et al. Semi-markov conditional random fields for accelerometer-based activity recognition. *Applied Intelligence*, 35(2):226–241, 2011.

[33] Wee Sun Lee and Bing Liu. Learning with positive and unlabeled examples using weighted logistic regression. In *Proceedings of the International Conference on Machine Learning*, volume 3, pages 448–455, 2003.

[34] Lin Liao, Dieter Fox, and Henry Kautz. Extracting places and activities from gps traces using hierarchical conditional random fields. *The International Journal of Robotics Research*, 26(1):119–134, 2007.

[35] Gideon S Mann and Andrew McCallum. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *Journal of machine learning research*, 11(Feb):955–984, 2010.

[36] Oded Maron and Tomás Lozano-Pérez. A framework for multiple-instance learning. *Advances in neural information processing systems*, pages 570–576, 1998.

[37] J Michael McGinnis, Pamela Williams-Russo, and James R Knickman. The case for more active policy attention to health promotion. *Health affairs*, 21(2):78–93, 2002.

[38] Christopher Merck, Christina Maher, Mark Mirtchouk, Min Zheng, Yuxiao Huang, and Samantha Kleinberg. Multimodality sensing for eating recognition. In *Proceedings of the 10th EAI International Conference on Pervasive Computing Technologies for Healthcare*, pages 130–137, 2016.

[39] Volodymyr Mnih and Geoffrey E Hinton. Learning to label aerial images from noisy data. In *Proceedings of the International Conference on Machine Learning*, pages 567–574, 2012.

[40] Ali H Mokdad, James S Marks, Donna F Stroup, and Julie L Gerberding. Actual causes of death in the united states, 2000. *Jama*, 291(10):1238–1245, 2004.

[41] Andreas C. Müller and Sven Behnke. pystruct - learning structured prediction in python. *Journal of Machine Learning Research*, 15:2055–2060, 2014.

[42] Inbal Nahum-Shani, Shawna N Smith, Bonnie J Spring, Linda M Collins, Katie Witkiewitz, Ambuj Tewari, and Susan A Murphy. Just-in-time adaptive interventions (jitais) in mobile health: Key components and design principles for ongoing health behavior support. *Annals of Behavioral Medicine*, pages 1–17, 2016.

[43] Annamalai Natarajan, Edward Gaiser, Gustavo Angarita, Robert Malison, Deepak Ganesan, and Benjamin Marlin. Conditional random fields for morphological analysis of wireless ecg signals. In *Proceedings of the 5th ACM Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 370–379. ACM, 2014.

[44] Radford M Neal et al. Mcmc using hamiltonian dynamics. *Handbook of Markov Chain Monte Carlo*, 2(11), 2011.

[45] David F Nettleton, Albert Orriols-Puig, and Albert Fornells. A study of the effect of different types of noise on the precision of supervised learning techniques. *Artificial intelligence review*, 33(4):275–306, 2010.

[46] Daniel Olguín and Alex Pentland. Assessing group performance from collective behavior. In *Proceedings of the ACM conference on Computer-Supported Cooperative Work and Social Computing*, 2010.

[47] Abhinav Parate, Meng-Chieh Chiu, Chaniel Chadowitz, Deepak Ganesan, and Evangelos Kalogerakis. RisQ: recognizing smoking gestures with inertial sensors on a wristband. In *Proceedings of the 12th annual international conference on Mobile systems, applications, and services*, pages 149–161. ACM, 2014.

[48] Sanjay R Patel, Jia Weng, Michael Rueschman, Katherine A Dudley, Jose S Loredo, Yasmin Mossavar-Rahmani, Maricelle Ramirez, Alberto R Ramos, Kathryn Reid, Ashley N Seiger, et al. Reproducibility of a standardized actigraphy scoring algorithm for sleep in a us hispanic/latino population. *Sleep*, 38(9): 1497–1503, 2015.

[49] Deepak Pathak, Philipp Krahenbuhl, and Trevor Darrell. Constrained convolutional neural networks for weakly supervised segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1796–1804, 2015.

[50] Christine A Pellegrini, Jeremy Steglitz, Winter Johnston, Jennifer Warnick, Tiara Adams, HG McFadden, Juned Siddique, Donald Hedeker, and Bonnie Spring. Design and protocol of a randomized multiple behavior change trial: Make better choices 2 (mbc2). *Contemporary clinical trials*, 41:85–92, 2015.

[51] Novi Quadrianto, Alex J Smola, Tiberio S Caetano, and Quoc V Le. Estimating labels from label proportions. *Journal of Machine Learning Research*, pages 2349–2374, 2009.

[52] Ariadna Quattoni, Sybor Wang, Louis-Philippe Morency, Michael Collins, and Trevor Darrell. Hidden conditional random fields. *IEEE Trans. Pattern Anal. Mach. Intell.*, 29(10):1848–1852, 2007.

[53] Vikas C Raykar, Shipeng Yu, Linda H Zhao, Anna Jerebko, Charles Florin, Gerardo Hermosillo Valadez, Luca Bogoni, and Linda Moy. Supervised learning from multiple experts: whom to trust when everyone lies a bit. In *Proceedings of the International Conference on Machine Learning*, pages 889–896. ACM, 2009.

[54] Jordan Reynolds and Kevin Murphy. Figure-ground segmentation using a hierarchical conditional random field. In *Computer and Robot Vision*, pages 175–182. IEEE, 2007.

[55] Akira Saito, Yoshihiko Nankaku, Akinobu Lee, and Keiichi Tokuda. Voice activity detection based on conditional random fields using multiple features. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[56] Nazir Saleheen, Amin Ahsan Ali, Syed Monowar Hossain, Hillol Sarker, Soujanya Chatterjee, Benjamin Marlin, Emre Ertin, Mustafa Al'Absi, and Santosh Kumar. puffMarker: A Multi-sensor Approach for Pinpointing the Timing of First Lapse in Smoking Cessation. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 999–1010, 2015.

[57] Sunita Sarawagi and William W Cohen. Semi-markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems*, pages 1185–1192, 2004.

[58] Peter Schulam and Suchi Saria. Reliable decision support using counterfactual models. In *Advances in Neural Information Processing Systems*, pages 1697–1708, 2017.

[59] Alexander Schwing, Tamir Hazan, Marc Pollefeys, and Raquel Urtasun. Efficient structured prediction with latent variables for general graphical models. *arXiv preprint arXiv:1206.6436*, 2012.

[60] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52(55-66):11, 2010.

[61] Qinfeng Shi, Li Cheng, Li Wang, and Alex Smola. Human action segmentation and recognition using discriminative semi-markov models. *International journal of computer vision*, 93(1):22–32, 2011.

[62] Saul Shiffman, Arthur A Stone, and Michael R Hufford. Ecological momentary assessment. *Annual Review of Clinical Psychology*, 4:1–32, 2008.

[63] Hyun Oh Song, Ross B Girshick, Stefanie Jegelka, Julien Mairal, Zaid Harchaoui, Trevor Darrell, et al. On learning to localize objects with minimal supervision. In *Proceedings of the International Conference on Machine Learning*, pages 1611–1619, 2014.

[64] Jaeyong Sung, Colin Ponce, Bart Selman, and Ashutosh Saxena. Unstructured human activity detection from rgbd images. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 842–849. IEEE, 2012.

[65] Charles Sutton and Andrew McCallum. *Conditional probabilistic context-free grammars*. PhD thesis, Citeseer, 2004.

[66] Kevin Tang, Li Fei-Fei, and Daphne Koller. Learning latent temporal structure for complex event detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1250–1257. IEEE, 2012.

[67] Ben Taskar, Carlos Guestrin, and Daphne Koller. Max-margin markov networks. *Advances in neural information processing systems*, 16:25, 2004.

[68] Edison Thomaz, Irfan Essa, and Gregory D Abowd. A Practical Approach for Recognizing Eating Moments with Wrist-mounted Inertial Sensing. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 1029–1040. ACM, 2015.

[69] Jaree Thongkam, Guandong Xu, Yanchun Zhang, and Fuchun Huang. Support vector machine for outlier detection in breast cancer survivability prediction. In *Asia-Pacific Web Conference*, pages 99–109. Springer, 2008.

[70] Bill Triggs and Jakob J Verbeek. Scene segmentation with conditional random fields learned from partially labeled images. In *Advances in Neural Information Processing Systems*, pages 1553–1560, 2008.

[71] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research*, pages 1453–1484, 2005.

[72] Ioannis Tsochantaridis, Thorsten Joachims, Thomas Hofmann, and Yasemin Altun. Large margin methods for structured and interdependent output variables. In *Journal of Machine Learning Research*, pages 1453–1484, 2005.

[73] Arash Vahdat. Toward robustness against label noise in training deep discriminative neural networks. In *Advances in Neural Information Processing Systems*, pages 5601–5610, 2017.

[74] Yonatan Vaizman, Katherine Ellis, and Gert Lanckriet. Recognizing detailed human context in the wild from smartphones and smartwatches. *IEEE Pervasive Computing*, 16(4):62–74, 2017.

[75] TLM Van Kasteren, Gwenn Englebienne, and Ben JA Kröse. Activity recognition using semi-markov models on real world smart home datasets. *Journal of ambient intelligence and smart environments*, 2(3):311–325, 2010.

[76] Sofie Verbaeten and Anneleen Van Assche. Ensemble methods for noise elimination in classification problems. In *International Workshop on Multiple Classifier Systems*, pages 317–325. Springer, 2003.

[77] Alexander Vezhnevets, Vittorio Ferrari, and Joachim M Buhmann. Weakly supervised structured output learning for semantic segmentation. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 845–852. IEEE, 2012.

[78] Tim Vieira and Jason Eisner. Learning to prune: Exploring the frontier of fast and accurate parsing. *Transactions of the Association for Computational Linguistics*, 5:263–278, 2017.

[79] Andrew Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE transactions on Information Theory*, 13(2): 260–269, 1967.

[80] Benjamin N Waber, Daniel Olguin Olguin, Taemie Kim, and Alex Pentland. Productivity through coffee breaks: Changing social networks by changing break structure. 2010.

[81] David J Weiss and Benjamin Taskar. Structured prediction cascades. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 916–923, 2010.

[82] D Randall Wilson and Tony R Martinez. Instance pruning techniques. In *Proceedings of the International Conference on Machine Learning*, volume 97, pages 403–411, 1997.

[83] Yan Yan, Rómer Rosales, Glenn Fung, Mark W Schmidt, Gerardo Hermosillo Valadez, Luca Bogoni, Linda Moy, and Jennifer G Dy. Modeling annotator expertise: Learning when everybody knows a bit of something. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 932–939, 2010.

[84] Jun Yu, Weng-Keen Wong, and Rebecca A Hutchinson. Modeling experts and novices in citizen science data for species distribution modeling. In *Proceedings of the IEEE International Conference on Data Mining*, pages 1157–1162. IEEE, 2010.

[85] Jian Zhang and Yiming Yang. Robustness of regularized linear classification methods in text categorization. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 190–197. ACM, 2003.

[86] Zhi-Hua Zhou. A brief introduction to weakly supervised learning. *National Science Review*, 2017.

[87] Ji Zhu and Trevor Hastie. Kernel logistic regression and the import vector machine. In *Advances in neural information processing systems*, pages 1081–1088, 2001.