Semi-Supervised Training for Infrastructure Mediated Sensing: Disaggregated Hot and Cold Water Sensing with Minimal Calibration

Eric C. Larson

A dissertation submitted in partial fulfillment
of the requirements for the degree of

Doctor of Philosophy

University of Washington

2013

Reading Committee:
Shwetak N. Patel
Les Atlas
Mari Ostendorf
James Fogarty

Authorized Program:
Department of Electrical Engineering

University of Washington

Semi-Supervised Training for Infrastructure Mediated Sensing: Disaggregated Hot and Cold Water Sensing with Minimal Calibration

Eric C. Larson

Chair of Supervisory Committee
Associate Professor Shwetak N. Patel
Electrical Engineering, Computer Science and Engineering

Abstract

This thesis formalizes and evaluates the use a central sensor for detecting and classifying water usage in a home or apartment building. While the discussion largely explores the theory and implementation of the sensing, the main impact of this thesis is to address a single assertion: we are running out of water. With this sensing technology, my aim is to support initiatives in increasing awareness of water consumption and understanding of the diversity in ways that vast populations consume water.

Through my sensing, I classify (1) what type of valve was activated in a home (*i.e.,* faucet, dishwasher, toilet), (2) the temperature of water that the valve uses (*i.e.,* hot, cold, or mixed), and (3) where in the home or apartment the valve resides (*i.e.,* kitchen, master bathroom). This is typically known as disaggregated water sensing or end-use water sensing. In my approach, I use a central sensing point based on two pressure sensors. The sensors monitor pressure fluctuations in the home, providing a time series upon which to classify events. I formalize the feature extraction process using template matching, expertly chosen features, and sparse codebooks. Ultimately I show that few features can be created that generalize across residences, necessitating that each residence go through a calibration process. I formalize this calibration process using semi-supervised techniques such as expert knowledge through a rule-based classifier, co-training, and virtual evidence. I evaluate the system in several studies, including a longitudinal deployment of 15,000 water usages across a 1 month period in five homes. The results show that, with about 30-40 labeled examples, the system is 93% accurate at detecting the type of fixture used, 89% accurate at detecting which fixture in the home is activated, and 81% accurate at also detecting the temperature state of the valve. These accuracies are reported on a test set of 2,500-4,000 water usage events per residence. This thesis, then, has implications in the water sustainability community, activity detection community, and the machine learning and signal processing communities.

For my children

In hope that anyone can leave the world in a better condition than when they found it

Acknowledgments

This thesis represents a body of work over several years in which I devoted my studies and efforts to collecting and analyzing water usage. During that time I have been influenced positively by a number of individuals.

I would like to thank my committee for their thoughtful commentary and helpful revision suggestions. They have truly helped me to formulate the probing, scientific questions with which any scholarly work should adhere. Thank you especially to Mari Ostendorf for her suggestions in active learning research.

I would like to thank my advisor Shwetak Patel and my co-advisor Les Atlas for their time in reviewing methodologies and their helpful suggestions in making the work more scholarly. Countless brainstorming sessions were pivotal in the exploration of this work. I aspire to be a mentor to other students as you both have been mentors to me. To Shwetak, I have never known someone so creative. I cannot remember how many times I came to you with a problem that I thought was insurmountable, only to leave our brainstorming session with no doubt in my mind that I could solve it.

I would like to thank my wife and family for their devotion, love, and understanding while I carried out this work. Chelsea, you are simultaneously what keeps me grounded and what keeps me reaching for the sky.

I would like to thank the members of my lab who helped me to build and deploy an array of different sensors in their homes, especially to Tim Campbell, Elliot Saba, TienJui Lee, Gabe Cohn, and Eric Swanson. Without your friendship this research would have been far less enjoyable and orders of magnitude more difficult.

Lastly I would like to thank Jon Froehlich for the countless hours that he spent with me collecting, analyzing, and fine tuning the ideas to make this research possible. Jon, you are like a co-advisor to me. I am a better researcher, better communicator and writer, and better person from your mentorship.

Table of Contents

List of Figures

List of Tables

CHAPTER I

1. Chapter One:
Semi-Supervised Learning and Infrastructure Mediated Sensing

In this thesis, I leverage a number prior works in water sensing, signal processing, and machine learning. As a researcher in signal processing, sensing, machine learning, and ubiquitous computing, my work has a number of contributions in cross-disciplinary domains. Because of the potential diversity in readership, this thesis outlines not only the signal processing and machine learning contributions, but also the contributions of my work in terms of sustainability, embedded sensing, and water end-use classification.

This dissertation introduces and formalizes how machine learning and signal processing techniques can enable multi-disciplinary initiatives in water sustainability. In this document I explore the idea of using a central sensing point to infer water consumption and patterns of water consumption—the idea is explored from its inception, to its feasibility, to practical considerations of its calibration and installation. On one hand, the system is a simple pressure sensor. On the other hand, it is a complex inference engine that requires sophisticated algorithms to be practical—practicality built upon a coalescence of multi-view classification, expert knowledge, sparse coding, and active learning. While the theoretical foundations of each technique are explored and evaluated, they are not the central focus. Through this work, I hope to effect real change in reducing our environmental water footprint. I believe in the idea that awareness and feedback can be the driving forces in that change, but that our methods of sensing are not yet up to the task. This work, then, is the bridge between current sensing limitations and the needs of sustainability initiatives, like eco-feedback. I address the most pressing shortcomings of current sensing approaches: (1) cost, (2) ease of physical installation, (3) maintenance, and (4) calibration. Each shortcoming has the potential to be a roadblock to adoption. Therefore I have painstakingly outlined the methods by which such a *disaggregation* system can be practically installed and trained.

Disaggregation, in terms of water sensing, refers to the ability of a system to sense which fixtures in the home are using water. It can also refer to a system that can disaggregate between hot water use and cold water use. I refer to the disaggregation system in this dissertation as HydroSense.

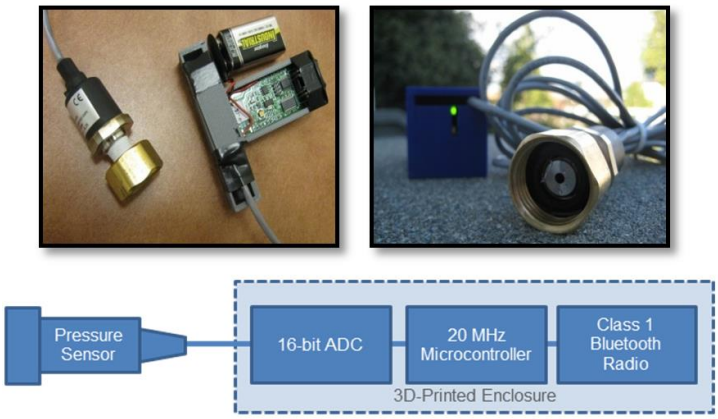

Figure 1-1. The HydroSense System

HydroSense is a pressure based water disaggregation system (Figure 1-1). It attaches to the existing plumbing infrastructure and infers water use throughout the home from a single point [59,85]. The key insight in HydroSense is that when water is not used in the home, the pressure in the home is constant and the plumbing system is in a state of equilibrium. When water is used, the plumbing system must shift to another state of equilibrium, which causes a pressure drop in the entire plumbing system and briefly causes pressure waves as the system shifts (*i.e.*, transients). Similarly, when the water is turned off, there is a pressure increase and transient pressure waves are created as the system shifts back to stable pressure (Figure 1-2). The pressure change and the pressure waves are different depending on which water fixture in the home is activated. Thus, HydroSense can identify fixtures based on observed pressure waves as long as it has a library of examples.

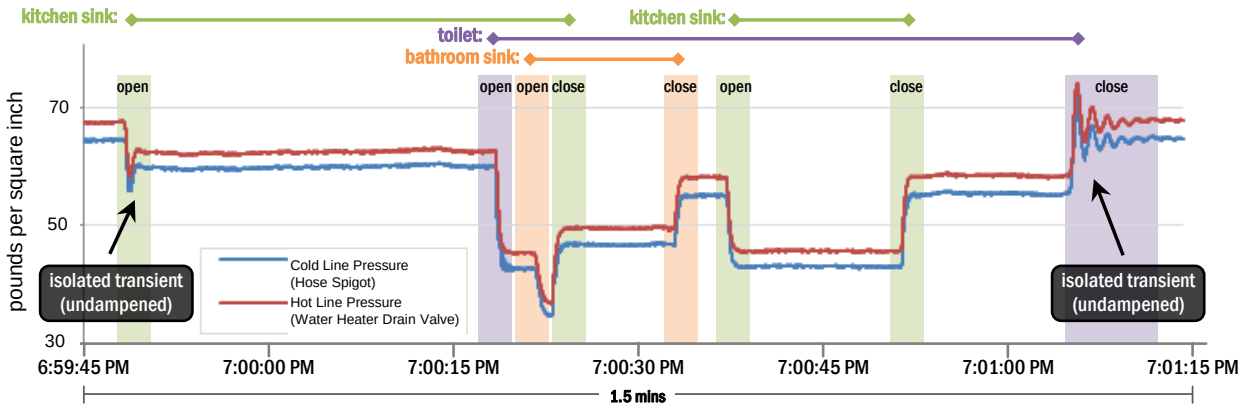

Figure 1-2 Example open and close pressure waves in isolated and compound cases.

In Chapter IV, I will introduce the HydroSense theory of operation, showing that "events" of interest are easy to segment but not label. I will also introduce a number of features used in HydroSense, only a

fraction of which generalize across homes, and many features that require periodic re-calibration as the seasons change and water usage behavior shifts.

The characteristics of an ideal HydroSense classifier can be summarized as:

- **Efficient initial calibration**: a system that can be efficiently trained for each home.
- **Model updating**: a system that can be retrained efficiently.
- **Model storage**: a classifier should be able to be held as compact as possible on a server so that many models from many homes can be stored efficiently.
- **Generalizing feature integration**: have the ability to incorporate trained models of generalizing features from other deployments and incorporate some labeled instances (*i.e.,* if the homeowner performs some initial calibration).
- **Prior domain knowledge**: have the ability to incorporate knowledge from previous water end-use studies to guide training. This is subtly different from generalizing features—the domain knowledge might be in regards to how one expects certain water usages to cluster. These observations are not directly used as features for machine learning, but can be used to guide the design of a machine learning model.

There are a number of interdisciplinary motivations for this work, which I outline below: (1) a practical system for water end use data collection, (2) a practical application of semi-supervised training and active learning, and (3) a new paradigm in infrastructure mediated sensing.

1.1 More aware, less water

There is consistent and evolving interest in understanding the way in which individuals use water. For instance, dating back thirty years there have been numerous studies that investigate the end-uses of water. Such studies were diverse in their actual aims—some wanted to increase community confidence in the future of the water supply, some wanted to build scientific knowledge into the policies governing rebates and conservation programs, and still others simply wanted to investigate wasteful practices [6,8,27,159]. Perhaps surprisingly, many utility companies have been at the forefront of understanding water waste—not only leaks, but actually getting individuals to use less water. From the utility perspective, less water use means fewer distribution costs, fewer pumping stations for new constructions and less legal controversy over an ever-dwindling water supply [18]. This is the fundamental motivation of sustainable water usage: we are using water faster than we replenish it [56]. New dams are not being built, new aquifers are not being discovered, and rivers and lakes are drying up. This is a problem for all individuals, not just utilities. As water becomes scarce, countries are throwing expensive resources at the problem.

China has started melting snow to increase water flow to lakes and streams, Spain is importing millions of gallons of water by barge, and, perhaps the most unsustainable, Saudi Arabia uses massive oil stocks to run desalinization plants [18].

The ability to breakdown water usage enables utilities to use tiered pricing models, enables the creation of more efficient plumbing codes, enables governments to evaluate the return on investments in a variety of conservation programs, and allows individuals to examine their own wasteful practices. This last assertion is particularly important because it helps to initiate and sustain behavior change by increasing awareness of water usage (*i.e.*, what does it cost to take a hot shower?). The metaphor is simple: if you went to a grocery store and at the end of the trip you received a receipt indicating you purchased "4,700 food units" rather than an itemized receipt, then how diligent would you be about the way you shop? This is the exact system we have in place for home utilities. My work strives to itemize this water usage down to each fixture in the home and break down the amount of hot and cold water used at each fixture. And, because I can look at hot water separately, it opens up a host of applications for increasing the efficiency in hot water use and distribution—which is the second largest electricity consumer in the average home (14% of all electricity usage) or is the highest energy consumer in many areas such as the pacific northwest (25% of electricity usage) [38].

Beyond sustainability: Because of the itemized breakdown of water usage, such a system also has many implications outside of sustainability [53]. For instance, activity inference and health applications can benefit greatly from such a system. One can easily imagine a system for "elder care" applications that monitors the water habits of an individual and updates the care giver on the number of showers taken, restroom, and kitchen usage—or even a system that looks for anomalies in water habits to detect the onset of dementia. Because many hygienic practices revolve around water usage, a number of health applications could use data from the system. This includes monitoring drinking and bathing habits, as well as diagnosing/managing an overactive bladder. The system may also have utility as a screening tool for congestive heart failure as, according to the American Heart Association, one of the first signs of a congestive heart failure is frequent urination at nighttime.

1.2 Motivation: Limited Applications in Semi-supervised Learning

Semi-supervised training of time-series classifiers has had limited but steady success over a number of years, but is still dramatically underutilized in modern applications [102]. Semi-supervised methods attempt to address the main limitation of traditional machine learning—traditional methods require labeled instances across all classes to ensure high-performing classification. When instance labeling is difficult, a more appropriate design is to train a classifier with a semi-supervised approach. For instance,

co-training has been used successfully over a number of different applications [50]. In this scenario, different classifiers are seeded with a few labeled examples and the output labels at each stage are used to bootstrap future training sessions, rather than requiring a ground truth label for each instance. Thus, a pool of unlabeled instances can be used to tune the parameters of a model. In still other applications, virtual evidence is used to reflect beliefs of experts about state assignments in a time series classifier [125].

Semi-supervised methods have seen limited use, partly because of the absence of a freely available and simple toolkit [120], and partly because of their limited application to existing problem spaces. Mann and McCallum explain the problem as,

> *Semi-supervised learning, where a small amount of human annotation is combined with a large amount of unlabeled data to yield an accurate classifier, has received a significant amount of attentions from the research community. However, there are surprisingly few cases of its use in applications, where the emphasis is on solving a task, not on advancing theoretical understanding. … due in large part to the inherent difficulty of the task and the unreliability of existing methods* [102].

I would add to this assertion that semi-supervised techniques are often not as cost effective for most applications as investing in attaining labels. Many times it is a better use of resources to extract labels and train traditional machine learning classifiers. If the results generalize, then the labeled data is vastly more useful than any semi-supervised assumptions. However, there are situations where the cost of gathering labeled data is too great. Either the generalization of the labeled examples is not "good" enough to warrant the resources for collecting the labels (*i.e.,* the system changes too much for different use cases), or the labels have a limited useful lifespan (*e.g.,* a system changes so dramatically that collecting labels now will not make any difference in the performance in a week). In these cases, a semi-supervised procedure is well warranted, but not many application spaces exist that have such limitations. As explained in the next section, the HydroSense application is ideally suited for semi-supervised techniques, indeed requiring semi-supervised techniques to be practical.

Another machine learning technique that has received considerable study is the field of active learning. In active learning, the idea is to query an "oracle" for labels to different instances [152] (this scenario sometimes more specifically referred to as active labeling). The use cases are similar to that of semi-supervised training—because it is expensive to attain labels, we only want to selectively ask for labels to the most informative examples (*i.e.,* to make the most of our unlabeled data). Active learning has received considerable use in the fields of speech recognition and labeling tasks in natural language

processing [117,147]. Although active learning has been employed in image classification [146] and even cancer diagnostics [96]. However, there are far fewer applications of active learning in tasks involving streaming data [4]. In these scenarios, the task is to decide whether or not a streaming input should or should not be queried—a form of "online" active learning. While active learning has seen varied use in applications such as visual object recognition and language processing, the majority of studies of active learning are focused on their theoretical mechanisms and bounds, rather than practical applications.[1]

1.3 Motivation: IMS as a Practical Application

In the area of ubiquitous computing, a unique application space for semi-supervised and active learning exists in Infrastructure Mediated Sensing (IMS). IMS uses one or two sensors attached strategically to a building's infrastructure (*i.e.*, on the electrical, plumbing, or air duct systems) to infer the uses and activities of everything attached to the same infrastructure throughout the building. For example, sensing that someone has entered a room from the pressure noise that is back-propagated through the HVAC system [122]. However, IMS systems are fundamentally limited by the information they can infer. To remain accurate IMS systems must be calibrated to every infrastructure to which they attach, and trained to classify every event type from the output of the sensor. Each installation point can easily collect "events of interest" but labels for the events may be difficult or impossible to obtain. For instance, in the given example the IMS system would need an example of what noise in the HVAC duct looked like as a person entered every room in the building and possibly additional examples for different sized people—and the training might be different if one of the doors in the building is closed, changing the HVAC system transfer function. By adopting the techniques of semi-supervised methods, like co-training [15], and leveraging prior work from active learning [152], IMS systems can be trained with only a few labeled instances.

I also use a framework that allows the incorporation of expert knowledge into a graphical model, known as virtual evidence [12]. Virtual evidence allows the incorporation of prior knowledge from different datasets in different partitions of the graph—a powerful tool in a cloud computing architecture. Finally, I explore the inclusion of an automatic feature extraction method—taking unlabeled waveforms and creating a codebook of features automatically from the waveforms, commonly referred to as sparse coding. This sparse coding solution is particularly interesting because it requires no knowledge of signal processing, time series classification, or machine learning, in general. Other problem domains could use these techniques to quickly prototype ideas—a black box model of time series data.

[1] This is not to say that applications in active learning and semi-supervised learning do not exist. Only that the largest bodies of work favor explorations of their theoretical aspects.

This thesis adapts semi-supervised and active learning techniques to IMS systems. Such methods allow IMS systems to be trained automatically after a brief installation period, and then use a cloud computing architecture to perform training and inference. As such, the methods herein solve the inherent calibration problem that limits widespread adoption of IMS and systems like IMS. I specifically focus my attention on IMS for plumbing systems because the signals do not generalize across different plumbing infrastructures, providing a challenging test case. I also discuss ways of continually updating the inference model through active learning, and ways of offloading certain feature calculations for optimal performance in the cloud. In this way, this work provides an outline for others working on problems where (a) labeled data is inherently difficult to attain, (b) where an ever adapting model is preferable to one in which training only occurs once, or (c) where it is possible to collect the most useful labels for training a system.

1.4 Contributions

The core advancements are direct outcomes of this thesis and contribute to the ultimate goals of my research in machine learning, ubiquitous computing, and sustainability (see Chapter IX for a more in-depth outline of my contributions):

The core advancements are direct outcomes of this thesis and contribute to the ultimate goals of my research in machine learning and ubiquitous computing. They include:

1. An improved training method for semi-supervised learning for time series classification that is:
    a. able to leverage co-training, expert knowledge, and virtual evidence in its decision making process,
    b. can incorporate different knowledge sources from different data sets,
    c. allows changes in confidences from an expert in the field without retraining the entire system
    d. easily combined with techniques of active learning to further minimize the number of labeled training examples
2. An evaluation of performance of using a few labels on real world HydroSense data streams, specifically investigating:
    a. co-training methods,
    b. the tradeoff between required number of unlabeled instances and accuracy,
    c. convergence time of the system,
    d. the schedule of calibration for the system,

e. the infusion of different knowledge sources through sparse coding and expertly selected features,
f. a strategy for active labeling that quickly boost the performance of the classifier, without placing an undue burden on the homeowner

The other core advancements pertain to the sustainability community, providing,

1. Larger and more comprehensive studies of the end uses of water. I consider these the "first adopters" of this type of technology. A semi-supervised, low cost framework allows institutions to conduct larger studies, faster. This allows the evaluation of wasteful practices, rebate policies and incentives, outreach campaigns, and comparative studies upon different groups (*i.e.*, different socio-economics, geography, country, *etc.*) [7,27,160].
2. More comprehensive control and understanding of hot water use [134]. By incorporating a technology like HydroSense, one can more adequately control when and how to manage water heating, the second highest consumer of electricity in the US [38].
3. A dataset of ground truth hot and cold water end uses available for the water resource community. Although I use this to evaluate my methods, the ground truth labels are of immediate utility to many institutions.
4. Awareness in everyday life, providing a scalable method for monitoring water usage through eco-feedback displays. HydroSense also has the ability to further research in the eco-feedback community by providing disaggregated feedback that can be used in deployments comparing different incentive structures, displays, and design goals [53,58,65].

1.5 Declaration of previous work

- Parts of Chapter II and Chapter IV appear in the Journal of Pervasive computing, 2012 [85]
- Part of Chapter IV, VI, and VII was also published in the international conference of Ubiquitous Computing 2009 and in the international conference on Pervasive Computing in 2011 [59,86]

Although the initial drafts of some of the text have been previously published, a significant portion has been revised and in many cases rewritten. Data and analysis have been updated to reflect the most recent state of all databases and algorithms.

1.5.1 Parallel Research in HydroSense not addressed in this Dissertation

This dissertation focuses on the ability of HydroSense to identify water fixtures based upon the pressure waves generated in a home. One aspect that is not explicitly discussed is the ability of HydroSense to

sense the *water volume* used. However, I have investigated this problem in several publications [59,85], and extended the work to detect the amount of hot water versus the amount of cold water [141]. The flow sensing work was a collaboration with Eric Swanson during his master's thesis. We evaluated 10 different homes, activating every fixture in the home at five different flow rates and three different temperature combinations for each flow rate—an exhaustive procedure designed to test the limits of flow sensing possible with HydroSense. We showed that the accuracy of sensing the flow rate with HydroSense was comparable to a typical water meter using a regression known as encapsulated k-Nearest Neighbor (E-kNN). Additionally, we showed that hot- and cold-water use could be efficiently separated using independent components analysis (if two sensors were used) or matrix factorization (if using only one sensor). This work specifically targeted how much calibration was necessary to make the flow rate calculation accurate, showing that one could calibrate the flow rate calculation using only a few examples from the kitchen sink and bathtub [141]. This work did not incorporate a prediction of what fixture was consuming water; only a prediction of the flow rate.

In other work, I evaluated how to display data from HydroSense so that it is actionable for a homeowner, such that as much information can be extracted from a glance-able display [58]. This was a collaboration with Jon Froehlich during his dissertation. We showed that homeowners believe they can be effectively motivated to use less water with a various bar graph displays, and that most homeowners wish for the option to see what exact fixture in the home is using water (*i.e.,* which specific sink in the home uses most water) and to be able to group fixtures by fixture category (*i.e.,* how much water was used by all the toilets in the home). Additionally, homeowners wanted to have that information broken down by hot and cold water. These are important design considerations that affect the expectations of a system such as HydroSense and what the eventual design goals should be tailored to reach in an unsupervised system.

Because the flow sensing work was the central finding of Eric Swanson's master thesis and because the eco-feedback work was a core contribution of Jon Froehlich's dissertation, they are not included as core contributions on this dissertation. This is also done to prevent any dilution of this thesis work and their thesis works. As discussed, the main contribution of this dissertation is the evaluation and application of semi-supervised and active learning methods for water usage classification. This thesis also has broader impacts than just those in sustainability as outlined in the previous section.

1.6 Navigating this thesis

The outline of this thesis is as follows. There are chapters and sections that can be skimmed or left out depending on readership. For those in machine learning and signal processing community, Chapter III,

IV, and VI explain the feature sets and algorithms used in the analysis. For those in the sustainability community, Chapters VIII and IX explain the results and implications of the system and will be of primary interest.

- Chapter II explains previous work in water end-use sensing and motivates the problem. This chapter is meant largely for those in sustainability engineering and the interested reader.
- Chapter III explains various machine learning techniques including semi-supervised approaches and active learning approaches.
- Chapter IV explain the theory of operation for the system and the results of the initial feasibility study, showing that water pressure waves in common plumbing systems are unique to the water fixture that originated them.
- Chapter V describes the comprehensive water usage dataset and the means by which I collected the ground truth labels.
- Chapter VI then presents the methodology for extracting features from a pressure stream. In particular I review template matching, features chosen by expert knowledge, and sparse codebook features.
- Chapter VII presents the previous methods and results I have achieved in water-end use classification using the system (HydroSense). Chapter VII closes by explaining the disadvantages of supervised approaches. This chapter is meant mainly for readers with experience in signal processing and machine learning—and is meant for those who wish to reproduce the work.
- Chapter VIII then discusses the semi-supervised machine learning framework that includes the use of co-training, rule based classifiers, and virtual evidence. I show that an active labeling system can be created with average accuracy greater than 80% at distinguishing the exact valve in the home activated and its temperature state—with only a handful of labeled examples.
- Chapter IX outlines the associated contributions for different communities.

CHAPTER II

2. Chapter Two:
Background and Related Work on Infrastructure Mediated Water Sensing

This chapter summarizes the current state of the art in infrastructure mediated sensing, with emphasis to disaggregated water sensing. First, I give a brief introduction to infrastructure mediated sensing and, second, a history of disaggregated water sensing. The casual reader can read the first section to get a better idea about the inherent problems with calibration in infrastructure mediated sensing. The final section can be skipped for the casual reader.

2.1 Infrastructure Mediated Sensing

At its core, HydroSense is a part of a broader research field known as Infrastructure Mediated Sensing or IMS. IMS was borne out of the struggles with distributed, direct systems used for activity detection in the home. Distributed direct sensing can be based on computer vision [23], microphones within the living environment [29], many simple sensors throughout the home (*e.g.,* reed switches on cabinet doors, accelerometer-based object manipulation sensors, infrared motion and break-beam sensors [142,143,161]), or a smaller number of targeted direct sensors (*e.g.,* strain sensors under floorboards at strategic locations [129]). Direct sensing provides valuable insight into home activities, but comes with practical costs. Installation and maintenance can be cost-prohibitive, direct sensing can create privacy concerns and a feeling of stigmatization (especially with cameras or microphones), and a littering of sensors throughout a home can be problematic with children and pets [10,68].

IMS has therefore evolved to help address many practical obstacles to everyday deployment of home activity sensing, but is inherently limited by what information can be practically and reliably extracted from a home's infrastructure. Recent work has examined whole-home activity sensing using just a handful of inexpensive sensors at strategic locations in a home's existing infrastructure, for example:

- A home's **water** infrastructure [53,57,59],
- **Electrical** infrastructure [60,64,123,124],

- Natural **gas** and **propane** infrastructure [35], and
- **HVAC** infrastructure [122] infrastructures.

The central idea in infrastructure-mediated sensing is to recognize human activities, such as running a dishwasher, turning on a reading light, or walking through a doorway, according to their manifestations in these existing home infrastructures. These manifestations can be used for monitoring resource usage (like HydroSense), for monitoring user activity in the home, or even for interacting with existing devices in the home [63].

The limitations of existing approaches are especially salient for water. Fogarty *et al.* used microphones pressed against the exterior of a home's major water pipes to demonstrate recognition based on patterns of water use [57]. However, they could not reliably differentiate among multiple instances of similar fixtures or identify concurrent activities and did not attempt to estimate the volume of water being used. The original work, HydroSense, was the first to show that pressure transients could be used to disaggregate water fixtures. HydroSense and its limitations are discussed in detail in Chapter IV and VII.

2.2 Existing Approaches To Disaggregated Water Sensing

This section highlights the use of disaggregated water sensing and its use in varied application spaces. Aside from HydroSense, there have been a number of sensing projects that try to disaggregate what fixture in the home is using water. These approaches normally use one or more flow meters and use volume and instantaneous flow rates to classify the fixture used. There are two areas of related work in disaggregated water sensing: disaggregated water flow sensing and disaggregating between hot and cold water usage. I address each of these related areas in turn. Table 2-1 provides a summary of the water sensing disaggregation related work and its evolution over time.

2.2.1 Disaggregated Water Flow Sensing

There are two basic approaches for measuring water flow: *inline direct* flow measurement and *non-intrusive flow* estimation. Inline systems typically use either positive displacement or velocity measurements to calculate flow. Positive displacement relies on water to physically displace the measuring element (*e.g.*, an oscillating piston or rotating disc) in proportion to water flow volume. Most residential meters use positive displacement because it is generally accurate at the low to moderate flow rates found in the residential sector [130]. Non-intrusive flow estimation systems use ultrasound and lasers to sense the water flow from the outside of the pipe (Figure 2-1, middle). These systems are typically used in manufacturing and industrial processes because of their accuracy and high cost.

Figure 2-1. A residential water meter (left), ultrasonic meter (middle), and smart water meter (right).

Other flow sensing methods have emerged from the UbiComp and pervasive communities that retrofit sensors on the piping. Kim *et al.* demonstrated the use of an aggregate water flow meter together with a network of accelerometers on pipes to infer flow rates throughout a home [80]. This approach requires placement of multiple sensors along water pipe pathways that are uniquely associated with each fixture of interest (*i.e.*, a distributed direct sensing method). This system does not explicitly disaggregate water use, but can tell which pipes in a plumbing system have water flowing. Thus, with enough sensors, disaggregation should be possible. Selover *et al.* used wireless flow sensors installed inline with every exposed fixture in the home [134].

The next generation of resource measurement systems (often referred to as "smart meters") will soon provide real-time (or near real-time) data on electricity, gas, and water usage in homes and businesses (Figure 2-1, right). It is unclear, however, if this data will be open or closed (*i.e.*, proprietary) and what temporal resolution will be available (most smart meters advertise fifteen minute interval data). There has been some work on disaggregating water use based on the low sampling rates from a smart meter [28]. However, there are considerable confusions when more than one water fixture is used and most sink usage is not detectable. Even so, some laundry and shower use can be detected with accuracies in the 70-80% range. These accuracies are reported on a synthetic dataset built from staged experiments.

2.2.1.1 Flow trace analysis

For those water meters not yet upgraded to their smart meter counterparts, a popular and low-cost non-intrusive retrofit has emerged using magnetic sensors. Most residential meters use magnetic coupling to transfer data between the rotating disc, which spins proportional to flow rate, and the counter unit, which converts this spin rate to flow. Magnetic sensors, such as a hall effect sensor, can be placed at the top or bottom of a residential water meter to sense this spinning magnetic field and convert the spin rate to flow (*e.g.,* see [31] and [103]). This approach was proposed by Dziegielewski, *et al.* in 1992 and is now the most common water usage event identification technique used in research studies, called *flow-trace analysis* [52]. Flow-trace analysis works by analyzing the aggregate flow trace patterns off of an inline

water meter to determine the source of the water usage event. Flow-trace analysis relies on the fact that most residential water uses exhibit highly consistent behavior over time (*i.e.,* a specific toilet will flush with the same volume and flow; similarly a specific dishwasher will exhibit the same series of flow patterns each time it is run [103]).

Flow-trace analysis has since been used in a number of government- and utility- sponsored studies of residential end uses of water [42,43,97,104] (see also Table 2-1). The most popular of which is a large study conducted by the American Water Works Association Research Foundation (AWWARF), which used AquaCraft's flow-trace analysis toolkit called *Trace Wizard* to study residential end uses of water in 1,188 households across 12 study sites in America and Canada [103]. The water data was collected using the aforementioned magnetic sensing retrofit solution. The data collection took place in the winter and the summer months from 1996 to 1998 and resulted in 1.9 million water usage events. A total of four weeks of water data per home was collected in separate two-week intervals. This study provided the canonical dataset upon which most US government statistics of residential end uses of water are based [153]. In the only known empirical investigation, Wilkes *et al.* conducted staged experiments of water usage over a five day period in one home [158]. Flow-trace analysis correctly categorized 83% of the *isolated* water usage events at the fixture category level. When water usage overlapped (*i.e.,* what I term *compound events*), performance dropped dramatically to 24% when two water fixtures were used in compound and 0% when three or more were used [158]. In addition, although flow-trace analysis is capable of classifying at the fixture category level, it cannot be used to determine the specific fixture or valve that was used (*i.e.*, it can sense that *a* toilet was flushed, but not *which* toilet was flushed).

Finally, in its current form, flow-trace analysis is not completely automated—its primary value is to trained professionals carrying out explicit water research. Usage of AquaCraft's Trace Wizard, for example, is a multi-step, iterative process. The program uses the following set of statistics to help classify water usage events: start time, stop time, duration, volume (gallons), peak flow rate in gallons per minute (gpm), the most common flow rate (gpm), and how often this most common flow rate occurs during the duration of the event. Events are classified according to their similarity with a pre-defined set of manually entered and tuned per-home parameters. For example, a toilet may be defined as using between 3.25 and 3.75 gallons per flush, the peak re-fill flow rate as between 4.2 and 4.6 gpm, the duration of the flush event between 30 and 50 seconds, and the mode flow rate between 4 and 4.5 gpm [103]. AquaCraft estimates that it takes their trained analysts approximately one hour per week of data from a home to complete a flow trace analysis; less time after the parameter file has been properly tuned. Figure 2-2 shows an example flow trace and the sensing attachment.

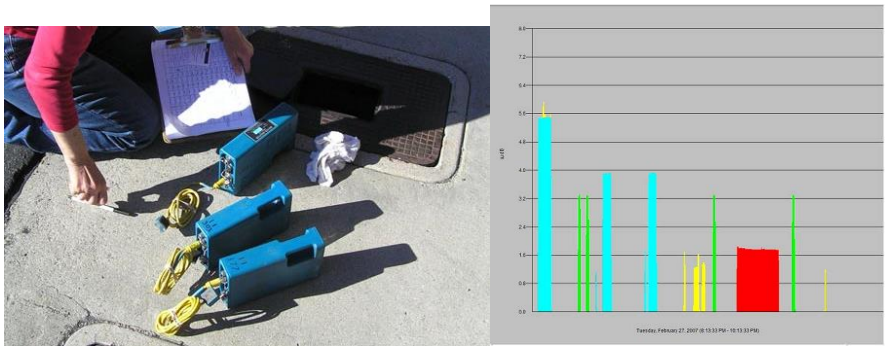

Figure 2-2. Aquacraft loggers and flow trace wizard software.

I note that flow trace analysis is not a competing technology to ours. Rather, the designs are complementary. In my most recent analysis, I use techniques from flow trace analysis to augment the pressure-based analysis. As I move to a semi-supervised system, there are a number of techniques used in flow trace analysis that can help the system to generalize. For example, the volume of water consumed by a toilet is standardized (*e.g.,* 1.6 or 2.2 gallons per flush). This type of knowledge is helpful for creating prior distributions for semi-supervised training of HydroSense.

More recent studies have used flow trace analysis with *journaling* of water activities to increase accuracy of the approach. Beal *et al.* used flow trace analysis along with water end use surveys and self-reported water use on 252 homes in 2010 [6,9]. They advised residents to keep a diary of as many water events as possible for one week. The authors noted that this was not always possible and that it required a "high level of commitment from the participants." They also note that "without a stock inventory and information on water use behavior/patterns for each dwelling it would be extremely difficult to create meaningful and accurate end use templates." Diaries have been shown to be more reliable than self-reported water usage during interviews [119,163]. In addition to water diaries, the authors also had trained technicians go to each residence to perform audits of all the fixtures in the homes and then create specialized templates for each home. They combined these with "stock" templates of water usage from previous studies and then used surveys/questionnaires to shape statistical analyses. It was a resource intensive project and the intention of the study was to leave the devices at the site of each home for a long period. However, sensor malfunctions and limited personnel resources have constrained this effort, hopefully temporarily.

2.2.2 Disambiguating Hot Water vs. Cold Water Usage

The lack of precise measurements about the quantities of hot water used in residences has been an obstacle in the design of hot water systems (*e.g.,* how much hot water do homes require at peak use? [44,67]) and in the analysis of conservation programs (*e.g.,* how much hot water does a low-flow shower

head save? [52]). Hot water usage also highlights the strong interconnection between energy and water conservation efforts in the home. According to the US Department of Energy, hot water heating is the second most energy consuming activity in the home behind air conditioning/space heating. However, *what* hot water usage activities contribute to this consumption are much less understood [66].

The measurement and analysis of domestic hot water consumption is typically accomplished using one of two methods: a temperature-based inference method or a flow-trace analysis method (where the inline meters are installed at the hot water heater intake line [2]). In 1985, Weihl and Kempton conducted the first quantitative study of hot water usage using automatic means [157]. They installed an inline flow meter and a temperature probe at the hot water tank as well as temperature probes at the hot water pipes leading to each fixture. Flow rate and temperature were recorded to a data logger once per minute. From this data, the authors were able to deduce which fixtures were using water, how much hot water was being used, and also calculate pipe heat loss. Ladd and Harrison used a similar distributed sensing approach; instead of temperature probes, however, they used power consumption monitors on the dishwasher and clothes washer, an inline flow meter installed at the clothes washer, and an inline flow meter at the hot water tank [83].

Lowenstein and Hiller (1996) simplified the above approaches by eliminating the need for multiple sensors [97]. They collected flow trace data at 15 second intervals at an inline meter installed at the hot water tank feed line. The volume of flow and the average flow rate for each hot water draw were used to classify the hot water usage events into their end use-category, a technique they referred to as "bin analysis," but is simply a reduced form of flow-trace analysis with fewer parameters (*e.g.*, without time of day, event duration). Although this technique was used successfully in a year-long study of seventeen sites (and repeated by DeOreo in a study of 14 Seattle homes [44]), it was noted by the authors in a follow-up paper [98] to have two major limitations: (1) they were not able to discriminate between end uses when multiple hot water activities occurred at the same time and (2) hot water draws for different end-uses can have the same flow rate and total flow volume, making it difficult to unambiguously associate specific end-uses. Thus, in their follow-up study, they attached thermocouples 1-3 inches downstream from where the hot water line branched, up to three thermocouples per home (depending on its plumbing layout) to mitigate these limitations.

2.3 Summary

In summary, I presented the related work in disaggregated water sensing, concluding that the current approaches have significant scaling issues in terms of the sensing requirements or the limited accuracy. HydroSense does not suffer from many of the downsides in sensing, but requires labeled training data. The HydroSense system presents a unique opportunity for discriminating between hot- and cold-water

usages because it is much easier to install than the above approaches. Pressure waves are shaped not just by the valve type but also by their propagation pathway through the pipe system, allowing HydroSense to disambiguate hot water events from cold water events—even though the hot and cold water valves are right next to each other at a faucet, their pipe pathway through the home is quite different. Of course, this is also the same phenomenon that creates features that do not generalize across different homes—no two plumbing infrastructures are identical, and is a major reason for needing a semi-supervised approach. The non-generalizing feature discussion is explored in depth in Chapter VI.

Study (year)	Flow Measurement?	Hot/Cold Disaggregation	Class. Method	Classification Level	Sensing
Ladd and Harrison (1985)	Hot water only, inline	Yes, sub-metered at water heater	Direct sensing	Hot valves only	Inline flow meters (at washer and hot water tank) Elec. Current sensors
Weihl and Kempton (1985)	Hot water only, inline	Yes, sub-metered at water heater	Direct sensing	Hot valves only	Inline flow meter (at hot water tank) Temperature sensors
Lowenstein and Hiller (1996)	Hot water only, inline	Yes, sub-metered at water heater	Flow-trace analysis	Hot valves only	Inline flow meter at hot water tank
Mayer, *et. al.* (1999)	Yes, inline	No	Flow-trace analysis	Fixture category	Inline flow meter
DeOreo and Mayer (2000)	Yes, inline	Yes, sub-metered at water heater	Flow-trace analysis	Fixture category	Dual inline flow meter (whole home meter plus hot water tank meter)
Fogarty, *et al.* (2006)	No	No	Machine learning	Fixture category	Acoustic sensors on intake pipes and drain pipes
Kim, *et al.* (2008)	Yes, inline	Yes, direct	Direct sensing	Untested (lab study only)	Inline meter accelerometers on pipe
Beal *et al.* (2010, 2011)	Yes, inline	No	Flow-trace analysis and water use diaries	Fixture Category	Manipulated flow meter with attachment

Table 2-1. A view of the related work in the evolution of water usage measurements. Most studies before 2006 were designed to measure end uses of water and thus focus on monitoring water at the fixture category level rather than the individual valve or specific fixture.

CHAPTER III

3. Chapter Three:
Background and Related Work on Supervised, Semi-supervised, and Active Machine Learning

The approaches for water disaggregation were largely built from professionals in civil and environmental engineering and were geared towards modeling water usage templates. Their focus was mainly on finding a quick and reliable way to obtain water end-use using a team of trained professionals. As such, scalability was a secondary concern for the earliest studies. The aim of this thesis is to increase awareness of water use; to create a system that can be used in tens of thousands of homes. Whether it is larger, longer-term professional studies or a system available for the interested homeowner, scalability is of primary interest. It is with this characteristic in mind that I will survey existing machine learning approaches.

This Chapter is divided as follows: first I illustrate the differences between vector based classifiers and sequence based classifiers. I also discuss the difference between discriminative classifiers and generative classifiers, motivating the use of both conditional random fields (CRFs) and a dynamic graphical model (DGM) such as a hidden Markov model (HMM). I then introduce the traditional approach to training a DGM using EM and traditional approach to training a linear chain CRF. Then I introduce approaches to unsupervised learning, semi-supervised learning, and active learning.

3.1 Vector Models

There are many different types of machine learning algorithms (referred to also as learners, classifiers, *etc.*). Often communities and individuals have their own "favorite" classifier for their different applications. In their simplest forms, a learner takes as input a vector of observed features x and creates a predicted class output y. When the input to a classifier is a single array of features and the output is a single class, these are often called instance classifiers or vector classifiers. This is because the output, y, can be estimated from a single vector, x. Vector classifiers come in many forms, Naïve Bayes, k-nearest neighbors (KNN), decision trees [20], support vector machines [55] (SVM), and a list of many others.

The implementations of each approach are not entirely significant here. The general knowledge is that there is no "one-size-fits-all" classifier [49]. Different applications have different performances, and there is really no consistent algorithm that routinely outperforms others. Often, the reason one classifier outperforms another is not obvious for a particular dataset. However, the general knowledge is to start simple and work in more complexity as needed—"simplicity is a virtue" [100]. Moreover, the performance of an algorithm often resides on its ability to generalize for a particular dataset, or its ability to avoid the curse of dimensionality. The best performing algorithms are usually those that employ many different classifiers, called ensemble classifiers—which combine classifiers through bagging, boosting, or stacking [49,162].

As such, I have tried several classifiers in this progression for IMS applications: namely, KNN, SVMs [55], and bagged decision trees [20]. Instead of delving into the implementations for each here, the intricacies of the algorithms are explained in Chapter VII. The main problem with these types of classifiers is that they do not account for the relationship *between* instances. The order of classes and observed features in a sequence is often as important as the feature values, especially in time series classification. HydroSense is no exception. One method of solving this is to add observed features from the instances before and after the current instance. Although it may seem simplistic, this method can yield surprisingly accurate results [37,78]. However, it adds further dimensions to your data, and can require more training examples to train properly [148].

Another, perhaps smarter, method of leveraging relationships in a sequence is to use dynamic sequence models (*e.g.,* dynamic graphical models), which explicitly model the relationship of classes and features in a sequence. However, instance classifiers still have their place. Many times sequence models can be less powerful than instance classifiers—this is where the concept of ensemble stacking comes into play. It is common practice to take the outputs of an instance classifier and use them as features in a sequence model [33].

Note also that I have not addressed the problem of semi-supervised classification. Even though ensemble classification is a powerful tool, one still needs labeled data to train the algorithms.

3.2 Sequence Models: Generative and Discriminative Graphical Models

Traditional sequence learners use labeled instances to train classifiers. An input sequence of observed variables, $\boldsymbol{x}$, is used to predict an observed output sequence, $\boldsymbol{y}$. For HydroSense, each pressure wave is a segmented event, y, which is generated from the opening or closing of a water valve. Many events compiled together form a sequence of T events, $\boldsymbol{y} = \{ y_1, y_2, \ldots y_T \}$. The most popular sequence models are dynamic graphical models. In these models, a "prologue" and a "chunk" are specified which defines

one or more hidden variables and the observed features. These variables are nothing more than parameterized probabilities. That is, the graphical structure tells us what the assumed model is, and one needs to find the parameters of each probability in the model that best describes our data. The "prologue" is always initialized to represent the first instances in a sequence. The "chunk" of probabilities is repeated over and over until it covers an entire sequence of data. We also need to specify the connections between each chunk as it is rolled out. This is achieved by looking at the connections to the prologue. The same connections between the prologue and chunk are repeated for the chunk-to-chunk connections. Figure 3-1 shows a simple graphical model, with the prologue and chunk specified. To train the parameters of the probabilities in a chunk and its interconnections, one can use a generative or discriminative model.

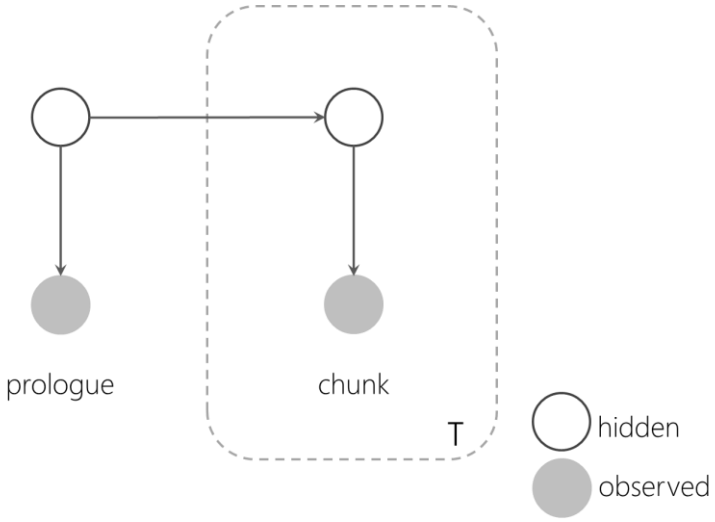

Figure 3-1. A linear Chain graphical model.

In a generative model, the idea is to model the joint probability of $\boldsymbol{x}$ and $\boldsymbol{y}$, parameterized by $\boldsymbol{\theta}$, $p(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta})$ for each variable in the graphical model. Then, when we observe $\boldsymbol{x}$, we use the parameterization to find the most likely sequence, $\boldsymbol{y}^*$. A simple example, a Gaussian mixture model (GMM), is often trained using maximum likelihood estimation [41] for each class label. The labels are used to group each class, y_a, and then mixtures of Gaussians are trained for each grouping to approximate $p_a(x,y=y_a;\boldsymbol{\theta_a})$ for a being each of T classes, $a=\{1,2,\ldots T\}$. Each model, p_a, can be parameterized only from observations of $\boldsymbol{x}$ when $\boldsymbol{y}=y_a$ (hard class assignment) or each model can be parameterized from all observations (soft assignment). Either way, if two classes have highly correlated features, their models would also have similar parameters, $\boldsymbol{\theta}_1 \approx \boldsymbol{\theta}_2$. With generative models, one can use the parameters to generate samples of x and y. This is not true for discriminative models.

The most common form of training models with hidden states is expectation maximization (EM) [11,72,126]. For a given graphical model and sequences of labeled data, one can compute the probability

of all the sequences under the current parameterization, and then update those parameters such that the probability of the training sequences is increased. With enough iteration, one can maximize the expectation of the parameterized model. The use of EM is a mature and developed research area and its use has been widely implemented in industrial settings. EM is the main method by which hidden Markov models (HMMs) are trained and has seen widespread and successful use in a number of domains, including speech recognition (perhaps its most popular use case).

In the case of HMMs, the total probability of a sequence is given by [72,126]:

$$p(\boldsymbol{y}, \boldsymbol{x}) = \prod_{t=1}^{T} \underbrace{p(y_t | y_{t-1}; A)}_{\text{transition}} \underbrace{p(x_t | y_t; \mu, \Sigma, \gamma)}_{\text{emission}} \tag{3.1}$$

where the emission probabilities are parameterized by a diagonal covariance Gaussian mixture (means, covariance's, and component weights—μ, Σ, γ) and the transition probabilities are parameterized via a conditional probability table, A of size $N \times N$, where N is the cardinality of hidden states.

In discriminative models, the idea is that one should always assume that **x** is observed and only model $p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$. It is argued that the probability, $p(\boldsymbol{x})$, is always marginalized out, so modeling $p(\boldsymbol{x},\boldsymbol{y};\boldsymbol{\theta})$ is largely a waste of computation if classification is the only task. Practically, the parameters are chosen to "discriminate" between y_1 and y_2 when x_1 and x_2 are completely observed and can be conditioned upon to maximally separate the classes. A theoretical analysis of the asymptotic performance for generative and discriminative classifiers was carried out by Liang and Jordan [92], showing that generative models are more sensitive to model misspecification. Asymptotic performance, however, is of little consequence when dealing with a minimal set of training labels. Practically, discriminative classifiers and generative models can be equally accurate in sequence labeling tasks, and neither method truly has an edge over the other [19,92,114,150,151]. In Chapter VII, I will demonstrate that discriminative classifiers and generative classifiers have similar performance in the context of HydroSense, but that discriminative methods tend to work better when training data is less available. In Chapter VIII, I will show they can be combined to build a semi-supervised model through iterative training.

One of the most popular discriminative classifiers is the linear chain conditional random field (CRF) proposed by Lafferty and McCallum [84]. A linear chain CRF is a graphical model similar to a hidden Markov model (HMM) but is trained discriminatively. For the remainder of this thesis, the term "CRF" refers to a "linear chain CRF," specifically. An example graphical representation of a linear chain CRF, with Markov order one, is given in Figure 3-2.

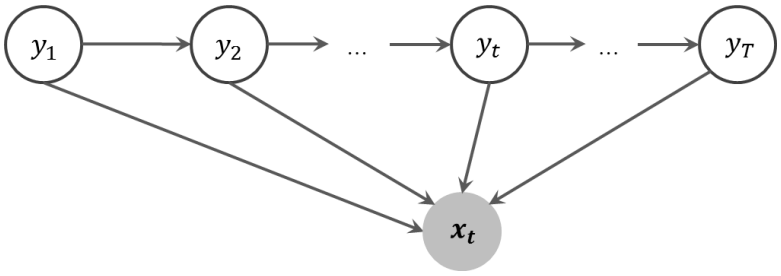

Figure 3-2. An example linear chain CRF

Because of the similarity, I will explain the conditional random field model using the Hidden Markov Models (HMM) as an introduction to the notation and concepts. I then focus on generative methods for training dynamic graphical models with EM.

3.2.1 From HMMs to CRFs

We can motivate the use of CRFs using the HMM as a prototype. This example follows somewhat closely with that of [140], but with extended derivations and unified with the notation of [51]. The classical definition of an HMM for a sequence of outputs, $\boldsymbol{y}$ and inputs, $\boldsymbol{x}$, of length T [126]:

$$p(\boldsymbol{y},\boldsymbol{x}) = \prod_{t=1}^{T} \underbrace{p(y_t|y_{t-1})}_{\text{transition}} \underbrace{p(x_t|y_t)}_{\text{emission}} \tag{3.2}$$

where I have defined $p(y_1)=p(y_1|y_0)$ to simplify the factorization. If I define the notation $\mathbf{1}_{\{y=i\}}$ to be an indicator function that returns "1" if $y=i$ and "0" otherwise, then I can rewrite this HMM [140] factorization as (if O represents the set of all observations and S is the set of all states):

$$p(\boldsymbol{y},\boldsymbol{x}) = \frac{1}{Z}\exp\left(\underbrace{\sum_{t=1}^{T}\sum_{i,j\in S}\theta_{ij}\cdot\mathbf{1}_{\{\boldsymbol{y_t}=i\}}\mathbf{1}_{\{\boldsymbol{y_t}=j\}}}_{\text{transition}} + \underbrace{\sum_{t=1}^{T}\sum_{o\epsilon O}\sum_{i\in S}\mu_{oi}\cdot\mathbf{1}_{\{\boldsymbol{y_t}=i\}}\mathbf{1}_{\{x_t=o\}}}_{\text{emission}}\right) \tag{3.3}$$

Such that θ_{ij} = log $p(y'=i|y=j)$ and μ_{oi} = log $p(x=o|y=j)$ and Z is a normalization so that everything sums to one. We can simplify this notation even further by defining "feature functions": So that $f_k(y_t,y_{t-1},x_t)$ exists for each transition $f_{ij}(y_t,y_{t-1},x_t)= \mathbf{1}_{\{y=i\}}\mathbf{1}_{\{y=j\}}$ and each observation/class pair $f_{io}(y_t,y_{t-1},x_t)= \mathbf{1}_{\{y=i\}}\mathbf{1}_{\{x=o\}}$. So that now we can write:

$$p(\boldsymbol{y},\boldsymbol{x}) = \frac{1}{Z}\exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K}\theta_k\cdot f_k(y_t,y_{t-1},x_t)\right) \tag{3.4}$$

where θ_k are the combined parameters from transmission and emission but appropriately matched to each feature function. The feature functions f_k are simply indicator functions that help simplify the notation.

With this in mind, I can now define the conditional probability by marginalizing out $p(\boldsymbol{x})$ with the sum over all possible $\boldsymbol{y}$.

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{p(\boldsymbol{y},\boldsymbol{x})}{\sum_{\boldsymbol{y}'} p(\boldsymbol{y}',\boldsymbol{x})} = \frac{\exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K}\theta_k \cdot f_k(y_t, y_{t-1}, x_t)\right)}{\sum_{\mathbf{y}'}\exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K}\theta_k \cdot f_k(y'_t, y'_{t-1}, x_t)\right)} \tag{3.5}$$

To keep this in line with the HMM, $f_k(y_t, y_{t-1}, x_t)$ must be an indicator function that codes the identity of the current class or observations (*i.e.*, $\mathbf{1}_{\{y=i\}}\mathbf{1}_{\{y=j\}}$ or $\mathbf{1}_{\{y=i\}}\mathbf{1}_{\{x=o\}}$), but these features can be extended so that other indicator function can be used such as $f_k(y_t, y_{t-1}, \boldsymbol{x}_t)$, where $\boldsymbol{x}_t$ can take knowledge from any portion of the sequence of observations. For example, adding a feature $f_k(y_t,y_{t-1},x_{t+1})=\mathbf{1}_{\{y=i\}}\mathbf{1}_{\{y=j\}}\mathbf{1}_{\{x=o\}}$ makes the transition from y_{t-1} to y_t depend on the observations in $t+1$. Note that this does not violate independence assumptions because we only assume independence among $\boldsymbol{y}$ not $\boldsymbol{x}$. Furthermore, even if I do violate independence assumptions (*i.e.*, using the next state y_{t+1} as a feature in x_t), discriminative models are less sensitive to violations of independence [111,140]. Lastly, I allow the feature function to be more than indicator functions, but instead be real valued scalar. It might return the value of a particular value in $\boldsymbol{x}_t$ or a sum of a combination of values, *etc*. This makes the interpretation of θ_k to be more in line with that of a "regression" parameter, but does not increase model complexity. In summary, the CRF can be thought of as a slightly more versatile model then the HMM, where one can add features from any observations into $\boldsymbol{x}_t$ and allow the feature functions to be more than indicators so that θ_k closely resembles a regression parameter.

We can now give the final definition of the linear chain CRF of length T.

$$p(\boldsymbol{y}|\boldsymbol{x}) = \frac{1}{\mathrm{Z}(\mathbf{x})}\exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K}\theta_k \cdot f_k(y_t, y_{t-1}, \boldsymbol{x}_t)\right) \tag{3.6}$$

where $\mathrm{Z}(\boldsymbol{x})$ normalizes the conditional by summing over all possible $\boldsymbol{y}$ of length T. I denote the enumeration of each possible sequence using the notation $\boldsymbol{y}=\langle y_1{:}y_T\rangle$.

$$Z(\boldsymbol{x}) = \sum_{\mathbf{y}'=\langle \mathrm{y}_1:\mathrm{y}_T\rangle}\exp\left(\sum_{t=1}^{T}\sum_{k=1}^{K}\theta_k \cdot f_k(y'_t, y'_{t-1}, \boldsymbol{x}_t)\right) \tag{3.7}$$

Often it is the case that one does not want to lump all of the observations and variables into a single sequence of length T. Instead, one wishes to break apart the linear chain into N non-overlapping sequences, $\boldsymbol{y}^{(i)}$ and $\boldsymbol{x}^{(i)}$, and the length of the i^{th} sequence is T_i. If one assumes each sequence and the observations in each sequence are iid. (that is, independent and identically distributed), one can write:

$$p(\boldsymbol{y}|\boldsymbol{x}) = \prod_{i}^{N} p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}) \tag{3.8}$$

and

$$p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}) = \frac{1}{\mathrm{Z}(\boldsymbol{x}^{(i)})} \exp\left(\sum_{t=1}^{T_i} \sum_{k=1}^{K} \theta_k \cdot f_k(y_t^{(i)}, y_{t-1}^{(i)}, \boldsymbol{x}_t^{(i)}) \right) \tag{3.9}$$

This is an often imposed and reasonable assumption for most time series classifiers. In HydroSense, for example, it is natural to assume independence in water usage from day-to-day or week-to-week. We can use another simplification of the notation to make it more compact by defining the column vectors $\boldsymbol{\theta}^{\mathrm{T}} = \{ \theta_1, \ldots \theta_K \}$ and $\boldsymbol{f}^{\mathrm{T}} = \{ f_1, \ldots f_K \}$ for the K feature functions. Now one has:

$$p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}) = \frac{1}{\mathrm{Z}(\boldsymbol{x}^{(i)})} \exp\left(\sum_{t=1}^{T_i} \boldsymbol{\theta}^{\mathrm{T}} \cdot \boldsymbol{f}(y_t^{(i)}, y_{t-1}^{(i)}, \boldsymbol{x}_t^{(i)}) \right) \tag{3.10}$$

where one only needs to estimate the parameters in $\boldsymbol{\theta}$. In this thesis, the main aim of training the CRF is so that one can infer the sequence with maximum probability given a sequence of observations, $\boldsymbol{y}^{*} = \mathrm{argmax}_{\boldsymbol{y}}\, p(\boldsymbol{y}|\boldsymbol{x};\boldsymbol{\theta})$. For HydroSense, this is the "most likely valve sequence," given a trained CRF model and a sequence of observations. I will eventually perform this in a semi-supervised manner. First, however, it is beneficial to discuss traditional parameter estimation using labeled data.

3.2.2 Traditional CRF training

In traditional CRF training there are sets of labeled sequences, $\boldsymbol{y}$, and observations for each label, $\boldsymbol{x}$. The aim is to estimate the parameters, $\boldsymbol{\theta}$, such that they maximize the log-likelihood of the entire model, log $p(\boldsymbol{y}|\boldsymbol{x})$. From Equation (3.8) one can define the log likelihood as:

$$\log p(\boldsymbol{y}|\boldsymbol{x}) = \log \prod_{i=1}^{N} p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}) = \sum_{i=1}^{N} \log p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}) \tag{3.11}$$

Substituting the CRF definition from Equation (3.10), the log likelihood simplifies to

$$\log p(\boldsymbol{y}|\boldsymbol{x}) = l(\boldsymbol{\theta}) = \sum_{i=1}^{N} \left(\left[\sum_{t=1}^{T_i} \boldsymbol{\theta}^{\mathrm{T}} \cdot \boldsymbol{f}(y_t^{(i)}, y_{t-1}^{(i)}, \boldsymbol{x}_t^{(i)}) \right] - \log \mathrm{Z}(\boldsymbol{x}^{(i)}) \right) \tag{3.12}$$

We can also add a regularization term to mitigate the effects of over fitting. I leave the term absent here because its addition is trivial to incorporate. Equations of the form log(Σexp(x)) can be shown to be convex [140] so a gradient based optimization can be highly efficient and guaranteed to find the global

optimum. I note that there might be some saddle points in the optimization, but a global optimum can be guaranteed when a regularization term is added (*i.e.*, making the space strictly convex).

We can calculate a gradient of:

$$\frac{\partial}{\partial\theta_k} l(\boldsymbol{\theta}) = \sum_{i=1}^{N} \left(\left[\sum_{t=1}^{T_i} f_k(y_t^{(i)}, y_{t-1}^{(i)}, \boldsymbol{x}_t^{(i)}) \right] - \frac{\partial}{\partial\theta_k} \log \mathrm{Z}(\boldsymbol{x}^{(i)}) \right) \tag{3.13}$$

$\frac{\partial}{\partial\theta_k} \log \mathrm{Z}(\boldsymbol{x}^{(i)})$ is more involved to calculate but it is still straight forward. The full derivation is given at the end of this chapter for the interested reader, lemma 3.1. The final result is:

$$\frac{\partial}{\partial\theta_k} l(\boldsymbol{\theta}) = \sum_{i=1}^{N} \left(\left[\sum_{t=1}^{T_i} f_k(y_t^{(i)}, y_{t-1}^{(i)}, \boldsymbol{x}_t^{(i)}) \right] - \sum_{t=1}^{T_i} \sum_{\boldsymbol{y}=<y,y'>} p(y, y' | \boldsymbol{x}_t^{(i)}) f_k(y, y', \boldsymbol{x}_t^{(i)}) \right) \tag{3.14}$$

Where $\boldsymbol{y}=<y,y'>$ enumerates all possible label pairs. The bracketed term is easily computed as it is just the k^{th} feature function. The second term is more difficult because it requires dynamic programming to compute $p(y, y' | \boldsymbol{x}_t^{(i)})$. Still, the probability can be efficiently calculated using forward/backward and has complexity $O(NTM^2)$ for each gradient computation; where M is the number of possible labels in the state space. This is because N sequences must be separately calculated using forward/backward, a complexity of $O(T_i M^2)$ for each sequence. I have abused the notation NT to refer to the sum of the sequence lengths, $\sum_{i=1}^{N} T_i$.

Notice also that the partial derivative of θ_k requires values for all sequences and all values of $\boldsymbol{\theta}$, so no closed form solution exists. We must use numerical optimization to update the parameters in batch mode; for instance, using gradient ascent, Levenberg Marquardt, or L-BFGS [101,112,138,156]. However, there are approximations that use stochastic gradient for "online" parameter estimation which have been shown to converge faster than the batch mode without loss in accuracy [154]. From the perspective of HydroSense, this could be useful for updating an already trained model, except appropriately modified to use a semi-supervised approach (Chapter VIII).

The result of Equation (3.14) is a common result in computing the gradient of exponential families [155]. The bracketed term is the expected value of the k^{th} feature using sequences $\boldsymbol{x}$ and $\boldsymbol{y}$, and the second term is the expected value of the k^{th} feature under the model distribution. When these two quantities are equal, the gradient is zero. If one defines each of these expected values explicitly Equation (3.14) can be rewritten as:

$$\frac{\partial}{\partial \theta_k} l(\boldsymbol{\theta}) = \sum_{i=1}^{N} \left(\mathrm{E}[f_k(\boldsymbol{y}^{(i)}, \boldsymbol{x}^{(i)})] - \mathrm{E}_{p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)})}[f_k(\boldsymbol{y}^{(i)}, \boldsymbol{x}^{(i)})] \right) \tag{3.15}$$

where

$$\mathrm{E}[f_k(\boldsymbol{y}^{(i)}, \boldsymbol{x}^{(i)})] = \sum_{t=1}^{T_i} f_k(y_t^{(i)}, y_{t-1}^{(i)}, \boldsymbol{x}_t^{(i)}) \tag{3.16}$$

is the expected value of the feature for the instances $\boldsymbol{y}^{(i)}, \boldsymbol{x}^{(i)}$ and,

$$\mathrm{E}_{p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)})}[f_k(\boldsymbol{y}^{(i)}, \boldsymbol{x}^{(i)})] = \sum_{t=1}^{T_i} \sum_{\boldsymbol{y}=<y,y'>} p(y, y'|\boldsymbol{x}_t^{(i)}) f_k(y, y', \boldsymbol{x}_t^{(i)}) \tag{3.17}$$

is the expected value of the features under the model $p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)})$ that is parameterized by $\boldsymbol{\theta}$, again not normalized to sum to one. This result can be generalized when one applies a semi-supervised training algorithm.

To explicitly define the dependence of $p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)})$ on $\boldsymbol{\theta}$ and to simplify notation, I will abbreviate $\mathrm{E}_{p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)})}[f_k(\boldsymbol{y}^{(i)}, \boldsymbol{x}^{(i)})]$ as $\mathrm{E}_{\boldsymbol{\theta}}[f_k(\boldsymbol{y}^{(i)}, \boldsymbol{x}^{(i)})]$. Furthermore, I will define the expectation of the column vector, $\boldsymbol{f}$, as the expected value of each individual feature function,

$$\mathbf{E}_{\boldsymbol{\theta}}[\boldsymbol{f}(\boldsymbol{y}^{(i)}, \boldsymbol{x}^{(i)})]^{\mathrm{T}} = \{\mathrm{E}_{\boldsymbol{\theta}}[f_1(\boldsymbol{y}^{(i)}, \boldsymbol{x}^{(i)})], \ldots \mathrm{E}_{\boldsymbol{\theta}}[f_K(\boldsymbol{y}^{(i)}, \boldsymbol{x}^{(i)})]\} \tag{3.18}$$

Using this notation, I can now define the entire gradient using matrix notation,

$$\nabla_{\boldsymbol{\theta}} l(\boldsymbol{\theta}) = \sum_{i=1}^{N} (\mathbf{E}[\boldsymbol{f}(\boldsymbol{y}^{(i)}, \boldsymbol{x}^{(i)})] - \mathbf{E}_{\boldsymbol{\theta}}[\boldsymbol{f}(\boldsymbol{y}^{(i)}, \boldsymbol{x}^{(i)})]) \tag{3.19}$$

such that the entire gradient vector is the sum of gradients of each of the *N* sequences.

Note that when using traditional parameter estimation, the training data must contain labels. This type of training performs well in HydroSense (Chapter VII), but the amount of training data needed limits the utility of the estimation. Now that I have a good basis for updating parameters of a discriminative time series classifier, the CRF, I will review the traditional approaches to unsupervised and semi-supervised machine learning.

3.3 Unsupervised learning: clustering and feature discovery

True unsupervised learning only uses only unlabeled data. In its purest definition, this means no prior knowledge or labeled subsets are used. Often this type of problem is intractably difficult. The most common unsupervised approaches use clustering of the features and assume that each cluster represents a class label. Clustering can be performed using EM and initialized via k-means, a classical, but popular method. Alternatively, stochastic sampling methods can be used (for instance, using Markov Chain Monte-Carlo), or approaches such as Latent Dirichlet Allocation [14] which is popularly used in document clustering with a bag of words model. In this approach, a "prior" is assumed upon the distribution of the parameters of the latent states.

More often, unsupervised learning is used to discover structure in a dataset. For example, using different clustering mechanisms to find semantically meaningful features over different clusters [110]. Classification using unsupervised learning has seen limited success. Instead, we almost always know something useful about the problem that allows semi-supervised learning or the inclusion of domain knowledge.

3.4 Semi-Supervised learning

There are numerous semi-supervised learning approaches that exploit some knowledge of how instances should cluster or exploit some initial labeling of a subset of the instances. The approaches can broadly be categorized into four classes based on the assumptions they make about the unlabeled instances [51]: (1) cluster assumption-based classifiers, (2) sparsity assumption-based classifiers, (3) manifold-based classifiers, and (4) multi-view classifiers. I also introduce the concept of generalized expectation because it is a form of semi-supervised training used explicitly for training CRFs. Ultimately I decided not to employ this method because it requires an expert's domain knowledge of the problem for providing expectations on feature-label distributions. My aim is to eventually provide a semi-supervised training toolkit where no expert knowledge is required.

3.4.1 Cluster Assumption Classifiers

The cluster assumption states that output variables of a given label are highly correlated. This means that clustering directly on the features should also cluster the classes together. Methods that use this assumption include bootstrapping methods and expectation maximization (EM) [45]. Bootstrapping uses a subset of labeled data to train a classifier and then extends the training using unlabeled data. Unlabeled instances are iteratively added and the model is re-trained. Even so, bootstrapping typically results in a large loss in performance unless there are a number of human interventions in the training process [128].

In EM, it is typical to first train the EM classes from labeled data, then weight the log-likelihood from the unlabeled instances less when calculating the log-likelihood [116]. EM methods in semi-supervised learning can be extremely powerful when properly initialized [81]. However, the correct initialization is many times difficult or impossible to know [54,108]. Also, these methods suffer from the same limitations of generative models—when features are not independent, the classes can be easily confused [111,140].

3.4.2 Sparse Decision Boundary Classifiers

The sparsity assumption states that decision boundaries between classes should lie in low entropy areas. Intuitively this makes sense. Low entropy regions (if the classes are somewhat separable) will correspond to optimal margins from the classes. A common example is entropy regularization [62], which composites the objective function calculation with a measure of scaled entropy on the classifier:

$$Obj = p(\boldsymbol{y}_{labeled}|\boldsymbol{x};\boldsymbol{\theta}) - \lambda\mathcal{H}\left(p(\boldsymbol{y}_{unlabeled}|\boldsymbol{x};\boldsymbol{\theta})\right) \quad (3.20)$$

where $\boldsymbol{y}_{labeled}$ and $\boldsymbol{y}_{unlabeled}$ are the different instances and λ is a tuned parameter. The tuning of λ drastically affects performance [73].

Transductive Support Vector Machines (TSVMs) find the optimal decision boundaries on labeled data, but add a constraint to the optimization that margins are preserved on the unlabeled portions of the data. That is, boundaries lie in regions of low entropy [75]. The training process sums over all possible labeling, however, so that a brute force approach is intractable and the approximation to reduce runtime is $O(n^3)$. As such, using TSVMs in a HydroSense framework places a significant burden on the initial calibration, and requires a number of labeled instances to bootstrap the classifier.

Even so, linear programming can be used in conjunction with the constraint that certain classes will show up in the data a certain amount of time [75,139]. In this framework, each class used in a TSVM is assigned an empirical value for the number of times it is expected to occur. The constraint in the optimization is given such that the current class must occur at least X percent of the time, where X is provided by prior knowledge or by assigning it based on the occurrences in a set of labeled data [139]. I use TSVMs as baseline classifiers in the HydroSense system, but show that they are ultimately not a good method, and generalize poorly with the dataset.

3.4.3 Manifold Classifiers

Manifold classifiers assume that a low dimensional manifold exists on which labels can be efficiently separated. For example, a geodesic structure inside a three dimensional features space. On the surface of this manifold, which is a non-linear mapping of the data, clustering is much more straight-forward, in

theory. Practically, structure of the manifold is usually difficult to find. However, graph based methods have been shown to perform well. A graph is created on the labeled data and another on the unlabeled data. Neighborhoods are then assigned from the labeled and unlabeled graphs that map the instance labels [168]. These methods also require that all of the training and test data be stored along with all the graph and edge weights. As such, these classifiers may not be appropriate for a cloud computing architecture, especially one that needs to make updates as new data arrives—the model must be completely retrained when new data arrives.

3.4.4 Multi-view Classifiers

Multi-view classifiers train an ensemble of classifiers that are complementary to one another. When combined, the ensemble is, ideally, able to combine the different hypotheses. Co-training, for example, trains different classifiers using the labeled data as a seed for each classifier. Unlabeled instances are iteratively added and the models are re-trained [15,164]. However, the number of labeled instances grows as the number of ensemble classifiers grows; meaning a number of labeled instances is ultimately needed.

Co-training has been shown to be a powerful tool in many different applications [15,48,106,166]. However, when an incorrect label is added, it can degrade the performance of the system [144]. Co-training is also considered a method of probably approximate learning (PAC learning). In these problems, it is assumed that a set of classifiers can be built that each captures different aspects of the problem. By combining their estimates, one can be more certain about the labels then add from the pool of unlabeled instances. Different views of the problem are typically created by grouping the features training different classifiers with the different groupings. If a natural split of the features does not exist, however, then Co-training is not guaranteed to increase performance. Moreover, one typically cannot guarantee that the features capture "independent" aspects of the problem, even when a natural split of the features exists. However, even in these scenarios where dependency exists between the classifiers, co-training can positively affect the classification performance [115]. I investigate the use of co-training for HydroSense, but show that it ultimately does not result in good generalization. It does have other advantages, however, such as the ability to classify selected classes with high accuracy. These advantages are explored and evaluated in Chapter VIII.

3.4.5 Conditional Random Fields with Generalized Expectation Criteria

There are also approaches, similar to traditional semi-supervised approaches, which use prior knowledge about the problem to guide the training of different classifiers. These types of classifiers came to be known as "weakly" labeled or "lightly" supervised approaches, and differ from semi-supervised approaches in that they do not require a subset of labels to prime the classification process [24,169]. Note that this is different than the notion of "weak labels" commonly used in relational learning where a human

creates deterministic rule sets for predicting a relation between two entities. In the context of HydroSense, these weak labels are stochastic and applied to the expected feature/label relationship. That is, scores are calculated that measure the relative "closeness" between (1) how we expect features of the model to be distributed and (2) how they actually are distributed once we have a parameterization of the model.

Unlike completely unsupervised approaches, however, they do require some type of domain knowledge of the problem. The body of work in this space remained largely application specific and required considerable parameter tuning to achieve "good" results. This was until the process was formalized by Mann and McCallum in 2010, and was extended to CRFs by the thesis of Druck in 2011 [50,51,102]. In these works, the concept of *generalized expectation* was introduced. The use of GE, however, is not without its downsides. The features for a given classification problem must be known beforehand and an expert must label the expected distributions of the features (sometimes the relationships are not intuitive and require a long process of labeling). Even so, the labeling time is often much less than labeling each instance. The last disadvantage of GE is that the training space is non-convex and the gradient is incredibly complex to calculate. As such, GE has seen limited adoption because of its practical downsides in training time compared to other approaches.

3.4.6 Virtual Evidence in Dynamic Graphical Models

Virtual evidence is a process for injecting knowledge about a particular problem by an expert. The use of virtual evidence was first proposed by Pearl [125] for Bayesian networks in the 1980s. It has, however, seen use in more recent applications of Dynamic Bayesian Networks (DBNs) [11,12] and in CRFs [91,93,99]. The basic concept is simple. We wish to bias the probability estimate of certain class assignment over other class assignments in a network, given some prior knowledge. This information is then coded into a conditional probability, called the virtual evidence.

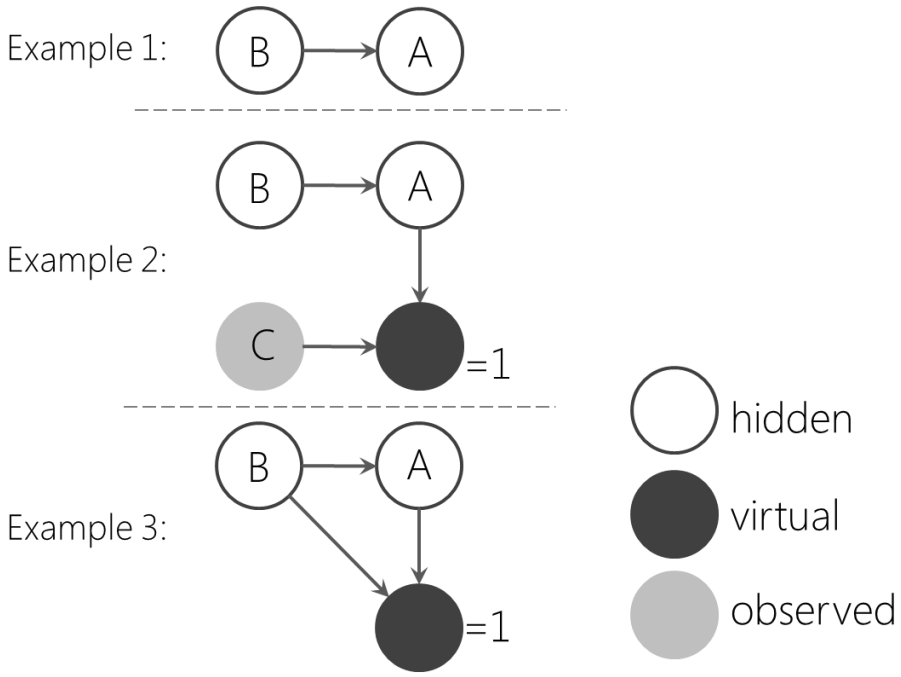

Figure 3-3. Three conditional probability examples

Figure 3-3 illustrates how virtual evidence can be used in a simple Bayesian network. Technically, the virtual evidence node is also just an observed value, but I give it a separate color to highlight it from the other hidden and observed variables. The two hidden state nodes in example 1 define the conditional probability $P(A|B)$. If we are trying to find the most probable state assignment of the hidden variables *A* and *B*, we typically will look at the maximum of entire sequence which is given by $P(B)P(A|B)$. If both *A* and *B* have a cardinality of two and are equally likely, then the conditional probability between them uniquely determines the most probable state assignment. The maximum transition probability will determine the values of *A* and *B*. Now consider the addition of the virtual evidence node. The probability of the node, $P(VE)$, is always equal to 1, but now the total probability of the sequence becomes $P(B)P(A|B)P(A)P(VE|A,C)$. In performing the assignment of *A* and *B* we must now also consider the conditional probability between *VE* and the variable *A*. The conditional probability can take on any prior knowledge we wish, from the observed variable *C*, for preferring different values of *A*. Perhaps we know that *A*=0 is twice as likely as *A*=1 for some observation *C*=*W*. In that case we define $P(VE|A=0,C=W)=1$ and $P(VE|A=1,C=W)=0.5$. Effectively we have now injected our prior knowledge into the Viterbi state assignment process.

However, we do not even need the variable *C*. Consider the next example at the bottom of the figure. We can give preference to certain transitions between *A* and *B*. For instance if we think that the transition between *A*=1 and *B*=1 is ten times more likely than any other transition, then we can achieve this through

assigning $P(VE|A=1,B=1)=1$ and $P(VE|A,B)=0.1$ otherwise. When we run decoding, these probabilities will be message passed into the state space, affecting the selection of *A* and *B*. The intuition here is that the transition between *A* and *B* is not directly determined by the trained transition probabilities, but is also affected by our prior knowledge. The actual values of the conditional probability do not matter, only the relative value of the probabilities. For instance we could have conveyed the same10:1 preference using the values 0.5 and 0.05. The Viterbi assignment does not change.

While simple, this can be a powerful tool because it means we rely less on the training data to find the statistical relationship between hidden variables. It is a means for giving a preference to different decoding through the use of our prior knowledge. When we combine virtual evidence with an observed parent node (such as parent *C* in Figure 3-3) we can form complex relationships between our hidden variables and the value of *C*. For instance, *C* could come from a completely different classifier and denote its belief in the value of *A*. I explore the use of virtual evidence in HydroSense later on, showing it to be a powerful means of combining different knowledge sources, including other semi-supervised classifiers.

3.5 Active Learning

Whereas semi-supervised learning uses prior information and unlabeled data to exploit hypotheses it has about the dataset, active learning uses the unlabeled data to explore the dataset [152]. In particular this usually involves the existence of an oracle which can provide labels to the system, known more specifically as active labeling. However, there is always a cost to querying the oracle. One uses the unlabeled data to decide if a query may or may not be useful. Different strategies exist for how best to decide on what labels to query from the oracle, and this is a typical means of categorizing different query strategies. Two of the most pervasive strategies are uncertainty sampling and query by-committee [137]. One can also categorize the systems by the scenario in which they ask the oracle for labels. For example, we may have a large pool of unlabeled examples and need to decide which labels in the pool are most informative (known as pool-based sampling [90]). Or, the system might receive streaming examples of the data and decides whether to ask for a label for that particular new instance (known as stream based selective sampling [4,34]). I focus my discussion to the stream based strategy because it is most pertinent to the HydroSense system. In our application, the homeowner is the "oracle" and we need to decide quickly whether or not to ask the homeowner for a label to water usage that just occurred. This ensures that (a) the water use occurs in a natural setting and, because we immediately decide whether or not to ask for a label, (b) the user of the water can provide as much information about the water usage as possible (*i.e.*, temperature setting of the valve). I discuss the assumptions of active labeling more in Chapter VIII. I turn attention now to distinguishing uncertainty sampling and query by committee.

3.5.1 Uncertainty Sampling

The basic concept in uncertainty sampling is to use the confidence of a particular classifier to decide if a label should be queried. Low confidence examples are preferred. However, various strategies can be used. Lewis and Gale [90] proposed sampling any instance whose posterior probability was lower than the inverse of the number of classes. It has also been proposed to use the absolute least confident example in an unlabeled pool [89]. The first method presumes that a threshold can be set, which is more appropriate for streaming selective sampling. The second presumes an unlabeled dataset exists, more appropriate for pool based sampling. This technique has been popular, but has some downsides; ultimately, the decision is made based upon the probability of a *single* class, rather than all the classes.

Another strategy is called margin sampling which selects the instance with the smallest margin between the most probable and next most probable classes [132]. Because this strategy involves more than one posterior, it can achieve better results than simply choosing the minimum confidence. However, in large class problems, it may not be any more advantageous. Finally, there are more information theoretic approaches which use the entropy across classes for a given instance. That is, the sum of the entropy for each class probability, for a given instance [70,136]. The approach chooses the instance with the lowest entropy, meaning the assignment of classes is most uncertain. This approach naturally lends itself to problems with larger state spaces.

The different approaches for active learning have been empirically evaluated in a number of studies [82,133,135]. In general the best algorithm is application dependent. Many approaches have shown all methods to be better than randomly selecting classes to query. These studies also were able to show that using the margin for selecting classes typically worked well for classification tasks, many times better than confidence and entropy methods (although not necessarily consistently).

3.5.2 Query by Committee

Query by committee is highly related to the process of ensemble learning and co-training discussed earlier [105,113,145,149,165,167,169]. One constructs a committee of classifiers, each with a different "view" of the data (*i.e.,* a different set of labels or features for each classifier). We wish to query, then, the instances where there is the most disagreement among the classifiers. This disagreement can be measured using Kullback-Liebler divergence between each classifier in the committee and the distribution of the entire ensemble. Choosing the instance with the largest KL-divergence, intuitively, is selecting the instance whose individual classifiers vary the most from the mean [105].

Other approaches use the "vote entropy," which takes into account the number of votes from each classifier to each class. The entropy of the distribution is the entropy of the class label distribution for the

ensemble [39]. This approach can also be considered a hybrid between uncertainty sampling and query by committee. The reason for this is that many ensemble classifiers attain class confidences through class label distributions (*i.e.,* [20]). Therefore the entropy of the committee is the same as the entropy of the classifier as a whole, making uncertainty sampling equivalent to query by committee when using these ensemble classifiers.

For HydroSense, I employ a hybrid approach of ensemble classifiers, multi-view committee, and margin-based uncertainty sampling. This method is outlined in Chapter VIII.

3.6 Summary

In summary, I presented the related work and methodology for training of dynamic graphical models such as conditional random fields and hidden Markov models. In particular, I reviewed expectation maximization and the supervised training of CRFs through gradient methods. I then introduced the concept of semi-supervised learning and active learning and their possible utility in a system like HydroSense. In particular, I have laid the foundation for methods that I will leverage later on: multi-view classifiers, classifiers that leverage virtual evidence, and stream based active learning (or selective sampling).

3.7 Lemmas

Lemma 3.1: Derivation of $\frac{\partial}{\partial \theta_k} \log \mathrm{Z}(\boldsymbol{x}^{(i)})$

$$\frac{\partial}{\partial \theta_k} \log \mathrm{Z}\left(\boldsymbol{x}^{(i)}\right) = \frac{1}{\mathrm{Z}(\boldsymbol{x}^{(i)})} \frac{\partial}{\partial \theta_k} \mathrm{Z}\left(\boldsymbol{x}^{(i)}\right)$$

$$= \frac{1}{\mathrm{Z}(\boldsymbol{x}^{(i)})} \frac{\partial}{\partial \theta_k} \sum_{\mathbf{y}'=<\mathrm{y}_1:\mathrm{y}_{T_i}>} \exp\left(\sum_{t=1}^{T_i} \sum_{k=1}^{K} \theta_k \cdot f_k(y'_t, y'_{t-1}, \boldsymbol{x}_t^{(i)}) \right)$$

$$= \frac{1}{\mathrm{Z}(\boldsymbol{x}^{(i)})} \sum_{\mathbf{y}'=<\mathrm{y}_1:\mathrm{y}_{T_i}>} \exp\left(\sum_{t=1}^{T_i} \sum_{k=1}^{K} \theta_k \cdot f_k\left(y'_t, y'_{t-1}, \boldsymbol{x}_t^{(i)}\right) \right) \cdot$$

$$\frac{\partial}{\partial \theta_k} \sum_{t\prime=1}^{T} \sum_{k\prime=1}^{K} \theta_{k\prime} \cdot f_{k\prime}(y'_{t\prime}, y'_{t\prime-1}, \boldsymbol{x}_{t\prime}^{(i)})$$

$$= \frac{1}{\mathrm{Z}(\boldsymbol{x}^{(i)})} \sum_{\mathbf{y}'=<\mathrm{y}_1:\mathrm{y}_{T_i}>} \exp\left(\sum_{t=1}^{T_i}\sum_{k=1}^{K} \theta_k \cdot f_k\left(y'_t, y'_{t-1}, \boldsymbol{x}_t^{(i)}\right)\right) \sum_{t\prime=1}^{T_i} f_k(y'_{t\prime}, y'_{t\prime-1}, \boldsymbol{x}_{t\prime})$$

$$= \sum_{\mathbf{y}'=<\mathrm{y}_1:\mathrm{y}_{T_i}>} \underbrace{\frac{1}{\mathrm{Z}(\boldsymbol{x}^{(i)})} \exp\left(\sum_{t=1}^{T_i}\sum_{k=1}^{K} \theta_k \cdot f_k\left(y'_t, y'_{t-1}, \boldsymbol{x}_t^{(i)}\right)\right)}_{\text{probability of sequence } <\mathrm{y}_1:\mathrm{y}_{T_i}>} \underbrace{\sum_{t\prime=1}^{T_i} f_k(y'_{t\prime}, y'_{t\prime-1}, \boldsymbol{x}_{t\prime}^{(i)})}_{\text{Sum of the constraints for } <\mathrm{y}_1:\mathrm{y}_{T_i}>}$$

$$= \sum_{\mathbf{y}'=<\mathrm{y}_1:\mathrm{y}_{T_i}>} p(<\mathrm{y}_1:\mathrm{y}_{T_i}> | \boldsymbol{x}^{(i)}) \sum_{t\prime=1}^{T_i} f_k(y'_{t\prime}, y'_{t\prime-1}, \boldsymbol{x}_{t\prime}^{(i)})$$

The Markov (order one) property of the linear chain means I can rewrite this as:

$$= \sum_{\mathbf{y}'=<\mathrm{y}_1:\mathrm{y}_{T_i}>} \sum_{t\prime\prime=1}^{T_i} p(y'_{t\prime\prime}, y'_{t\prime\prime-1} | \boldsymbol{x}_{t\prime\prime}^{(i)}) \sum_{t\prime=1}^{T_i} f_k(y'_{t\prime}, y'_{t\prime-1}, \boldsymbol{x}_{t\prime}^{(i)})$$

$$= \sum_{\mathbf{y}'=<\mathrm{y}_1:\mathrm{y}_{T_i}>} \sum_{t\prime\prime=1}^{T_i} \sum_{t\prime=1}^{T_i} p(y'_{t\prime\prime}, y'_{t\prime\prime-1} | \boldsymbol{x}_{t\prime\prime}^{(i)})\, f_k(y'_{t\prime}, y'_{t\prime-1}, \boldsymbol{x}_{t\prime}^{(i)})$$

Which can be further simplified because the constraint feature is zero for all the marginal probabilities except when *t'=t''*.

$$= \sum_{\mathbf{y}'=<\mathrm{y}_1:\mathrm{y}_{T_i}>} \sum_{t=1}^{T_i} p(y'_t, y'_{t-1} | \boldsymbol{x}_t)\, f_k(y'_t, y'_{t-1}, \boldsymbol{x}_t^{(i)}) \tag{3.21}$$

This sum enumerates all of the possible pair wise combinations in the sequence $<y_1:y_T>$ and does so T_i number of times for each input $\boldsymbol{x}_t$. Thus, this can be made more explicit with the notation:

$$= \sum_{t=1}^{T_i} \sum_{\boldsymbol{y}\prime=<y,y'>} p(y, y' | \boldsymbol{x}_t^{(i)}) f_k(y, y', \boldsymbol{x}_t^{(i)}) \tag{3.22}$$

which is exactly what I arrived at in Equation (3.14).

CHAPTER IV

4. Chapter Four:
Theory of Operation and Initial Feasibility

This chapter summarizes theory of operation of the HydroSense system. The theory of operation uses a transmission line equivalent system to mirror the physical properties of water in pipes. I then present the results of an initial feasibility study verifying that the pressure waves generated in a plumbing system are unique. This chapter is geared more towards water sustainability engineers, but computer scientists should find most topics highly approachable and interesting.

4.1 Theory of Operation: A Plumbing Primer

In this section, I provide background on residential water supply systems and in-home plumbing. I also introduce the basic theory of operation that motivates my approaches to feature extraction. Households obtain water from one of two sources: a public water supply or a private well. Public water is distributed by local utilities, relying on gravity and pumping stations to push water through major distribution pipes. Residences are connected to a water main by a smaller service line, where the water meter is typically found. A backflow valve accompanying the water meter prevents household water from flowing back into the main. In contrast, homes with private wells use a pump to draw the water out of the ground and into a small tank within the home. A pressured bladder pushes water from the tank when a valve in the home is opened, where it is stored under pressure. Private wells are typically unmetered.

Figure 4-1 depicts a typical in-home plumbing system. Cold water enters through the main service line, typically at 50-100 pounds per square inch (psi) depending on such factors as the elevation and proximity to a reservoir or pumping station. Many homes have a pressure regulator that protects the home from transients (or pressure spikes) from the main and also reduces the incoming water pressure to a level safe for household fixtures. The regulator acts as a buffer between the water main and the home and typically prevents “cross talk” on the lines. That is, pressure fluctuation on one side of the regulator does

not typically affect pressure on the opposite side. The regulator maintains a constant pressure in the home. As long as water is not flowing inside the home, the regulator can keep the pressure static.

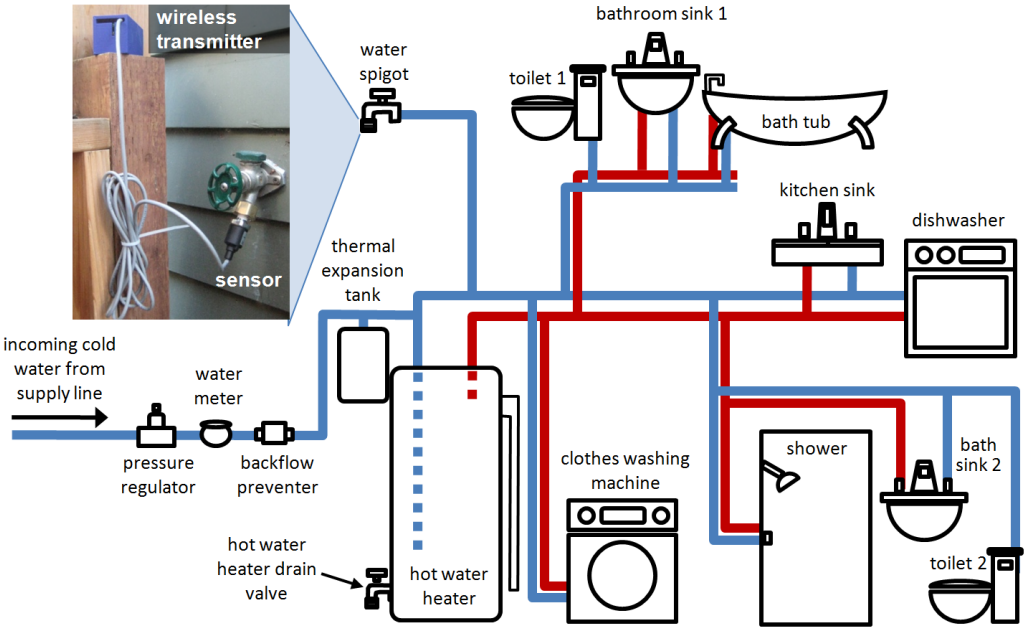

Figure 4-1. An illustrative schematic of a basic plumbing layout in a two-bathroom home. HydroSense can be easily installed at any accessible location in a home's water infrastructure, with typical installations at an exterior hose bib (shown above), a utility sink spigot, or a water heater drain valve.

After the regulator, there are two basic layouts found in typical residential piping, series plumbed and branched. Almost all multi-fixture homes have a combination. The size of the plumbing also varies, usually in one of three sizes based on the expected flow rate along the pipe. These are referred to as trunk, branch, and twig lines, in order of highest to lowest expected flow. The cold water trunk supply branches to the individual water fixtures (*e.g.*, toilets, sinks, and showers) and into the water heater. A traditional water heater heats water in an insulated tank using electric coils or gas. When hot water is used, the pressure from the cold trunk pushes hot water out of the tank and refills it with cold water. The cold water feed line stretches down to the bottom of the tank. Cold water remains at the bottom until it is heated (as cold water is denser than hot water) and the hot water line draws from the top of the tank. Every hot water tank has a pressure relief valve and a drain valve, which is important for maintenance as water heaters should be drained at least once a year to flush mineral deposits and increase operating efficiency. Many homes also have a thermal expansion tank connected to the water heater, providing space to store excess water as it expands during heating. Thermal expansion from the water tank can increase the pressure in the home. For example, over a single night the water heater heats a large volume of water without any outlet of the pressure, which can slowly increase the pressure in the home. This phenomenon is shown in Figure 4-2.

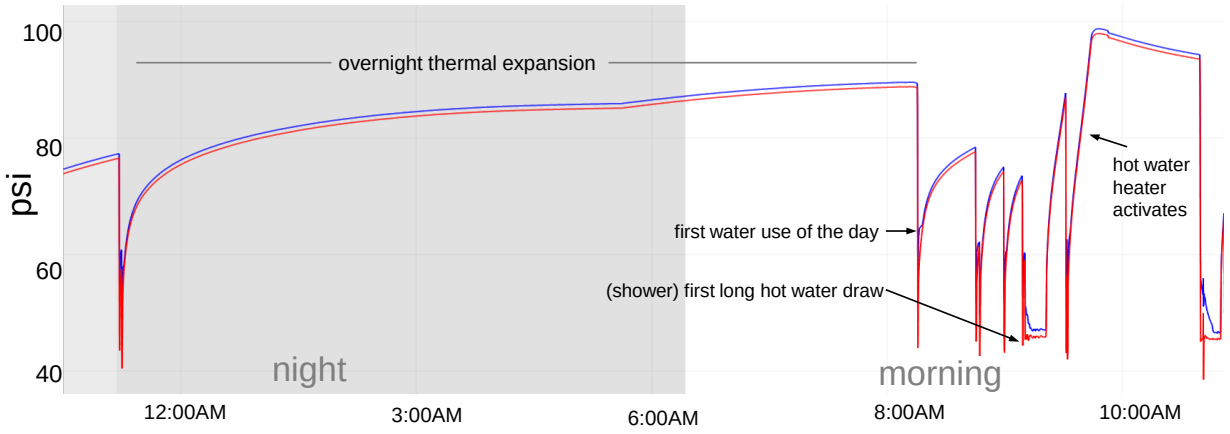

Figure 4-2. Example of thermal expansion in H2. Blue line represents pressure on the cold plumbing line. Red represents pressure on the hot water plumbing line.

Some homes instead use tankless heaters, which provide hot water on demand by circulating it through burners or electric coils. Both approaches connect cold and hot trunks of a home's plumbing system. The pressure waves leveraged in the approach travel through this connection, enabling the HydroSense unit to detect both hot and cold water activity with a single sensor.

In summary, the plumbing system forms a closed loop pressure system, with water held at a stable pressure throughout the piping when no water is flowing. Homes with a pressure regulator have stable pressure unless the supply pressure drops below the regulator's set point. Homes without a regulator may experience occasional minor changes in water pressure depending on neighborhood water demand.

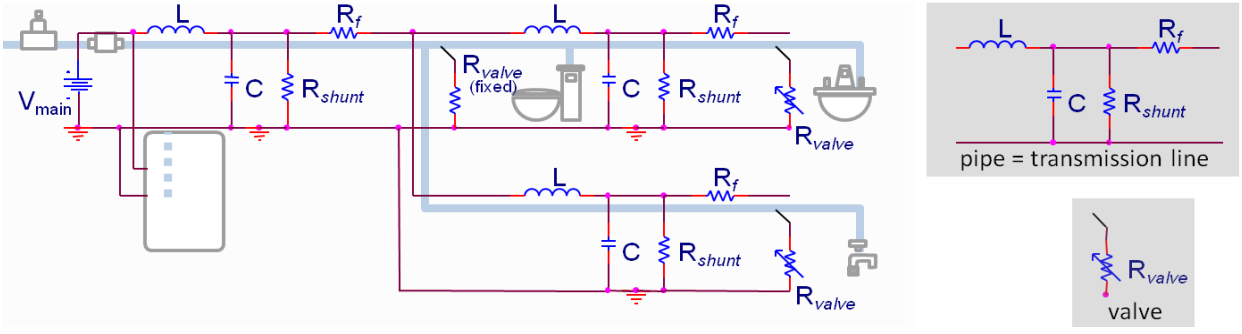

Figure 4-3. The transmission line equivalent network for a common bathroom fixture configuration. Only cold water is shown. In this analogy, pressure is analogous to voltage, water flow to electrical current, and the water valve and water pipe as shown.

4.1.1 Electrical Analogy, Transmission Lines and Equilibrium

For explanatory purposes, it is useful to draw an analogy between a household plumbing network and its electrical equivalent (Figure 4-3). Water pressure is analogous to voltage, water flow to electrical current,

and water valves are "electrical switches connected to ground." Each drain is like an electrical ground. **Regulator**: the regulator holding water pressure to a constant 45 psi is analogous to a 45VDC supply. Just as water flow occurs in the direction of higher pressure to a lower pressure, so does current flow from a higher voltage to a lower voltage. **Water Valves**: When a water valve is opened it is the same as if we closed an electrical switch connected to ground through a resistor (Figure 4-3). The value of the resistance controls the amount of current (*i.e.*, water flow) that escapes to ground. In this way, the size and amount that a valve is open controls its resistance and the aerator sets the minimum resistance possible. **Water Pipes**: In a simplified form, a water pipe is analogous to an electrical transmission line with an inductor and a capacitor (Figure 4-3, far right). In this way, the network of water pipes in a home can be modeled as a collection of electrical transmission lines (*i.e.*, a linear system with a transfer function given by the interconnection and length of pipes). The inductance in the transmission lines results from the momentum needed to move water that is at rest or to stop water that is in motion, the same way an inductor prevents a change in electrical current. The capacitance comes from the compressibility of water. I note that the water is considered to be highly "incompressible" compared to other liquids, and so the analogous capacitance values from fluid compression are quite small in the transmission lines. **Dampeners**: the "dampeners" installed in the plumbing system are designed to be highly compressible, which makes them analogous to large capacitors. In the same way that capacitors can store charge, the dampeners store potential energy when compressed. The dampener can be modeled efficiently by increasing the value of capacitance in the transmission lines, simplifying the system. **Water Heater**: the water heater also acts as a large capacitor. The buildup of water inside the tank is analogous to the buildup of charge in a capacitor. **Pressure Sensor**: The single pressure sensor acts like a voltmeter attached to one of the transmission lines. Changes to the circuit (*i.e.*, throwing a switch) will result in electrical transients throughout the system that we can observe using this single voltmeter. Although I will continue to use some electrical analogies in explanations throughout this thesis, they serve only as an additional supplement to the explanations and are not necessary to fully understand the principles upon which HydroSense operates.

4.1.1.1 Characteristics of Water Valve Activation and Deactivation

The instant a valve is opened or closed in a water fixture, a pressure change occurs and a pressure wave is generated in the plumbing system (Figure 1-2 or Figure 4-4). This occurs because we have added a "ground" to the system—we have changed it. As such, the system must shift to a new state of equilibrium. In the same way that there is a cyclic oscillation of charge and flow between inductors and capacitors in the transmission lines, there is a cyclic transfer of water flow until the system settles to a new state of equilibrium. Once equilibrium is reached there is a pressure gradient in the home while water

flows, the same way that the voltage in the electrical system is no longer constant when current is flowing. The pressure drop from the regulator to the open valve depends on the resistance in the pipes and the resistance of the valve. The pressure drop that we observe depends on where we place the pressure sensor (*i.e.*, the voltmeter). For example, if the "toilet" is running in Figure 4-3, then the pressure will drop along each pipe resistance where there is flow. There will be no pressure loss in pipes without water flow. Since there is no pressure drop along the branch where the sensor connects it will observe only the pressure drop from the last point in the system where water is flowing. When water is flowing on the branch where the sensor is connected (*i.e.*, if the hose spigot was activated), it will see the pressure drop along the entire branch. I have shown in past work that this observed pressure drop can be used to calculate the flow rate of water from a valve [59,85,141]. In this thesis, I use this drop as a feature for identifying different valves. I discuss how to measure pressure drop reliably later in this chapter.

Aside from the pressure drop, pressure transients are also created during the shift between equilibrium. Transient pressure waves results from the rapid change of water velocity in a pipeline. For example, when a running faucet is turned off, the valve closure abruptly stops the water flow. Flowing water in a pipe, however, has momentum and is compressible (*i.e.*, the inductance and capacitance in the transmission line equivalent). So, rather than stopping outright, the momentum compresses a finite volume of water against the closed valve and against the dampeners, which results in a buildup of pressure. Once this buildup reaches a critical point, the pressure starts to decompress and the water flow is reversed until it reaches the backflow prevention valve of the house, where the cycle repeats. This is analogous to the cyclic oscillation of charge between inductors and capacitors in the electrical system. Eventually, the resistances in the system dampen the oscillation and it reaches steady state equilibrium.

The water pressure wave is often referred to as a surge or water hammer and can create a loud hammering noise as the wave travels through pipes. The magnitude of the transient is dependent upon the operating pressure of the home, the flow of water through the valve, and the observation point. Appliances such as dishwashers or clothes washers control their valves mechanically and thus often create the most pronounced water hammer. An abrupt change in flow can create dangerously high pressure transients that exceed safe operating limits for residential pipes. A thermal expansion tank and water hammer dampeners offer partial arresting of these transients. Without the arrestors, the flow decreases so quickly that the resulting pressure wave can burst the pipe (like the arc voltage produced by opening a switch near an inductor). However, most valve openings/closings manifest as a water hammer transients that are harmless. Water hammer typically lasts several seconds, as the pressure in the entire system oscillates back and forth through the pipes. We can detect this water hammer effect anywhere along the plumbing infrastructure, thus enabling the single-point sensing approach.

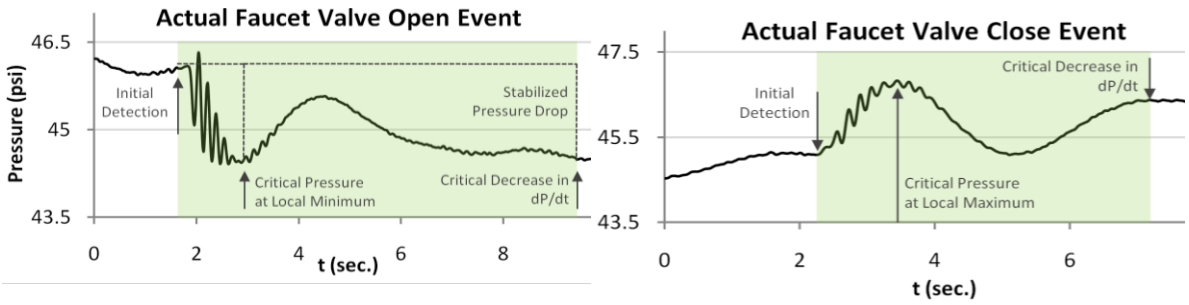

Figure 4-4. An actual event generated when a kitchen faucet is turned on and off, detected by the sensor at an exterior water bib.

The unique transient or water hammer signature that is sensed for a particular fixture depends on three factors: (1) the valve type, (2) the valve's location in the home pipe network, and, to a lesser extent, (3) the way in which the valve is opened or closed. Intuitively, you can think of each valve exciting the plumbing system differently along different points in its infrastructure. The physical reasons why these factors produce a unique transient, on the other hand, is best explained using the electrical analogy and knowledge of network analysis. The valve type controls the resistance of the analogous electrical switch. When the switch is closed, a low resistance path to ground is created forcing the system to shift to another state of equilibrium. The initial conditions from the circuit can be modeled as step inputs after the switch is closed. Valves with lower resistances allow more water flow and create step inputs with greater magnitude. The valve location affects where water flows and which initial conditions become step inputs. When the same step is applied in a different location, the system responds differently. New resonances will be excited and the dampening of those resonances will be different. The resonances that are excited are dependent on the system—different configurations of transmission lines (pipe) can create unique resonances. For example, in Figure 4-4 it is easy to see two resonant frequencies in each transient, the lower frequency resonance results from a long length of pipe (which excites a larger wavelength) and the high frequency resonance from a shorter pipe length. This point provides great discriminative power allowing me to distinguish between two fixtures of the exact same model such as the same toilet located in two different bathrooms of a home. I have even observed that the pressure transient generated between two cold sink valves located in the same bathroom (such as those found in a side-by-side bathroom sink setup, Figure 5-2) have distinguishable transients even though there is only a small additional length of pipe between them. From this perspective, one can predict the similarity of transients in a system by how close they are in the plumbing system. Valves that are close together in the plumbing system will excite many of the same resonances. If two close valves have substantially different flow rates, however, they will dampen the resonances differently, and even potentially altering the resonance that is excited. We can

expect valves in close proximity and with similar flow rates to have the most similar pressure waves. However, the way the valve is operated also has an effect that can skew this similarity.

For manually operated valves, the way the valve is operated can affect the step applied to the system. This is because the step input is not "ideal": the speed at which the valve is opened affects the slope of the step. Moreover, the flow rate the valve opens to affects the magnitude of the step, which is human controlled. As the speed of the valve opening becomes slower, the input excitation is not modeled well by a step, and instead is better modeled by a ramp function. In practice, slowly opened valves are rare occurrences (see section 4.2), but opening a valve to different flow rates is quite common (*e.g.*, opening a bathroom sink cold valve full stop to fill a cleaning bucket vs. opening the cold valve partially to wet a toothbrush). As the magnitude of the step becomes smaller, the amplitude of the resonances excited in each transmission line also becomes smaller, but the resonant frequency does not change dramatically. The relationship between step magnitude and resonance amplitude is non-linear and system dependent. Transmission lines near the switch are affected less than lines farther away. With this knowledge, one can try to select features which are resilient to these artifacts.

4.1.1.2 Plumbing Dynamics: Pressure Drop, Flow, and Transient Response

Changes in pressure and the rate of transient onset allow me to accurately detect and identify the source of valve open and valve close events. They also allow me to estimate flow. This stems directly from the electrical analogy, where knowing the resistance and the change in voltage allows one to determine the current. Unlike electrical current, however, water flow comes in two forms: laminar and turbulent, which affects the relationship between pressure and flow. It is beneficial to explore this relationship and quantify its effect on the linearity of pressure drops we may observe, as well as the non-linearity it may impose on the amplitude of pressure transients.

Laminar flow is characterized by the movement of fluid particles parallel to each other, with no transverse movement or mixing. In contrast, turbulent flow is characterized by vigorous intermixing within the flow field resulting in small, random fluctuations in flow. Physically, the two flow states are linked in that any laminar flow can become turbulent with a change in fluid velocity—think of the water flowing out of a typical bathroom faucet: when the faucet is opened partway, water flows in a clear, solid-looking stream and does not splash; however, when the faucet is opened all the way, the water flow turns more opaque, bubbly and chaotic.

The conditions under which liquids transition from laminar to turbulent flow are characterized by a dimensionless value known as the Reynolds number [127], which is dependent on the kinematic viscosity v of the fluid, the volumetric flow rate of the fluid in a pipe Q, and the radius of the pipe r:

$$R_e = Q\frac{2\,r}{\upsilon} \tag{4.1}$$

It is generally accepted that a Reynolds number less than 2,300 results in laminar flow and greater than 4,000 result in turbulent flow. In between there is transition between both [127]. Using the most common flow rates for residential water fixtures, it is possible to calculate the Reynolds numbers for a typical home. The Reynolds numbers for water flow in a typical home are in the range of 1,000-50,000 for 3/8" diameter pipe segments (twig lines) and in the range of 400-22,000 inside the 1" diameter trunk lines—which means fluid flow in a home can be laminar, turbulent, or a mix of both. I address the laminar flow condition first, showing that the relationship between pressure drop and flow is linear. I then show that the nonlinearity induced by turbulent flow and the difference in observation point of the sensor can cause many fixtures of different flow rates to have similar observed pressure drops. This highlights the inherent difficulty in feature selection.

Laminar Flow: When laminar, water flow is directly proportional to the pressure drop sensed at the HydroSense unit. Poiseuille's Law offers a precise definition of this relationship: when the flow through a pipe is laminar, the volumetric flow rate of fluid in a pipe Q is dependent on the radius of the pipe r, the length of the pipe L, the viscosity of the fluid μ and the pressure drop ΔP from the start and end of the pipe length:

$$Q = \frac{\Delta P\,\pi\,r^4}{8\,\mu\,L} \tag{4.2}$$

Note how increasing pipe length reduces flow (by creating more resistance) and increasing the pipe radius pipe increases flow. Pipe length has a linear relationship with flow; if you double the pipe length, the flow is halved. If, instead, you halve the pipe diameter, flow is reduced by a factor of sixteen. Note here that the viscosity of the fluid, μ, is dependent on the temperature of the water. For the ranges of temperature in the cold water line (5^{O}-15^{O}C), the viscosity is in the range of 1.5-1.1 mPa. The viscosity in the hot water line is held more constant and usually sits around 0.5 mPa. Because of this fluctuation with outside temperature, it is beneficial to continually update the model, a vital part of the semi-supervised architecture.

Poiseuille's law can be simplified by the fluid resistance formulation, which states that the resistance of flow is proportional to the drop in pressure divided by the volumetric flow rate.

$$R_f = \frac{\Delta P}{Q} \equiv \frac{8\,\mu\,L}{\pi\,r^4} \tag{4.3}$$

Thus, we can use fluid resistance to abstract some of the variable complexity from Poiseuille's law, resulting in:

$$Q = \frac{\Delta P}{R_f} \tag{4.4}$$

This is analogous to Ohm's law ($I = V / R$). HydroSense measures the change in pressure ΔP from the pressure regulator to the sensor. In order to compute Q, we must estimate the remaining unknown R_f . R_f is bounded by two factors: (1) water viscosity, which can easily be calculated according to temperature and (2) the radius of residential pipes, which are either 1/4" or 3/8" in diameter (1" for supply lines in an apartment). This leaves L, the length of the pipe, as the main unknown. L will change depending on the water fixture being used, as each path from intake to fixture is different. We observe that pressure drop with the sensor somewhere along the pressure gradient from water intake to the open valve.

Turbulent Flow: When the flow of water through the pipes is turbulent instead of laminar, Poiseuille's law does not directly apply. Turbulent flow results in a non-linear relationship between pressure and flow. Instead of pressure being proportional to flow, the relationship becomes such that P is approximately proportional to $Q^{1.75}$. The mixing induced in turbulent flow dissipates the force from the head pressure, requiring that more pressure be exerted to sustain the same flow of water. The relationship between pressure and mean flow can be determined empirically, and is given by

$$\Delta P = Q^2 \frac{L\,\rho\,0.3164}{4\,r\,\sqrt[4]{R_e}} \tag{4.5}$$

where ρ is the density of the liquid. This is known as the Blasius formula [13] and is valid for liquid flow with Reynolds numbers less than 100,000. Also note that the Reynolds number is temperature dependent. The pressure and flow relationship given by the Blasius formula is plotted in Figure 4-5 (blue) for a typical pipe length of 2 meters. Also shown is the typical operating range for flow in a household and the best fit linearization (black) of the Blasius formula in the most common operating range (top left inset). The slope of the black line is proportional to the estimated R_f from Poiseuille's law. The actual R_f is proportional to the instantaneous slope of the blue line. The worst case approximation error of R_f is 70% at a flow rate of 10 gpm.

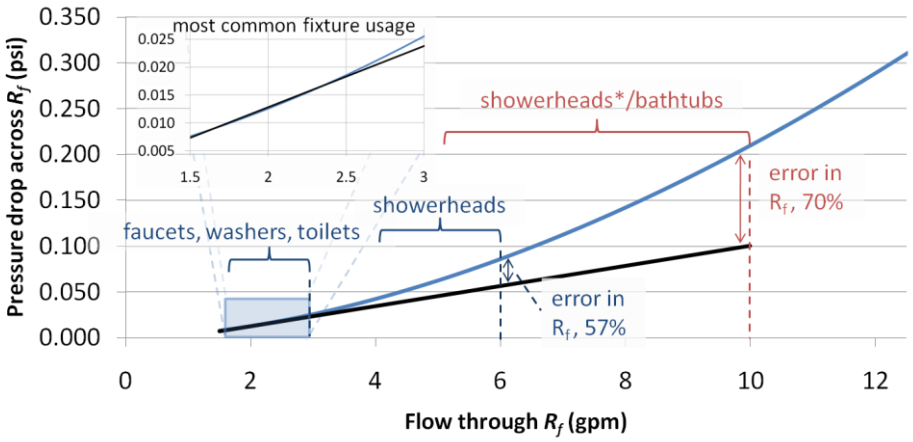

Figure 4-5. The pressure versus flow in a 1" diameter pipe of length 6.5' using the Blasius formula (blue) and the linearization of the equation (black) around the point of most common fixture usage. Also displayed are the most common fixture flow rates. The (*) denotes a fixture built before the 1992 Energy Policy Act, requiring US manufactured fixtures to use less water.

Note that, for disaggregation, I am not necessarily interested in attaining the exact flow rate (as in [141]), but instead I am interested in how unique the pressure drop for a particular fixture is compared to others in the household. The differences in pressure and flow rate for turbulent and laminar flow and the observation point means that the pressure drop of a valve is not necessarily a unique feature. Turbulent flow from a bathtub could result in a pressure drop that is similar to that at a kitchen sink if the sensor is nearer to the sink. Moreover, the pressure drop from hot water use on the kitchen sink is different than cold water use, which is also different than when we combine hot and cold together. However, one main observation stays intact: a change in flow results in a change in pressure. This means that a manually controlled valve like a shower or faucet will change its pressure drop if the valve is adjusted. This could be reducing the flow rate at the kitchen sink, or adjusting the temperature in a shower. A toilet, on the other hand, will have a predictable and stable pressure drop because there is no mechanism to change the flow rate—an observation that I leverage in the feature extraction. Also, if a bathroom sink and a bathtub are on the same branch, the tub will always have the larger pressure drop, and the shower will be somewhere in the middle. From this perspective, even though I have shown there can be dramatic shifts in the pressure and flow relationship, it is still a powerful feature for disaggregating fixtures. In fact, this ambiguity can sometimes help make a particular pressure drop unique, especially if there is a fixture on an isolated branch in the plumbing system.

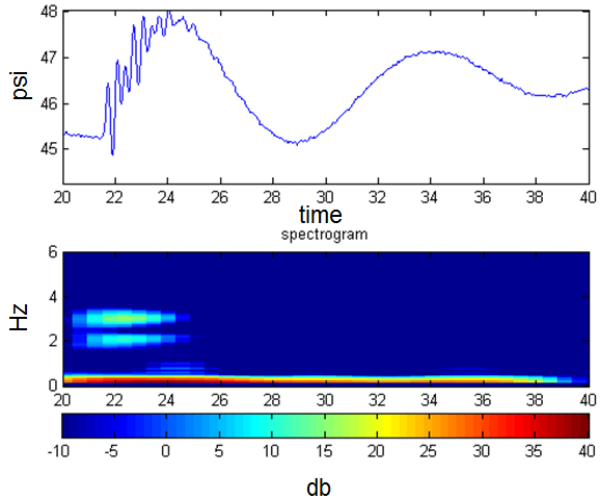

Figure 4-6. Spectrogram illustrating widening of bandwidth as the resonant frequency increases.

Lastly, the laminar and turbulent flows can provide insight into how resonances are dispersed across frequency. Recall that different valves can excite different resonances. Also recall that there is almost always turbulent flow somewhere in the piping system, usually along the smallest diameter pipe or in bends in the piping system. This turbulence has the effect of dispersing the energies of resonances and widening the bandwidth of the observed resonance. At the basic level, the pressure waves are compressions and decompressions of fluid. As they move through a turbulent mix the compressions' energies become dispersed. For low frequencies below 1 Hz, turbulent piping is only traversed once a second or less. Higher frequencies, however, traverse the turbulent flow more often, and thus, these higher frequencies are more dispersed by the turbulent flow than low frequencies (Figure 4-6 shows a spectrogram of the excited resonances and their bandwidth). We can leverage this phenomenon in the feature extraction (next section) by choosing a filter bank representation that has widening bandwidths.

Characterizations: I note that these equations are not comprehensive. They do not account for the smoothness of the inner pipe surface, the number of bends, valves, or constrictions in pipes, nor pipe orientation (*e.g.*, the forces of gravity and changes in barometric pressure). While many of these factors can be estimated using home size, type of plumbing (PVC, copper, *etc.*), and number of fixtures, I have found these effects can be treated as negligible for home pipe networks.

4.2 Initial Study: Are the pressure waves unique enough for classification?

The transmission line equivalent system tells us that if the pipe pathways are different enough, the pressure waves can look different from one another. I investigate whether the pressure waves truly are unique by collecting data in 10 different homes and performing template matching of the transient pressure waves.

ID / Water Supply	Style / Built / Remodel	Size / Floors / Fixtures	Exp. Tank/ Regulator / Recirc. Pump	Water Heater / Plumbing/ Static PSI	Sensor Install Point
H1 Public Utility	Single-Family 2002	3200 sqft 2 flr + bas 12 fixture	Yes Yes No	Tank PVC 46 PSI	Hose Bib
H2 Public Utility	Multi-Family 1909/96	2160 sqft 2 flr + bas 5 fixtures	No No No	Tankless Copper 46 PSI	Hose Bib
H3 Public Utility	Single-Family 2003	4000 sqft 2 flr + bas 6 fixtures	Yes Yes No	Tank Copper 41 PSI	Hose Bib
H4 Public Utility	Single-Family 1921	1630 sqft 1 flr + bas 4 fixtures	No No No	Tank Galvan. 43 PSI	Hose Bib
H5 Public Utility	Single-Family 1913	2000 sqft 2 flr + bas 5 fixtures	No No No	Tank Copper 55 PSI	Hose Bib
H6 Public Utility	Single-Family 1974/85	3100 sqft 2 flr 8 fixtures	Yes Yes Yes	Tank Galvan. 46 PSI	Hose Bib
H7 Public Utility	Aprtmnt 1927	746 sqft 1 flr 5 fixtures	No Yes No	Tank Cop+Gal 33 PSI	Water Heater
H8 Public Utility	Single-Family 1922 / 2006	3650 sqft 2 flr + bas 3 fixtures	Yes Yes Yes	Tank Copper 75 PSI	Utility Sink Faucet
H9 Public Utility	Single-Family 1904 / 95 est.	1790 sqft 2 flr + bas 4 fixtures	No No No	Tank Copper 72 PSI	Hose Bib + Water Heater
H10 Private Well	Resort Cabin 1950/80	900 sqft 1 flr 4 fixtures	No No No	Tank Galvan. 65 PSI	Hose Big

Table 4-1. A summary of the homes in which I collected data, including the style, size (1 sqft ≈ .093 sqm), age of the home, how many fixtures I tested, characteristics of the plumbing system, and where I installed the sensor.

4.2.1 Data Collection for Feasibility Study

In order to validate the general approach, I collected labeled data in ten homes, in four cities, of varying, style, age, and diversity of plumbing systems (see Table 4-1).

For each home, I first measured the baseline static water pressure and then installed the HydroSense unit on an available water hose bib, utility sink faucet, or water heater drain valve. Each collection session was conducted by a pair of researchers: one would record the sensed pressure signatures to a laptop while the other activated the home's water fixtures. The pressure signatures were recorded using a graphical logging tool, which also provided real-time feedback of the pressure data via a scrolling time-series line

graph. I conducted five trials per valve on each fixture (*e.g.*, five trials for hot water and five trials for cold water). For each trial, a valve was opened completely for at least five seconds and then closed. For the toilet trials, the toilet flush and full fill cycle were logged. Note that for the faucet experiments, I did not collect data on partially opened valves nor the speed with which they were opened. I return to this issue in the discussion section.

For four of the ten houses (H1, H4, H5, and H7), I also collected flow rate information for the faucet (kitchen and bathroom) and shower fixtures. In addition to logging sensed pressure, I measured the amount of time it took to fill a calibrated bucket to one gallon (a method preferred by water utilities for accurately measuring flow). This was repeated for five trials for each valve. This in-home data collection yielded a total of 706 fixture trials and 155 flow rate trials across 84 fixtures.

4.2.2 Fixture Event Identification

Given the collected data, I now pursue a three-step approach to examine the feasibility of identifying individual fixture events according to the unique transient pressure waves that propagate to the sensor. I first *segment* each individual pressure wave from the stream, identifying its beginning and end to enable further analysis. I then *classify* each a wave as either a *valve open* or a *valve close* event. Finally, I *classify* the event according to the *individual fixture* that generated it. This feasibility study explicitly considers only events that occur in isolation, deferring the analysis of compound events (when more than one fixture is active) until a later chapter.

4.2.2.1 Segmentation

Segmentation must be effective for many different types of events, and so it is important to consider only features that are likely to be most typical of all valve events. The approach is illustrated in Figure 4-4 and involves the filtering the signal using low pass filters and a derivative filter. The raw signal is smoothed using two low-pass linear-phase finite impulse response filters (FIR) with different cutoff frequencies (1 Hz and 25 Hz, $p_1(t)$ and $p_2(t)$ in Figure 4-7, respectively). Almost all the spectral energy is below 15 Hz. Thus the 25 Hz filter captures a de-noised version of the transient (most noise is generated from RF noise on the ADC line). Below 1 Hz the relative *decrease* or *increase* in pressure is more apparent which can aid in classifying the transient as a valve open or valve close event. Lastly, I also calculate a derivative using a band-pass FIR filter with a cutoff of ~25Hz and linear magnitude response with frequency below 25Hz.

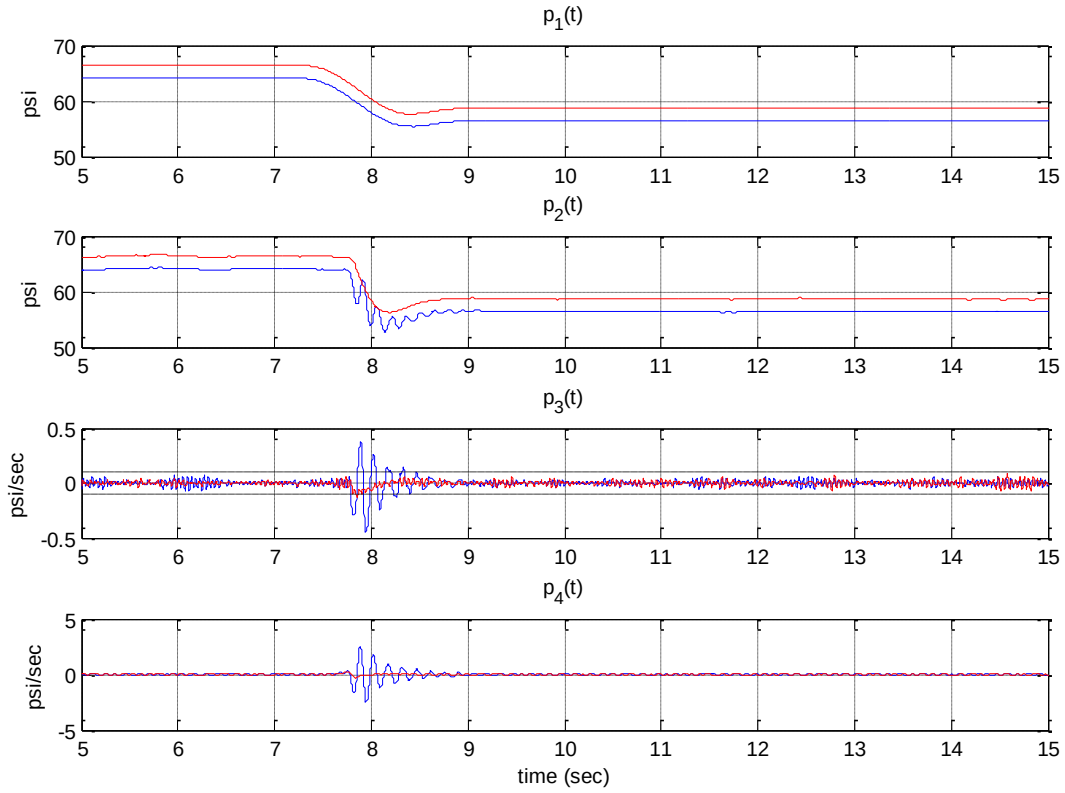

Figure 4-7. Example filterings for a single pressure wave open event, toilet in H2.

The beginning of a valve event corresponds to one of two conditions. The most common is when the derivative of the smoothed signal exceeds a specified threshold relative to static pressure, indicating a rapid change. For a pilot home (static pressure of 45 psi), this value was found empirically to be approximately 2 psi/sec (Figure 4-7, bottom). For generalizing to other homes, this value was scaled by the home's actual static pressure. The less common second condition is when the difference between the maximum and minimum pressure values in the sliding window exceeds a threshold relative to static pressure, indicating a slow but substantial change (approximately 1 psi for a home with 45 psi static pressure, scaled by the actual static pressure).

Each transient represents a damped sinusoid. After the beginning of a valve event is detected, the next change in the sign of the derivative represents the time at which the extreme value of the pressure transient occurs (which may be a maximum or a minimum). This extreme value is used to assist in the calculation of the end of the transient. The end of a segmented valve event is typically detected as the first point at which an extreme of a fluctuation (a change in the sign of the derivative, *dP/dt*) is less than 5% of the magnitude of the first extreme following the beginning of the event. It is also possible for an event to be ended by a rapid increase in the magnitude of fluctuation. This corresponds to the occurrence of a

compound event, as I will discuss in greater detail in a later section. Applying this method in a controlled study yielded appropriate segmentations of 100% of the valve events (706 events in 10 Homes) from their surrounding sensor stream [59]. After segmentation, various features from the transient and surrounding sensor stream are calculated.

4.2.2.2 Classifying Open and Close Events

After segmenting each valve event, I classify it as either a *valve open* or a *valve close* event. I apply a hierarchical classifier that first considers the difference in the smoothed pressure at the beginning and the end of the segmented event. If the magnitude of this difference exceeds a threshold (approximately 2 psi for a home with 45 psi static pressure, scaled by the actual static pressure), the event can be immediately classified (a pressure decrease corresponds to a *valve open* and a pressure increase to a *valve close*). Otherwise, the event is classified according to the average value of the derivative between its beginning and its first extreme. A *valve open* creates an initial pressure decrease (a positive average derivative), while *valve close* events create an initial pressure increase (a negative average derivative).

Applying this method to the segmented valve events from the collected in-home data yields 100% correct classification of *valve open* and *valve close* events.

4.2.2.3 Classifying Different Fixtures

For the initial feasibility study, I associate *valve open* and *valve close* events with individual fixtures in a home using a template-based hierarchical classifier. When classifying an unknown event, I first filter potential templates according to four complementary distance metrics. The distance metrics are discussed in depth in the Chapter VI. Here I simply wish to convey a flavor of the matching that can be used to compare different pressure waves.

The first distance metric I use is a *matched filter*. Very common in signal detection theory, the matched filter is the optimal detection mechanism in the presence of additive white noise. Its primary limitation is that the signals I want to differentiate are not orthogonal. Making them orthogonal would require specific knowledge of the source of each event, exactly the information I want to infer.

The second distance metric is a *matched derivative filter*. I include this because the derivatives of the events always resemble exponentially decreasing sinusoids. It is therefore reasonable to believe the derivatives are more orthogonal than the original pressure signals, and that this filter might provide value distinct from the above filter.

The third distance metric is based on the *real Cepstrum*, which is the inverse Fourier transform of the natural log of the magnitude of an event's Fourier transform. This approach attempts to approximate the

original version of a signal that has been run through an unknown filter (the valve event I am trying to classify has been transformed by propagation through an unknown path in a home's water pipes). It can be shown that the lower coefficients of the Cepstrum result largely from the transfer function (an event's propagation through a home's pipes) and the higher coefficients largely from the source (the original valve pressure event). I am interested primarily in the transfer function (in part because it allows differentiating among multiple instances of identical fixtures in a home), and so I truncate the Cepstrum to the lower coefficients. The resulting space is highly orthogonalized, yielding a third effective and complementary matched filter.

Finally, the fourth distance metric is simple *mean squared error*, computed by truncating the longer of two events.

Similarity thresholds used to filter potential templates based on these distance metrics are learned from training data (filtering templates whose similarity to the unknown event are less than the minimum within-class similarity in the training data). If no template passes all four filters, the unknown event is not classified (an application might ignore the event, prompt a person to label an unrecognized fixture, or consider the possibility that the new event indicates the presence of a leak). If templates corresponding to multiple fixtures pass all filters, I choose among them using a nearest-neighbor classifier defined by the best performing distance metric, the matched derivative filter.

4.2.3 Initial evaluation

I evaluate fixture classification using an experimental design selected to demonstrate robustness of learned model parameters across the multiple homes in the collected data. Specifically, I conduct a cross-validation experiment that folds the data according to the home in which it was collected. There are ten trials in the cross-validation, with each trial using data from one home as the test data and data from the other nine homes as the training data. After learning model parameters from the test data (the four similarity filter thresholds), I classify each event in the test home using a leave-one-out method. Each test home event is classified using the other events as templates together with the model parameters learned in training.

Home	Fixture Open Identification	Fixture Close Identification
H1 (12 valves)	100%	100%
H2 (8 valves)	96.4%	100%
H3 (6 valves)	100%	100%
H4 (5 valves)	96.2%	100%
H5 (9 valves)	100%	100%
H6 (8 valves)	100%	90.0%
H7 (8 valves)	100%	100%
H8 (6 valves)	100%	97.1%
H9 (7 valves)	97.1%	97.1%
H10 (7 valves)	97.1%	77.1%
Aggregate	98.9%	96.8%
	97.9%	

Table 4-2. In a cross-validation test of the robustness of learned model parameters across multiple homes, the template-based classification enables identification of the individual fixtures associated with valve open and valve close events with aggregate 97.9% accuracy.

Table 4-2 presents the results of this evaluation. The figure shows the accuracy of fixture-level identification of *valve open* and *valve close* events within each home (and thus each test fold of the cross-validation), as well as the aggregate 97.9% accuracy of fixture-level classification. The relatively poor performance in identifying fixture close events in H10 (77.1% vs. > 90% for other homes) was due to noise from the eleven cabins that share the same supply line at the resort. The sensor was picking up water events from a portion of these cabins during data collection, because the cabin was not separately metered. More work is needed to disambiguate signals in a single-meter multi-unit domain (*e.g.*, a duplex, small apartment building), but these results indicate a single sensor may be sufficient to sense across more than one housing unit on a shared supply line.

Fixture Type (number of fixtures)	Fixture Open Identification	Fixture Close Identification
Sinks (27 in 10 homes)	98.1%	95.1%
Toilets (14 in 10 homes)	98.7%	97.5%
Showers (8 in 8 homes)	95.5%	89.4%
Bathtubs (3 in 3 homes)	100%	100%
Clothes Washer (2 in 2 homes)	100%	100%
Dishwasher (1 in 1 home)	100%	N/A

Table 4-3. A different view of the results, showing accuracy of identification of individual fixtures by fixture type.

Table 4-3 presents a different view on the same data, showing the accuracy of fixture-level classification for different types of fixtures across homes. The overall fixture-level classification across all

homes is above 90%, including a number of cases where classification accuracy is 100%. The dataset contains only a few instances of clothes washer or dishwater use, in part due to time constraints during data collection and in part because Fogarty *et al.* found these fixtures can be easily recognized by their structured cycles of water usage (an approach that I combined with ours in Chapter VIII). However, I note that using the pressure waves signature is independent of the number of fill cycles (important if a dishwasher is sometimes run with a pre-rinse cycle) and allows recognition as soon as these appliances first use water (in contrast to being able to recognize them only after their pattern of fill cycles becomes apparent).

The initial results presented in this chapter show significant promise for single-point sensing of whole-home water activity via continuous monitoring of water pressure. I have presented a reliable method for segmenting valve pressure events from their surrounding sensor stream and for determining whether a segmented event corresponds to a valve being opened or closed. Using data collected in ten homes, I have shown 97.9% aggregate accuracy for identifying the individual fixture associated with a valve event. The ability to identify activity at individual fixtures using a single sensor is itself an important advance.

4.3 Summary

I explained the theory of operation of the HydroSense system and performed an initial feasibility study of whether or not valve signatures are unique enough in a home for classification. I note that there are a number of limitations in this feasibility study which I address in subsequent chapters. However, I was able to show that the pipe pathways and the flow rate of the valve dramatically affect the observed signal, enough that the pressure waves are distinguishable.

CHAPTER V

5. Chapter Five:
Real World Water Usage Data Collection

This chapter summarizes the data collection procedure for collecting real world water usage events. Before evaluating each of the algorithms, I need a large dataset of water hammer transients to inform and test the algorithms. To evaluate the performance using *real-world* data, I deployed a large ground truth water usage sensing network in three homes and two apartment units. At each deployment site, I installed two pressure sensors and directly instrumented *all* water fixtures and appliances with custom wireless sensors that provided ground truth labels of water usage activity for the pressure stream. Here, I describe the ground truth data collection system and the five-week study deployment.

5.1 Acquiring Ground Truth Labels in a Real-World Deployment

A key challenge in evaluating any new sensing technique is acquiring ground truth data. In the original feasibility work [59], I manually labeled the pressure stream during staged experiments, which clearly would not work for a real-world evaluation. Thus, an automated method for labeling was derived. An ideal labeling system would accurately detect when fixtures are turned on/off, be easy to install, work across a large variety of fixtures, and preferably provide flow and temperature information for each fixture valve. An accurate and direct approach would be to install small, wireless flow meters at each hot and cold fixture inlet (*e.g.*, a sink would require two flow meters). Unfortunately, inline flow meters could actually distort the very phenomena I am interested in studying by impacting the pressure-wave signal itself. Instead, I instrumented fixtures externally, such as on faucet and toilet handles, so that I did not disturb the water stream. Unfortunately, this also makes it impossible to know the exact flow rate of water from the valves. As such, direct flow rate is not a contribution of this thesis. My collaborations with Swanson *et al.* cover this topic in depth. See [141] for a comprehensive analysis of performing regression on the pressure stream to infer flow.

I designed an array of ground truth sensors to accommodate the variety of home water fixtures: from hand operated fixtures like sinks to electromechanical appliances such as dishwashers. Even for a single

fixture type, design variation affects how flow and temperature are selected and how they can be sensed. For example, some single-handle faucets move left to right for temperature and up or down for flow while dual-handle faucets select both temperature and flow by the open position of each handle.

5.2 Water Usage Activity Ground Truth Sensors

I developed seven ground truth sensors to accommodate all fixtures across the deployment sites. Each interfaced with a *parent sensor board* (wireless platform in Figure 5-1, top right) to communicate water usage data in real time. At a minimum, I tracked when each valve was opened or closed and categorized temperature into *hot only*, *cold only*, and *mixed*. The parent sensor board was placed in a location protected from water and preferably next to a power outlet (5 of 33 ground truth sensor boards relied on battery power). All sensors and parent boards were waterproofed to protect against water damage using a mixture of conformal coatings. XBee Pro wireless modems (Figure 5-1, top left) transmitted sensor state to a logger on a laptop installed at each deployment site. The sensor boards were configured to transmit a watchdog signal once every four minutes so failures could be quickly identified and corrected. The ground truth architecture and sensors went through several design cycles and took approximately three months to build and evaluate before being deployed.

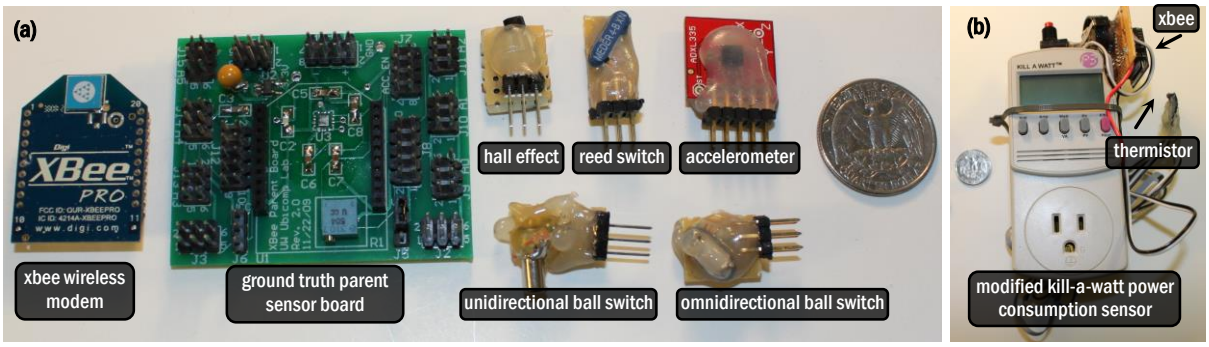

Figure 5-1. The ground truth water usage sensors directly attached to (a) *fixtures* and (b) *appliances* and monitored valve *openings and closings.* This data was transmitted wirelessly in real time via the ground truth parent sensor board and a XBee wireless modem (a, left side) to a data logger.

For sinks, showers, and toilets, sensors to detect handle position were affixed directly to the fixtures themselves and linked to the wireless parent board via low-voltage wires (Figure 5-2). I used three types of handle sensors: *reed switches* (*N*=34 sensors deployed), *accelerometers* (*N*=14), and *Hall effect sensors* (*N*=3). Reed switches are electrical switches that react to the presence of a magnetic field and produce binary output: on or off. They are inexpensive, robust to water exposure, and provide easily analyzable data. For toilets, I instrumented the flush handle, which only provided data on the beginning of the fill and not on the end. I discuss how this end fill information was recovered in the next section.

For faucets where a single handle controls flow rate and temperature, the reed switches were insufficient. Instead, I used three-axis accelerometers (Figure 5-2, *a* and *b*) to measure acceleration and interpret the handle's flow position (typically up and down movement) and temperature (typically left and right movement). Finally, I used Hall effect sensors for sensing faucets which control temperature using planar rotation but control flow through an up/down motion (*i.e.*, where an accelerometer alone could not sense the planar motion). A Hall effect sensor provides a voltage difference representing the distance between two magnets, so I placed magnets on both sides of faucet handles and attached the Hall effect sensor to the handle itself.

Additionally, each hand-operated fixture had at least one omni-directional *ball switch* (*N*=39) that acted as a vibration sensor and woke the parent board to read and transmit handle position sensor data. This allowed me to limit power consumption and unnecessary XBee wireless traffic.

For washing machines and dishwashers, I used three types of sensors: *power usage sensors* (*N*=7)**,** *push buttons* (*N*=2), and *thermistors* (*N*=3). Power consumption patterns were used to reconstruct when appliances used water. I could not gain access to the power outlets in two cases (deployment site A1's washing machine and H1's dishwasher), so I used push buttons and a note reminding the resident to "*please push button when turning on <appliance>*." For sites with washing machines, I also attached thermistors to the water drain pipe to measure the temperature of the previous fill cycle and infer machine settings (*e.g.*, Hot/Cold, Warm/Cold).

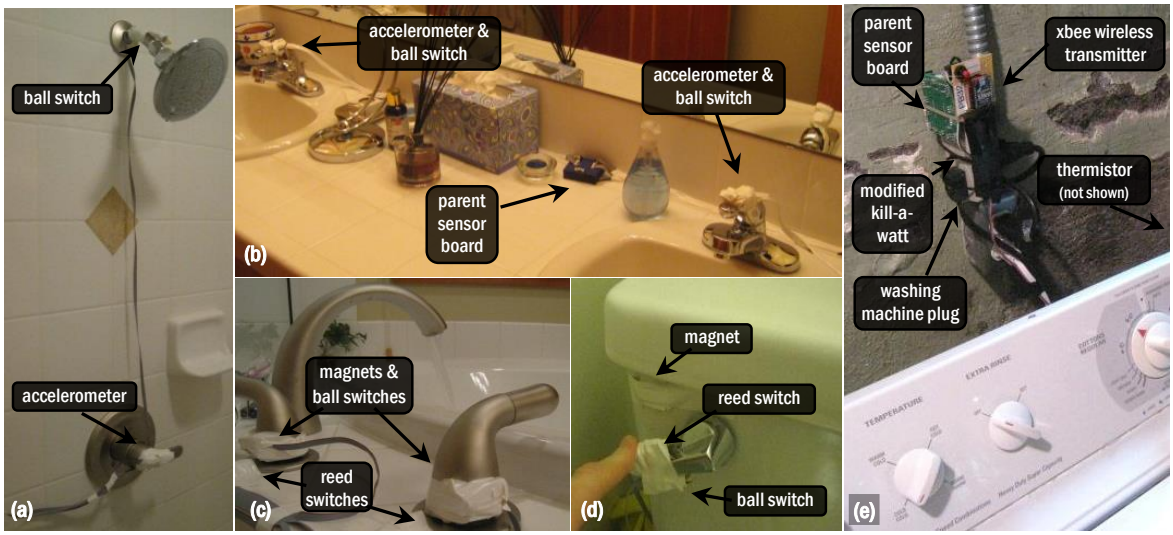

Figure 5-2. A sample of instrumented fixtures from the ground truth deployments. Note how different sensors (*e.g.*, accelerometers, reed switches) are used to accommodate the variety of fixture types.

5.3 Pressure Sensors and Software Tools

The above sensor network was deployed at each deployment site to provide *ground truth labels* for the pressure sensors. For the pressure sensors, I used Pace Scientific P1600s with a resolution of 0.03 psi. Each was connected to a 16-bit Texas Instruments ADS8344 ADC and AVR microcontroller, which interfaced with a Class 1 Bluetooth radio implementing the serial port profile with an approximate wireless range of 10m. This is the same setup as the original HydroSense study with three exceptions. First, instead of one pressure sensor, I connected *two* sensors to collect data from hot *and* cold water access points simultaneously (Figure 5-3). This allowed me to investigate the effect of installation point as well as the effect of two pressure streams compared to one on classification performance. Second, the original HydroSense work tested only $^{3}/_{4}$" water access points (*e.g.*, hose spigot). I built adapters to connect to $^{3}/_{8}$" access points, which allowed me to install pressure sensors below kitchen and bathroom sinks (Figure 5-3, right). This was particularly important for the apartment installations, which did not have accessible $^{3}/_{4}$" access points. Finally, I used a sampling rate of 500Hz rather than 1,000Hz, as I found 500 Hz was more than sufficient to capture these pressure waves.

Figure 5-3. Two pressure sensors were installed at each deployment site (one on a hot water access point, one on a cold) in order to study the effect of installation points on classification accuracies.

To communicate with the ground truth sensor network and the pressure sensors, a 2GHz Dell Inspiron 1545s laptop running Windows XP was deployed at each site. The laptops were configured with a USB XBee wireless modem and Bluetooth dongle. The laptops continuously ran a custom data logger written in C#, which received, compressed and archived data locally for backup. This was uploaded to a backend webserver at 30-minute intervals. The server backend was implemented using Python and web2py. In addition to serving as a data repository, the backend automatically sent e-mail notifications when a ground truth sensor or pressure sensor was not heard from for 10 minutes or more. For analysis, I constructed a suite of tools in Matlab and C#. Because not all of the ground truth sensors provided direct

labels about water usage (*e.g.*, the power usage sensors and toilet handle sensors), I also used a custom annotation tool in C# that allowed me to quickly review the ground truth sensor streams and semi-automatically annotate the pressure stream.

5.4 Deployments

I deployed the ground truth sensor network and two pressure sensors at five sites: three houses and two apartments. Each site was a home or apartment of a lab member. This was done because of the invasiveness of the direct sensing approach used for the ground truth data collection. There was, however, a large variation in the type, size and plumbing systems across the deployments sites (Table 5-1). The deployments began February 2010 and lasted for five weeks.

It took approximately two full work days per deployment site for two people to install and test the ground truth sensors. After the five-week ground truth deployments ended, I used a custom annotation tool to convert the ground truth sensor stream to labels. This was accomplished in a semi-automatic fashion—the annotation tool visualized the ground truth sensor values and the pressure streams together in a common time-series view. The ground truth sensor values could then be automatically or manually converted to labels. It took approximately 8-12 hours per week of data collected for one research assistant to convert the sensor stream to labels. These labels were then reviewed by a second research assistant for consistency, which took roughly half the time (4-6 hours per week of data).

Note that the main bottleneck was in the labeling phase. I actually collected much longer than 5 weeks of data at each site, but could only afford the time of labeling 5 weeks. In the future I could label many of the data streams to test the performance of the algorithms over longer periods. Additionally, I could run algorithms on the unlabeled data for various experiments and to test hypotheses regarding what I expect the data to look like.

Also note that I did not label the first week at a deployment site when the sensors were installed to mitigate the novelty effects that the sensors might have upon actual water use. Most subjects in the deployment said that after a few days they did not even register that the sensors were installed. In A1, one of the subjects actually did not notice that the sensors had been uninstalled until three days after they were taken out. This observation supports the hypothesis that after the sensors are installed, they tend to drift into the background and water use goes back to normal (if changed at all).

5.5 Analysis of the Collected Dataset

I collected a total of 16,056 labeled events across the five deployment sites. Table 5-1 provides an overview. Due to ground truth sensor failures, 2.9% of this data is marked as *uncertain* and is not used in

the classification experiments. Nearly 80% of the uncertainties were due to malfunctioning kitchen sink handle position sensors in H1 and H2, which were replaced within a few days of discovery. The dataset also includes *unknown* events (3.9% of the dataset), which are pressure stream transients whose origin cannot be determined because they occurred without accompanying data from the ground truth sensors. A1 has the highest proportion of unknown events (9.1%) because of water usage activity coming from other apartments. Although I do not attempt to classify uncertain or unknown events, they were not removed from the dataset and can impact classification performance when they overlap with other events. After accounting for uncertain/unknown events, I am left with 14,960 labels.

	H1	**H2**	**H3**	**A1**	**A2**
# Residents	2	2	4	2	2
Gender/Age/ Profession	M/27/professor; F/29/professor	M/31/professor; F/32/office worker	4 Males/19-21/ undergrad students	M/31/grad student; F/30/post-doc	M/26/grad student; F/26/pharmacist
Fixtures/Valves	17/28	8/13	13/21	6/10 *(8/13)**	8/13
Style/Built	House/2003	House/1918	House / 1923	Apt/1920s	Apt/2000
Size/Floors	3000 sqft/ 2 floor + basement	750 sqft/ 1 floor + basement	1200 sqft / 1 floor + basement	700 sqft/ 3rd floor of 3	750 sqft/ 6th floor of 7
Expansion Tank/ Regulator	Yes/Yes	No/No	No/No	N/A	N/A
Water Heater Tank Size/ Plumbing	50 gal/ Copper	50 gal/ PEX	50 gal/ Copper	Two 100 gal tanks/ galvanized	N/A/ PEX
Pressure Sensor Install Point Hot/Cold	Main floor bathroom sink/outdoor hose spigot	Water heater drain valve/outdoor hose spigot	Downstairs bathroom sink/outdoor hose spigot	Bathroom sink hot/cold inlet	Kitchen sink hot/cold inlet

Table 5-1. Occupant demographics and deployment site characteristics. In A1, The toilet and shower head were replaced with low-flow equivalents ~3.5 weeks into the deployment. I discuss the effect of this change on classification performance in the results section.

Table 5-2 shows valve activity at individual fixtures by temperature state (hot, cold, mixed). I use *M.* for Master and *S.* for secondary to distinguish primary and secondary bathrooms. The *M. Bath Diverter* and *S. Bath Diverter* are for the tub/shower switch that diverts water flow from the bath to the shower and vice versa; I distinguish between a shower that is turned on straightaway and a shower that is diverted from a bath. The *Other* category includes data from only one deployment site, H1, and encompasses the *Laundry Basin* and the *Refrigerator Water Dispenser*. On average across all deployment sites, there is a nearly even proportion of cold and hot events (40.7% for cold only, 39.2% for hot, and 20% for mixed). This implies that any indirect water disaggregation sensing method, such as flow-trace analysis and HydroSense, must be equally capable of sensing usage regardless of temperature. The overall frequency of fixture usage follows a power-law distribution where the first four fixtures (*kitchen sink, master bathroom sink* and *toilet,* and *secondary bathroom sink*) account for 84.7% of the events in the dataset.

For purposes of human activity inference, these fixtures are thus critically important. Note that the table summarizes the frequency of use for each valve, not the percentage of consumption. As noted, I could not collect flow rate for each water draw and therefore cannot gauge consumption.

	H1	H2	H3	A1	A2	Totals
Days of Data	33	33	30	27	33	156
Total Events	2374	3075	4754	2499	2578	14960
Avg Events/Day	71.9	93.2	158.5	92.6	78.1	95.9
Cold Only Events	855 (36.0%)	1418 (46.1%)	1637 (34.3%)	633 (25.3%)	1657 (64.3%)	6087 (40.7%)
Hot Only Events	607 (25.6%)	1329 (43.2%)	1766 (37.5%)	1818 (72.8%)	498 (19.3%)	5870 (39.2%)
Mixed Temp Events	912 (38.4%)	328 (10.7%)	1351 (28.2%)	48 (1.9%)	423 (16.4%)	3003 (20.1%)
Isolated Events	1981 (83.5%)	2477 (80.6%)	4131 (86.9%)	1914 (76.6%)	2149 (83.4%)	12393 (82.8%)
Compound Events	393 (16.6%)	598 (19.5%)	623 (13.1%)	585 (23.4%)	429 (16.6%)	2567 (17.2%)
Transient Collisions	142 (6%)	72 (2.3%)	166 (3.5%)	219 (8.8%)	120 (4.7%)	701 (4.7%)
Uncertain Events	22 (0.9%)	175 (5.3%)	189 (3.7%)	52 (1.9%)	37 (1.4%)	467 (2.9%)
Unknown Events	52 (2.1%)	79 (2.4%)	184 (3.6%)	254 (9.1%)	85 (3.1%)	629 (3.9%)

Table 5-2. High level ground truth data collection statistics. An *event* is one occurrence of either a valve open or a valve close. Uncertain and unknowns are *not* included in the totals events row.

Although I ultimately used this data to evaluate the classification algorithms, an equally important goal was to identify potential challenges in classifying real-world water usage compared to simulated, isolated water events. A *compound valve event* is a valve event that occurs while another fixture is using water (*e.g.*, the bathroom sink events in Figure 5-4a). A *collision valve event* is a valve event that occurs within *two seconds* of one or more other valve events (Figure 5-4b and Figure 5-4c). Previous water disaggregation sensing approaches have performed poorly in the face of compounds and collisions (*e.g.*, [158]). This is because compounds and collisions often mask or distort features used for classification. Although a collision is technically also a compound, for the purposes of the analysis I separate them to investigate the individual effect of each on classification performance. In the dataset, 17.2% of all valve events are compound while 4.7% of valve events are collisions (Table 5-2 and Table 5-3). The most common compound/collision events are master bathroom sink opens and closes, comprising 41.8% of all bathroom sink activity and 11% of all valve activity overall (Table 5-3).

Fixtures	Cnt	Total	Hot	Cold	Mixed	Compound	Collision
Kitchen Sink	5	5494 (36.7%)	2438 (44.4%)	1415 (25.8%)	1641 (29.9%)	342 (6.2%)	206 (3.7%)
M. Bathroom Sink	7	3934 (26.3%)	2114 (53.7%)	1294 (32.9%)	526 (13.4%)	1459 (37.1%)	185 (4.7%)
M. Bathroom Toilet	5	1886 (12.6%)	0 (0.0%)	1886 (100%)	0 (0.0%)	87 (4.6%)	117 (6.2%)
S. Bathroom Sink	4	1369 (9.2%)	618 (45.1%)	637 (46.5%)	114 (8.3%)	430 (31.4%)	57 (4.2%)
Washing Machine	4	430 (2.9%)	93 (21.6%)	325 (75.6%)	12 (2.8%)	12 (2.8%)	66 (15.3%)
M. Bathroom Bath	5	423 (2.8%)	224 (53%)	35 (8.3%)	164 (38.8%)	87 (20.6%)	20 (4.7%)
S. Bathroom Toilet	3	341 (2.3%)	0 (0.0%)	341 (100%)	0 (0.0%)	11 (3.2%)	21 (6.2%)
M. Bathroom Shower	5	261 (1.7%)	55 (21.1%)	4 (1.5%)	202 (77.4%)	30 (11.5%)	10 (3.8%)
Dishwasher	3	261 (1.7%)	261 (100%)	0 (0.0%)	0 (0.0%)	9 (3.4%)	6 (2.3%)
M. Bath Diverter	5	228 (1.5%)	17 (7.5%)	1 (0.4%)	210 (92.1%)	92 (40.4%)	5 (2.2%)
Other	1	181 (1.2%)	28 (15.5%)	149 (82.3%)	4 (2.2%)	0 (0.0%)	4 (2.2%)
S. Bathroom Bath	2	59 (0.39%)	5 (8.5%)	0 (0.0%)	54 (91.5%)	2 (3.4%)	2 (3.4%)
S. Bathroom Shower	2	47 (0.31%)	11 (23.4%)	0 (0.0%)	36 (76.%)	0 (0.0%)	1 (2.1%)
S. Bath Diverter	2	46 (0.31%)	6 (13%)	0 (0.0%)	40 (87%)	6 (13%)	1 (2.2%)
Totals	53	14960	5870 (39.2%)	6087 (40.7%)	3003 (20.1%)	2567 (17.2%)	701 (4.7%)

Table 5-3. A breakdown of valve activity by fixture, by temperature state (hot, cold, mixed) and by compound/collisions. The *Cnt* column tabulates the number of fixtures across sites.

With the pressure-based approach, compound valve events result in a dampening and often a severe attenuation of the high frequency component of the pressure transient. As a result, the transient signal is homogenized, making it difficult to classify. With collisions, the two colliding transient waveforms become highly distorted; although it is rarely the case that two transients occur simultaneously (more often they are offset by 200-500ms), the distortions may still render the transient unrecognizable. In **Figure 5-4**b, the shower close and bathroom sink open occur 1.1s apart. In Figure 5-4c, the toilet close and bathroom sink close occur 200ms apart, making it unlikely that both will be classified. I also breakdown the performance of different classifiers based on if the event was a compound or a collision.

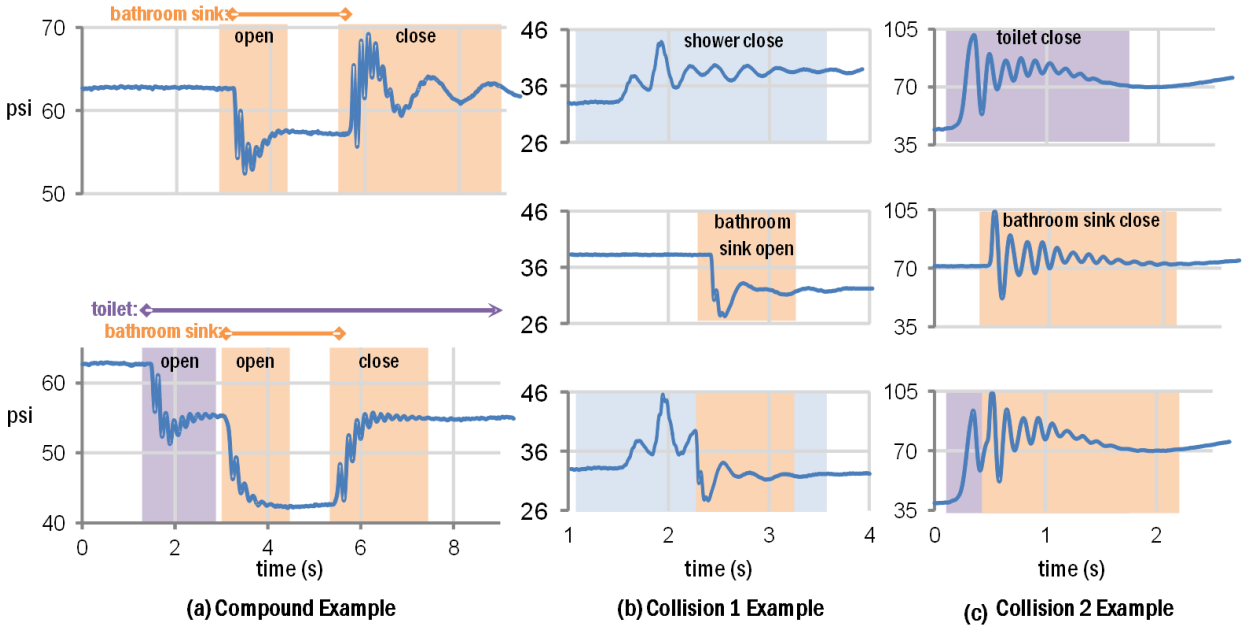

Figure 5-4. (a) Bathroom sink open and close transients occurring in isolation and in compound from H2. (b) A shower close and bathroom sink close transient in isolation and colliding from A2. (c) A toilet close and a bathroom sink close transient in isolation and colliding from H3.

5.6 Summary

In summary, I have collected a ground truth set of labeled examples with which to inform and evaluate supervised and semi-supervised machine learning methods. To the best of my knowledge this is the most comprehensive collection of hot and cold water ever collected and is of immediate utility to a number of communities such as sustainability groups and different utilities.

CHAPTER VI

6. Chapter Six:
Feature Extraction and Sparse Codebooks

This chapter summarizes the feature extraction in HydroSense, which is motivated by the theory of operation for fluid flow in pipes as discussed in Chapter IV. Features are initially selected using expert domain knowledge of water flow through a pipe network. Additionally, I explore the incorporation of features that are derived from a codebook, rather than expert knowledge, using matched pursuit to form sparse feature vectors. These features are selected using unlabeled data and the raw templates from the pressure wave database. This chapter is geared more towards experts in signal processing and engineering. However water sustainability engineers and computer scientists should find most topics approachable and interesting.

6.1 Introduction

In my work with HydroSense I have tried and created several machine learning methods. The initial approaches employed template matching with different similarity metrics. Despite these having good results, template matching does not scale well if HydroSense were installed in 10,000 homes—where one would require hard disk space to save every template for comparison with for each home. Moreover, I have found that the number of templates needed for good accuracy can be somewhat large. From this perspective, the distance calculation is not always computationally efficient. For example, running a matched filter for each template is a costly computation for an embedded device or a cloud computing system. A more efficient, less data storage intensive approach is to only save key features from each transient and save them as an instance vector. Furthermore, working with instance vectors is more generalizable to other applications beyond water sensing, which may not be able to use templates.

I also note that some features will or will not generalize across homes. It might be appealing to choose only features that generalize in order to create one global classifier that works in all homes. I also explore this, but show that such a global classifier is only weakly able to disaggregate water usage. This Chapter

is divided into four sections: (1) non-generalizing template based features, (2) non-generalizing vector features, (3) generalizable vector features, and (4) feature extraction via sparse coding.

6.2 Non-Generalizing Templates

It is interesting to look at how template matching works as a feature for a baseline classifier, despite its disadvantages. In this approach the final goal is to build a library of pressure wave templates and compare them to an unknown pressure wave. We can have any number of template types for each pressure wave in the database, but I have found that six template transformations tend to work well for classification. These templates are four filtered versions of the pressure wave and two transformations. I have already discussed the first three filtered transforms in the previous section—two low pass filtered waves and the derivative of the transient. I also employ another derivative filter that is the derivative of the difference between the two low pass filtered signals (*i.e.*, the derivative of a bandpass filtered transient). This is done to isolate higher frequency resonances because lower frequency resonances tend to be similar for many waveforms. I can formalize the notations by denoting the raw transient signal as $p(t)$, with length of T. Each filter is applied in the frequency domain such that f_X is a low pass filter with cutoff X Hz, and d_X is a derivative filter with cutoff of X Hz (also see Figure 4-7):

- Low pass filter with cutoff of 1Hz: $p_1(t)=[p(t)*f_{1Hz}(t)]_T$, where * denotes convolution and $[]_T$ denotes the aligning and truncation of the signal such that $p_1(t)$ and $p(t)$ are aligned in time after filtering and have the same length, T.
- Low pass filter with cutoff of 25Hz: $p_2(t)=[p(t)*f_{25Hz}(t)]_T$
- Derivative filter with cutoff of 25Hz: $p_3(t)=[p(t)*d_{25Hz}(t)]_T$
- Derivative filter of the bandpass signal: $p_4(t)=[p(t)*((f_{25Hz}(t)- f_{1Hz}(t)) * d_{25Hz}(t))]_T$ where the final filter is calculated in advance from the other filters for computational efficiency.

I also employ two transformations of the signal in the frequency domain. The first is just the magnitude of the FFT of the signal. I truncate the template to frequencies below 25 Hz (25 samples). Mathematically, I denote this as $P(k) =[|\text{FFT}(p(t))|]_{1:25}$. Where $[]_{1:25}$ denotes the truncation of the coefficients from 1 to 25. The second transformation uses Constant-Q weighted Cepstral Coefficients (CQCC). This transformation is a variant of cepstral coefficients that uses filter banks similar to mel-frequency scaling [21]. Each filter in the transform has the exact same quality factor, Q = *center frequency/bandwidth*. This has the effect of logarithmically increasing the bandwidth of the filters as the center frequency increases. There are 16 filters in the filter bank spanning the frequencies of 0.1 Hz to 20 Hz. The constant-Q filter bank transform results in 16 coefficients proportional to the energy in each filter. Each filter bank output is calculated from windowing the input signal $p(t)$ in the time domain with

windows of decreasing length for higher frequencies. The length of the k^{th} window is given by $N(k)=Q\, f_s / f_k$ where f_k is the center frequency of the filter bank and f_s is the sampling frequency. Q is the quality factor, which can be shown to be the integer number of cycles at the frequency k. Therefore each window becomes $W(k,t) = 0.54 + 0.46\cos\left(\frac{2\pi t}{N(k)-1}\right)$ which is a variant of the hamming window. The coefficients are then calculated as [22]:

$$q(k) = \frac{1}{N(k)} \sum_{t=1}^{N(k)-1} W(k,t)p(t)e^{-j\frac{2\pi Qt}{N(k)}} \tag{6.1}$$

Denoting the vector of constant-Q transform coefficients **q** and the vector of CQCC's as **s**, the relationship can then be defined as defined as $\mathbf{s} = \mathrm{DCT}_2(\ln|\mathbf{q}|)$, where DCT_2 denotes the type-II discrete cosine transform and ln is applied element wise.

CQCC have the advantages of reducing dimensionality (there are only 16 coefficients from the filter bank) and the space has approximately diagonal covariance, a common property of the DCT. Another compelling argument for using the constant-Q filter comes from the way that resonances propagate in pipes. At higher frequencies, the bandwidth of the resonance becomes wider (see previous section) meaning that one should sample the energies in the frequency domain non-linearly with increasing bandwidth at higher frequencies. The constant-Q filter bank captures this aspect. Also, higher frequency resonances die out more quickly, so the decreasing window size helps to capture the magnitude of the resonance without averaging it with the portion of $p(t)$ where the resonance had already died out. Furthermore, another argument for the use of Cepstra is that the electrical analogy of the plumbing system is linear. In a linear system, Cepstral coefficients tend to separate the excitation source from the system transfer function [118,131]. The transfer function from the valve in use to the sensor provides great discriminative power, which is captured efficiently using CQCCs.

Therefore the final templates can be defined as (employing a slight abuse of notation for k):

- Truncated FFT: $P_5(k)=[|\mathrm{FFT}(\, p(t)\,)|]_{1:25}$
- Constant-Q weighted cepstral coefficients: $P_6(k)=\mathrm{CQCC}(\, p(t)\,)$

When computing the distance between templates, the filtered templates, p_1-p_4, are not aligned so a matched filtering approach or aligned euclidean is appropriate to use. P_5 and P_6 are already aligned because the magnitude is taken when converting to the frequency domain (all phase information for alignment is lost). Therefore Euclidean distance or the correlation coefficient is more appropriate to use.

In practice, CQCC can have slight changes depending on the alignment of the windows, but I found these issues to not significantly affect the CQCC values in the application.

6.3 Non-Generalizing Vector Features

There are several features that I use in the vector feature extraction. However, each of the features is meant to summarize the transient into a single scalar. In total I use 24 scalar features. Each feature is explained below in two ways: (1) the intuition behind its inclusion, and (2) its implementation.

Pressure Drop: The pressure drop, as described earlier, is related to the flow rate of the valve. For a signal with a relatively constant static pressure and relatively stable pressure the pressure drop can be calculated as the pressure at the beginning of the transient minus the pressure once stable, $p(0)$-$p(T\text{-}1)$. However, this is rarely the case in practice. The static pressure is almost always fluctuating to some degree, especially in apartments or homes that do not have regulators. Compound events also have activity on the pressure line surrounding the transient, which makes specific calculation problematic. The most common problem, however, is that the water usage duration is short. The transient never has time to completely reach equilibrium before another pressure wave is generated. Therefore it is better to "fit" a pressure step to the transient response. I do this in three ways: linear fit, linear fit with don't care region, and exponential fit with don't care region. First I fit a step to the entire transient using linear regression to find (a) the time where the step up or step down occurs, and (b) the magnitude of the step. First I guess a time, t_0, when the step occurs. I then use an input time matrix with two rows: the first row is a vector of zeros up to t_0 and ones after t_0, the second row is a vector of "ones" up to t_0 and "zeros" after t_0. I denote the matrix as $\boldsymbol{X}$, where t could be used to index each column. This can be used to solve a system of equations $\boldsymbol{X}\,\mathbf{a} = p(t)$, where $\mathbf{a}$ is a column vector with two elements. The difference in the elements of $\mathbf{a}$ is the step magnitude. I use the normal least squares solution of $\boldsymbol{X}\,\mathbf{a} = p(t)$, namely, $\mathbf{a}=(\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})^{-1}\boldsymbol{X}^{\mathrm{T}}\mathbf{p}$.

I also implement this regression by enumerating a "don't care" region in the regression where the mean error is not calculated. I use this region because I do not necessarily want to bias the regression with areas where the transient has a large dynamic range. It does, however, limit the amount of data for which I can calculate a fit. I limit the "don't care" region to be less than half of the entire transient response. For each time t that I calculate the linear regression, I also add "don't care" regions around t_0 that eliminate 5, 10, 15, 25, and 50 percent of the surrounding transient (Figure 6-1). Lastly I repeat this calculation using an exponential fit. The rationale behind this fit is that some regulators have longer responses that appear exponential. This typically happens when the regulator to the home has been damaged or is in need of servicing. In my experience, this is a fairly common problem seen in about 15% of all homes surveyed (N=28).

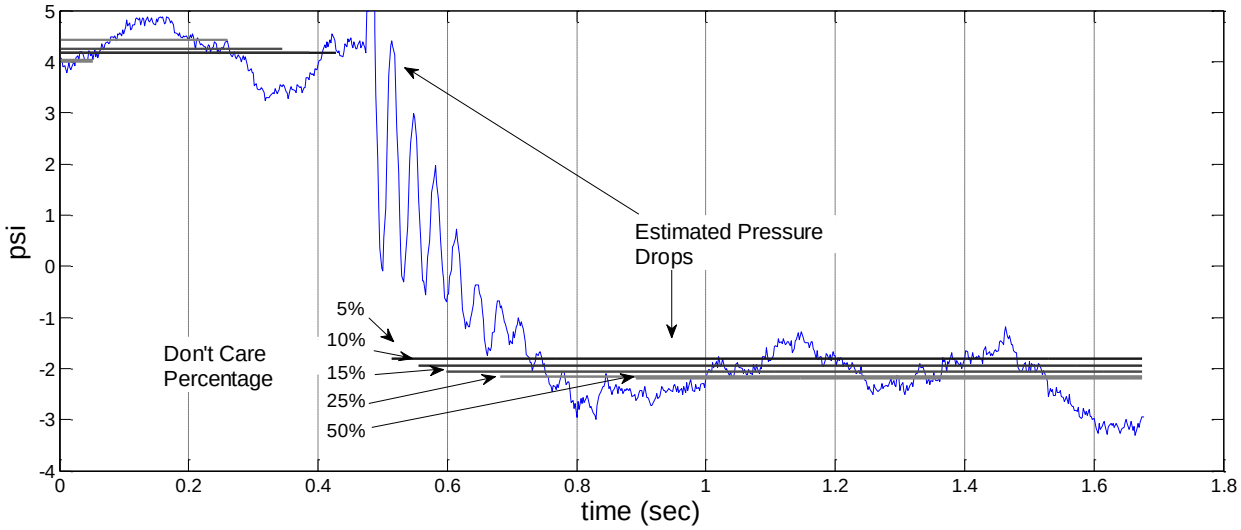

Figure 6-1. Example of the step change regression using 5 different don't care regions. Each grayscale line represents an estimate of the beginning and ending pressure of the transient.

Max and Percentile Amplitude, p_2: The idea here is to look at the amplitude chnge in the transient, which may be more than the pressure drop itself. The calculation is simple max $p_2(t)$ – min $p_2(t)$. In place of min and max, I also try using the 90th percentile – 10th percentile so that the features will not be overly sensitive to spurious jumps in the waveform.

Max and Percentile Derivative, p_3 **and** p_4: Similarly the idea here is quantize the maximum rate of change of the transient waveform, max | $p_3(t)$| and max | $p_4(t)$|. I also take the 90th percentile to prevent spurious max values.

Time Constant: The wave in which the resonances of the signal die out are very important. Usually there are only one or two resonances in the signal and the derivative filter usually isolates one of them. I take advantage of the isolation to find the time constant of each. The time constant of resonances is taken from the derivative filter, | $p_3(t)$|. I first isolate the local maxima (*i.e.*, peaks of the sine waves) using a maximum of a sliding window such that all local maxima are in the set $\hat{t}$, which is a subset of t. Whenever the maximum within the window and the signal value are equal, I assume the value is a peak of a sine wave (Figure 6-2). I then fit an exponential to the decaying maxima points, $\left[|p_3(t)| = \mathrm{A} \cdot \exp\left(-{}^{t}/_{\tau}\right)\right]_{t\in\hat{t}}$.The regression is done in log space such that linear regression can be utilized and the inverse slope of the line is actually the time constant of the exponential, $\left[\ln(|p_3(t)|) = -{}^{t}/_{\tau} + \ln(A)\right]_{t\in\hat{t}}$. I save this time constant value.

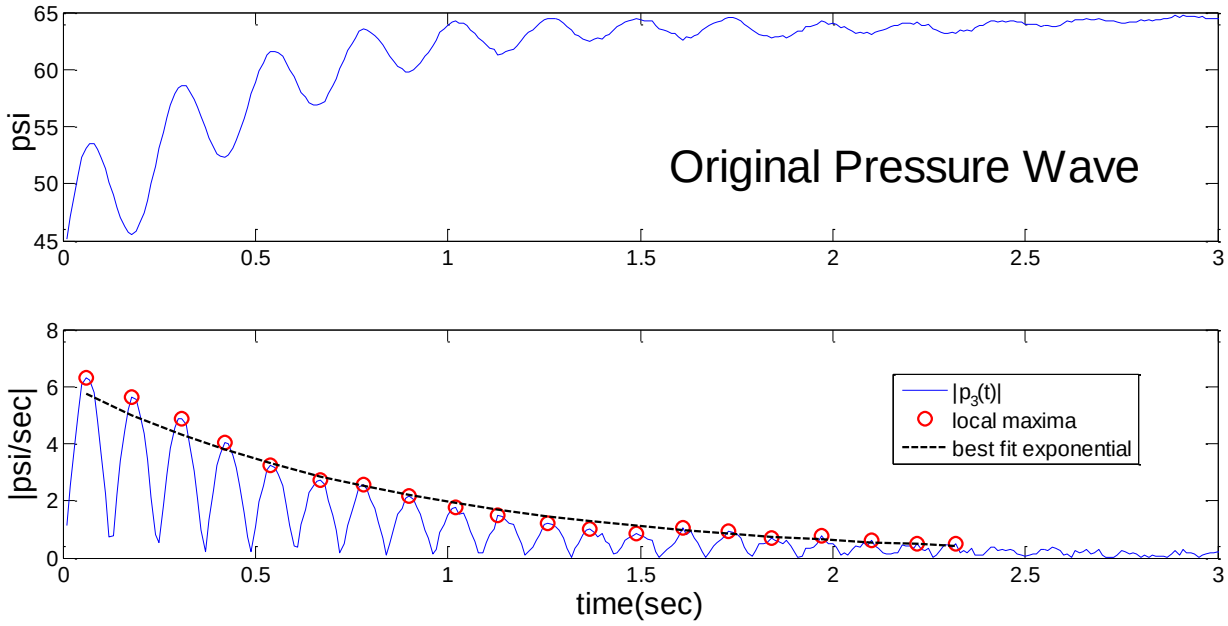

Figure 6-2. Example of processing for finding time constant of higher frequency resonance.

Auto Regressive (AR) Estimate, Resonance Angle and Magnitude: I take an AR estimate of the transient wave (order 4) and find the pole with the largest magnitude on the unit circle. This magnitude on the unit circle and its angle are saved. These correspond to the actual frequency of the most prominent resonance excited and its magnitude in the transient. To prevent over-fitting at high frequencies I down-sample the wave to 25 Hz, allowing some aliasing to occur. Though not published widely, this is a common trick in the practice of AR estimation.

Line Spectral Pairs [76,107]: Line spectral pairs are derived from the AR estimate. The LSP estimate has the advantage of having roots only on the unit circle, which is many time less sensitive to noise in the angle of the root location. I convert the AR estimate in two polynomials, $P(z)$ and $Q(z)$. I use the same calculation as in the AR estimate, measuring the largest resonance and magnitude of the symmetric spectral pairs.

Truncated CQCC: I also use the CQCC coefficients from template matching, truncated to the first 5 coefficients. They have the largest magnitudes and can be considered the most important coefficients because of the energy compaction property of the DCT.

Slope before Transient Wave: For each transient, I also use the slope of the static pressure directly before the start of the transient. This is important for distinguishing many showers and toilets from other fixtures. I observed that when a single handle shower is turned off the cold water is opened slightly before being shut off. This is a physical property of the valve—people take hot showers and they need to turn the

handle through cold to get to the off position. In the pressure stream, one can see a very subtle drop in the pressure before the off transient occurs (Figure 6-3). For toilets, when the fill duration is almost finished the floating valve in the reservoir slowly begins to shut the valve as it rises with the water level. Once it gets to a critical position, it bobs up above the water level and closes the valve quickly. In the pressure stream this can be seen as a slow closing of the valve before the quick turn off. Each feature is calculated by finding the beginning of the transient wave using the derivative,$|\ p_3|$. I find the maximum of the derivative (which corresponds to the greatest slope in the beginning of the pressure transient) and then begin tracking backwards until I reach a value that is 10% of the maximum magnitude—this is defined as the transient "starting point." I then fit a linear regression to the raw pressure data in the 50 ms before the "starting point" and save the slope of the line (Figure 6-3).

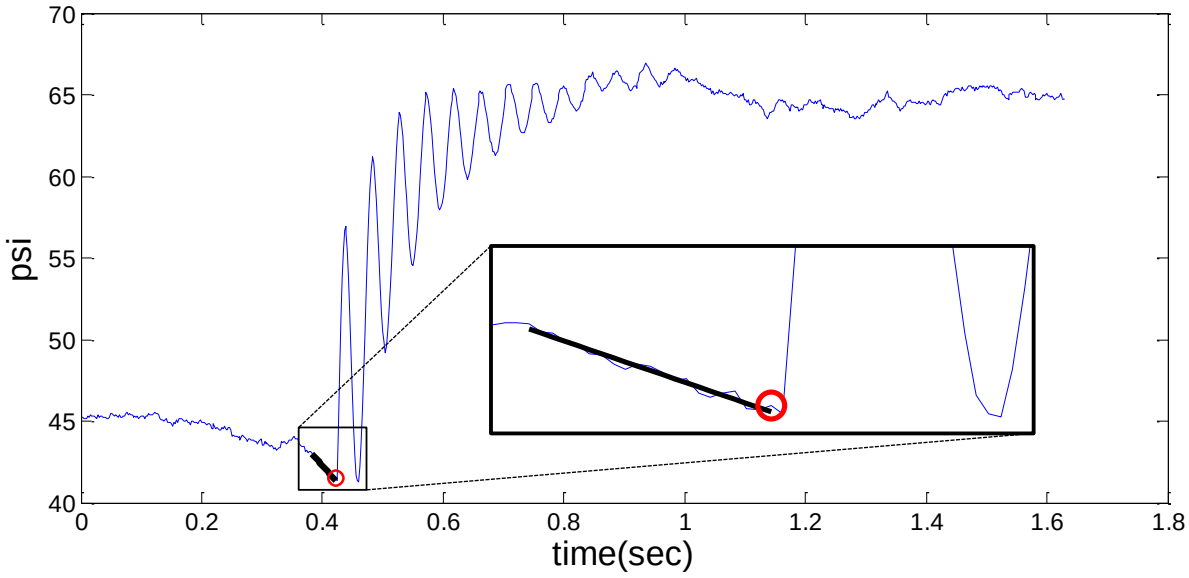

Figure 6-3. Example processing for calculation of slope before transient begins. This is an example from a shower where the dip is highly apparent and negative.

Flowtrace Mode/Max/Mean/Sum: When water is not flowing in the home, I am continually updating the value of the "static pressure" in the home, P_s. This is the psi value at which the home normally sits. It can change, but does so at a slow rate (typically 0.1-0.2 psi per 15 minutes). When water is flowing, the difference between the static pressure and the current line pressure (P_s -p_1) is related to the flow rate of the home. I refer to this difference over a two second interval to be the "flow trace." This is highly similar to the pressure drop calculated before. The difference in this feature is that I look at the difference 2 seconds before the transient occurs. Over those two seconds I measure the mode, max, mean, and sum of the flow trace, which are approximations of the most common, average and max flow rate as well as the volume of water used, respectively.

I note that the static pressure, P_s, of the home requires some additional processing to be sure I do not update the calculation while water is flowing and also to be sure the value does not go above a certain magnitude. I typically restrict P_s to grow no larger than 5 psi above its typical value once the water is turned off in the home for 5 minutes or longer. The rationale here is that the hot water tank, if heating, will many times thermally expand water in the pipes (Figure 4-2). If water is not used, this pressure continues to build, but is not the operating point of the regulator. The max increase of 5 psi, while simple, works remarkably well to keep the flow trace calculation robust to this phenomenon. The only danger in this assumption is that the building of pressure is caused by significant damage to the regulator in the home, not thermal expansion. I do not explicitly correct for this type of pressure increase because many damaged regulators will still regulate to a specific psi value—they simple have more trouble holding the psi value at higher flow rates.

Dual Stream Cross Correlation Lag: Lastly, I now introduce a feature that can only be used if two pressure sensors are installed. The previous features can be calculated for a single pressure sensor only, or they can be calculated separately for each sensor installed in a home. This feature attempts to fuse information from two sensors. I use cross correlation of the two streams to find at what point in time the two streams are most similar. Naively, it might seem like I am calculating the "delay" in signal between two observation points. However, recall from the theory of operation that the entire plumbing system responds *instantaneously* when a water valve is opened. To be sure, there is a physically response delay in the pipes' pressure, but it occurs at the speed of sound and is not detectable by the commodity pressure sensor. So I cannot calculate the signal delay (if I could, this would be a very valuable signal for disaggregation). Even though there is not a lag in the response time, I am observing flow in the piping at two different points—and flow takes time to build up in the plumbing. If the sensors are far enough apart, I can get an idea of how long the flow takes to build up at one point, compared to another. The cross correlation approximates this because the signals will generally appear to have the same shape at very low frequencies (*i.e.*, if I use p_1).

Paired Valve Features: There are also non-generalizing features that apply to paired transients and sequences of transients, rather than just a single pressure transient. These include the volume of water used, approximated by the sum of the flow trace between two transients, and usage duration. These are not guaranteed to generalize across homes, but in many instances they can. For example, similar toilet reservoirs or hand washing is of the about the same duration, *etc.*. Additionally, for a sequence of transients one can look at grammatical correctness (*i.e.*, a close event with no open event or vice-versa) which does generalize across homes. These types of features can be useful for navigating through and re-

sorting an N-best list (next section), but are difficult to apply before some machine learning or inference has already paired transients together.

6.4 Generalizing Vector Features

I now turn my attention to features that have the ability to generalize across homes. That is, I can train a machine learning algorithm in a set of homes using these features and then expect that it will have reasonable performance in a previously unseen home. These features can also be used to train a non-generalizing algorithm.

Open/Close Estimate: I want to estimate if the valve is an open event or a close event. To do this, I look at the sign of the pressure drop or I look at the derivative. If the magnitude of the pressure drop is greater than 1 psi, then I use the *step regression with a "don't care" region*, described previously. A positive sign indicates the valve was opened, a negative sign indicates the valve was closed. If the magnitude is less than 1 psi, then I look at sign of the derivative at its largest magnitude. If it is negative (indicating the pressure is falling) then it is likely an open event. If it is positive (indicating a quick pressure increase) then it is likely a close event. I note that the valve use may also be an "adjustment" to the temperature or flow. In these cases, the valve event is more properly called "opening" or "closing." Rarely, it is both, where the valve is turned from completely cold to completely hot.

Last Pressure Wave Time: This is simply the time since one last saw a pressure wave transient. I measure this in seconds and truncate to be no more than 2 hours.

Last Open Event Time and Close Event Time: Similar to the "last pressure wave time" except the valve must have been assigned an estimate of "open" for the first feature or "close" for the second feature.

Time of Day: This is the time of day (in hours) at which the event occurs.

Weekend/Holiday: This is if the event occurred on a weekend, which can be particularly useful for laundry machines, which are slightly more likely to be run on the weekend.

Evening/Morning: This is an explicit feature that codes whether the event happens from 6AM-10AM (code = 1), whether the event happened between 5PM and 9PM (code = 2), or neither (code = 0). In theory, this feature is not capturing any new information than Time of Day. However, because the morning and evening are semantically meaningful distinctions (*i.e.*, water use in the morning is related to getting ready; the evening is for cooking, etc.), it may be easier for a classifier to condition upon a integer feature rather than learning the differences from the Time of Day feature.

Is a Compound Event: This is an estimate of if the event occurs in compound with another fixture. I use the "open/close" estimate in assigning this value. In the dataset (explained later on) I observed that almost all compound events were between two fixtures. Only rarely did I observe three or four fixtures running simultaneously. Therefore I only consider the effects of two concurrent fixtures (*i.e.*, four pressure waves). For two fixtures, I only need to look at the two previous transients to know whether the current event is compound (the final pressure wave will never be compound). The current event is compound:

- if the current event is an open event and the previous event was an open
- if the current event is a close and the previous two events were open events

All other combinations signify that the valve use is in isolation. This does not work in the rare case that more than two fixtures are running concurrently.

These features can now be incorporated with a machine-learning algorithm in order to classify different sequences of valve events.

A note on using vectors versus raw pressure templates: Working without raw pressure templates can be highly advantageous because it requires a smaller memory footprint and data storage, making it more amenable for cloud computing. For example, a raw pressure wave might be several hundred points long, whereas the feature vector I use is only 30 points. However, it also makes the features extraction easier because I do not need to compare an unknown template to every template in the library (which can be quite computationally costly, especially when using a matched filter). From a design perspective, the feature extraction could be offloaded to the embedded device in the home and then only the features of the pressure transients sent up to the cloud. This reduces network traffic, which could enable the technology for a variety of users with limited bandwidth connections. It also may reduce the cost of a cloud "service" because of the reduced computational time and data storage, reducing the number of servers needed, possibly to the point where the cloud service is free. The only caveat is that the embedded processor must be able to handle the feature extraction. This may at first seem worrying, but practically it is not. A cloud service requires a WiFi enabled device (*i.e.*, a router) that already has a host of computational power that remains largely idle. For example, even bottom of the line routers have typical speeds above 500 MHz and DRAM sizes greater than 128 MB (again, this remains mostly idle) [170]. With this computational power, it is likely that no additional hardware would be needed for feature extraction.

6.5 Features over time in collected dataset

Figure 6-4 shows the values of a single feature over the duration of the study (February 1st to March 6th). Values are shown in blue markers, the best fit line is shown in black and the Spearman rank order correlation coefficient (SROCC) between time and feature value is shown. Weekends are marked with a dashed box. The feature selected is one of the line spectral frequencies and the residence is H1.

The correlation between the feature and time is much higher than chance. There appears to be a general shift in the magnitude and angle of the AR coefficients for all fixtures in the home over time, which is clearly captured by the LSF. The SROCC for other factors such as minimum, mean, and max daily temperature are 5%, 14%, and 12%, respectively. **Implication**: This graph is meant to show that features can change drastically over time. A classifier built with data from the first of the month may not be appropriate when used at the end of the month. Although the relationship here is clearly monotonic (and possible linear), the relationship is not always as distinct and predictable. For instance, behavior change around seasons and events may drastically affect the state transitions learned by a time series classifier. Work on the water mains and infrastructure upgrades can affect a regulator's performance—and thus changing pressure drops, transient magnitudes, and time constants.

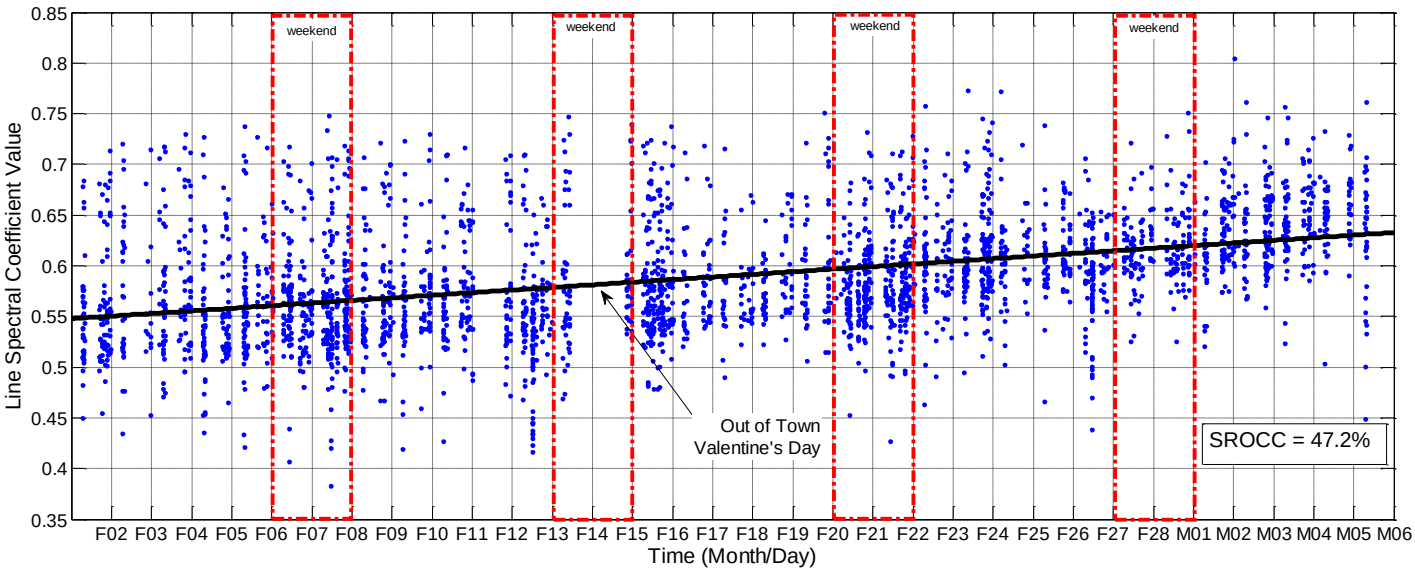

Figure 6-4. Plot of feature value for a line spectral coefficient over the duration of the 5-week deployment.

Figure 6-5 shows the average SROCC across all homes for four different features discussed in section 5.1. Other features with correlations exist, but have lesser average correlations than the ones summarized here. Clearly the percentile derivative features are related to each other because of similar processing and the LPC/LSP features are also related, measuring similar quantities. The error bars shown are the

minimum and maximum values across the residences. For each feature, averages are shown of the SROCC between the feature and time, the minimum daily temperature, the mean daily temperature, and the maximum daily temperature. For each feature, the correlation to time is always more than any correlation to temperature. The dynamic range of the correlation is also large. Some residences have highly correlated features over time, and some have almost no significant correlation at all. **Implication**: the main conclusion I draw from this is that periodic retraining of the classifier would be appropriate, possibly even indicated. It is uncertain how many features and for which homes this retraining would be most beneficial. Additionally, predicting the change in features may not be as predictable as it appears temperature is not the main culprit in the correlation over time (although could explain some of the correlation). Moreover, this analysis only looks at features changes, not behavioral changes. For example, interaction with the ground truth system could affect the way in which individuals use water—however I am less inclined to believe that this explains the correlation dependencies in the features. Clearly this is a physical phenomenon.

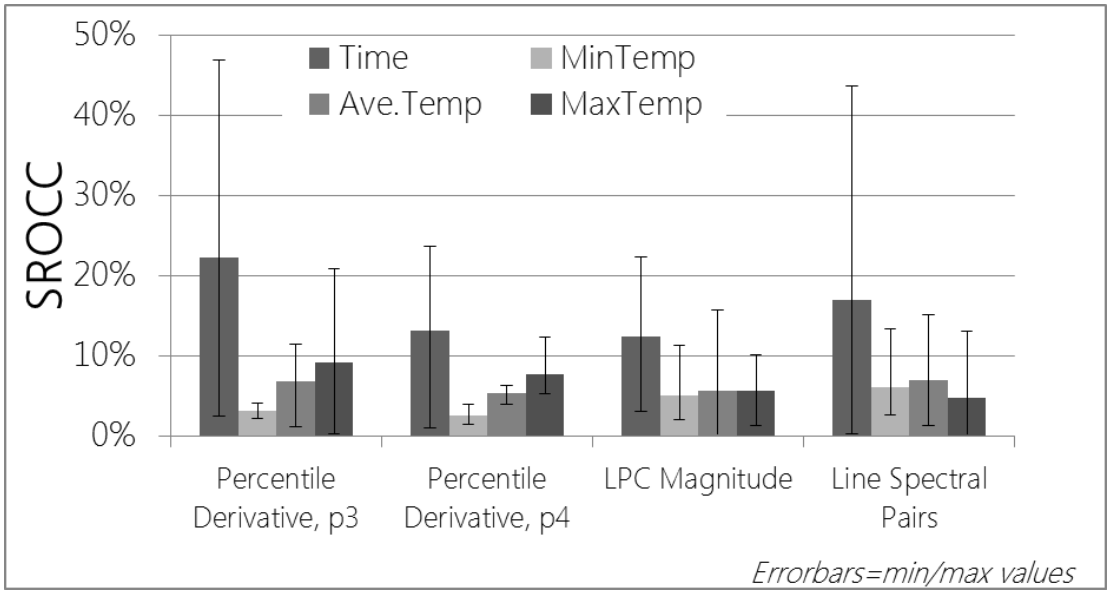

Figure 6-5. Average SROCC of four features compared to time, minimum daily temperature, maximum daily temperature, and mean daily temperature.

Because the features are changing over time, acquiring labeled data becomes exponentially more time consuming. For example, a selective journaling methodology may become more and more inaccurate as I get further in time from when calibration occurred. In the worst case, I need ask for examples when the system is coming out of calibration. Knowing that the system is out of calibration, however, is a difficult task. Later on, I will show that the use of semi-supervised learning (in particular, using multiple views) can begin to tell the system when pressure waves look dissimilar to the labeled examples, which indicates

that a new calibration is needed. Even so, I note that most of the expertly chosen features do not display any correlations with time or temperature. With that in mind, periodic retraining may not be an important aspect of the HydroSense system.

6.6 Codebook features via sparse coding

I now turn my attention away from using expertly chosen features and template matching and instead focus on the use of sparse codebook features to represent pressure waves. This approach can be thought of as a hybrid approach between template saving and expertly chosen features. The intuition behind this approach is that we want to form a codebook of short templates (perhaps 1.5s in duration). Then we want to look at the projection of different pressure waves onto the various small templates in the codebook. The codebook can be used to reconstruct the pressure wave using projection weights. The projections, then, become a feature vector that we can use in any number of classification methods. The key factor, however, is that we want these features vectors to be sparse—that is, as many elements of the feature vector should be zero as possible. With this constraint, one can dramatically decrease the complexity that a machine learning algorithm needs to model. For example, sparse features are dramatically easier for a support vector machine to form class boundaries upon [74] and the number of support vectors needed can be low. CRFs also can take advantage of sparse feature sets quite efficiently. This methodology is also known as sparse coding and has similarities to the problem space of compressed sensing [5,26].

6.6.1 Finding a sparse code

To form a sparse codebook of features I adopt the methods of Leifeng *et al*. [16], who use a technique known as Orthogonal Matching Pursuit (OMP) with K-SVD [3]. They use OMP to find a sparse representation from an existing codebook. After finding this representation, they update the codebook to reduce the reconstruction error of the sparse code.

I first construct a code book $D = [d_1, d_2, \dots d_m, \dots d_M] \in R^{H \times M}$ that encodes the information of a pressure wave. There are M separate columns in the codebook, and each column d_m contains H different descriptors of the data. In my implementation I use M=18,000 and H=1,000. The descriptors in each d_m are initialized by randomly sampling the pressure waves of a given residence at key points in time. That is, the first element of each d_m is selected by randomly drawing 1.5s of pressure data from the first 1.5s of the pressure waves in the database. The second element of d_m contains 1.5s of pressure data selected after the first 10ms of each randomly selected pressure wave. The third is sampled after 20ms from the start of the pressure waves, *etc*. This is continued until the pressure waves have been completely traversed in time. In this way, one can imagine that each element of d_m encodes a pressure window with duration of 1.5s, and start time in the pressure wave given by the index of d_m multiplied by 10ms. Note also that each

pressure wave is subtracted by the original pressure at which the wave is occurring. This is to ensure that a projection onto the extracted patch is not inordinately affected by the pressure of the home. It also makes the use of a linear sum of the codewords more appropriate.

Using the code book I now use OMP to select *K* nonzero elements that can be used to represent the entire pressure wave. I choose the value *K*=400. This means that I wish to use a maximum of 400 dictionary entries to represent the pressure wave. However, if the reconstruction error falls below 0.5 psi, the process is ended. For the HydroSense dataset, this typically results in around 30 non-zero elements, but some pressure waves have more than 300 non-zero elements in the feature vector. Formally, this is:

$$\text{argmin}_x \|y - Dx\|^2 \; s.t. \; \|x\|_0 < K \tag{6.2}$$

Where *y* is the actual pressure wave and *x* is the sparsely coded vector. $\|\cdot\|_0$ denotes the zeroth norm (a count of the non-zero elements). This is the same constraint using in compressive sensing where the L1-norm is many times used [26]. The problem of selecting *K* elements is NP-Hard. However OMP gives us a framework to efficiently calculate the sparse code using a greedy algorithm. OMP first selects the codeword, d_m, that best correlates with *y*. OMP then takes the residual of the reconstruction. At continued iterations, OMP finds the next codeword that correlates with the residual error of the current orthogonal reconstruction. The residual is updated and the process continues until I have selected *K* code words. OMP has a number of advantages in terms of its simplicity and the computation time to find the sparse codewords for a given example, and has been shown to work quite well in practice for imaging applications [16,17].

6.6.2 Updating the codebook

At this point the codebook of sparse examples is not necessarily an optimal representation of the pressure waves. The reconstruction error, for example, could still be quite large. To reduce the reconstruction error of the codebook, I employ singular value decomposition (SVD) upon the total reconstruction error of the codebook and sparse representation. This is commonly known as K-SVD optimization.

In K-SVD, the aim is to minimize the total reconstruction error between all of the sparse codes $X = [x_1, x_2, \dots x_n, \dots \; x_N]$ and the pressure waves $Y = [y_1, y_2, \dots y_n, \dots \; y_N]$. The reconstruction error is given by the froebenius norm of the error (*i.e.*, the square root of the trace of the construction error),

$$\text{argmin}_{D,X} \|Y - DX\|_F^2 \; s.t. \; \forall n, \|x_n\|_0 < K \tag{6.3}$$

This optimization is solved in an iterative manner, where first the sparse code is found using OMP, and then the codebook is updated using SVD. For the new codebook, a new set of sparse features is

found, and the process repeats. At each update, each codeword, d_m, is updated sequentially using any observations in X that selected that codeword. The mth codeword update is given by

$$\|Y - DX\|_F^2 = \left\| Y - \sum_{i \neq m} d_i {x_i}^T - d_m {x_m}^T \right\|_F^2 = \left\| R_{\setminus m} - d_m {x_m}^T \right\|_F^2 \qquad (6.4)$$

Where $R_{\setminus m}$ is the reconstruction error without using the specified m^{th} codeword, and x_i are the rows of X. The codebook is then updated according the largest singular values (as summed by the Froebenius norm).

A separate codebook is created for each residence using 9,000 sparse features from the cold pressure line and 9,000 sparse features from the hot pressure line. I note that even though the dictionary of pressure transients D is over-complete it is still a smaller representation than saving all of the pressure transients, as would be necessary in a template matching approach. For example, each codebook is roughly 5 MB of data—barely larger than a few songs on an iPod. Moreover, the OMP algorithm is much faster than running matched filtering across all elements in the template database. However, it still captures many of the same aspects as the template library. Moreover, because the code is sparse it can be used efficiently to train various machine learning algorithms or transmitted over wireless channels efficiently.

The only disadvantage is that the approach requires a number of segmented examples from the home to form the codebook. The segmented examples do not require a label, however, so in practice I can leave the HydroSense system installed for some time, collecting unlabeled examples. When I receive enough examples, the code book can be created and saved.

6.7 Summary

In this chapter I discussed the various features employed by the machine learning algorithms. These included an array of different templates, a set of carefully selected features, and a sparse codebook of features. The main aim was to introduce how features can be extracted from pressure data such that the phenomena are exploited appropriately for a machine learning algorithm.

CHAPTER VII

7. Chapter Seven:
Natural Water Usage Classification with Supervised Machine Learning

This chapter summarizes the performance of different machine learning methods on the HydroSense database. I discuss the implementation of several supervised machine learning algorithms, including different time series classifiers: a classifier that uses only "generalizing" features, a custom classifier that uses methods from natural language processing, various supervised conditional random field classifier implementations, and a traditional hidden Markov model with Gaussian mixtures used to parameterize emission probabilities. I also analyze the performance of vector classifiers: support vector machines, bagged decision trees, and k-nearest neighbor approaches with random sub-space sampling. The differences between classifiers are discussed and the limitations of the supervised approaches are illustrated with final experiments evaluating their performance when only a few labels are available.

7.1 Supervised Machine Learning Methods

Water activity occurs in clusters—the morning routine, cooking dinner, using the restroom, and getting ready for bed. Chapter IV analyzed if different pressure waves were unique enough between fixtures to distinguish them. However, the data from Chapter IV was from staged experiments, not natural water use. This chapter asks:

1. Are the pressure sequences unique enough during real world water usage that classification is still possible? Can a classifier be built that generalizes across homes?

To answer this question, I look at several different supervised classification methods. The main aim is to see how well the different classifiers perform when using real world water pressure examples instead of canned examples. I also investigate how performance is affected when trying to generalize across homes.

Secondly, this chapter asks:

2. Which is better for classification, using a vocabulary of templates, sparse codebooks, or using expertly chosen feature vectors?

The advantage of feature vectors is that they do not necessarily need to be saved, and are many times lower dimensional than a template or a codebook. This is important for the practicality of the approach. An approach based on template matching may need to save hundreds or thousands of templates for a single home. When an unknown event is sent to the classifier, it needs to compare many templates against the unknown template—which is both time consuming and memory intensive, even when using subsampling techniques. Note also that the templates are not necessarily aligned or of the same length so each comparison includes a matched filter, which is many times more time consuming than Euclidean distance.

The sparse code is a hybrid approach between vector and template matching. It needs to save a codebook of several thousand small templates (about 150 points each). By today's standards, this is not very large (about the size of three songs on an iPod). Moreover, once we compare a given template to the codebook, we can throw the template away and keep its sparse code—which typically has only a few hundred non-zero elements. Therefore, even though the feature vector is technically thousands of elements long, only several hundred elements are non-zero—a fact that can be exploited efficiently by a number of different classifiers during their training such as conditional random fields and support vector machines. Moreover, the sparse code can be saved and used to reconstruct the pressure wave later on, which may be important for different applications such as leak detection or detection of regulator failure.

To answer this question, I compare several machine learning methods that use template matching versus using expert features versus using sparse codes.

Finally, this chapter asks:

3. Which is more suitable for classification, predicting from sequences or predicting from single vectors? How much improvement does one have over the other in terms of classification accuracy and the number of labeled vectors required for training?

This core question I want to answer here is what does sequence classification really afford to the system? There are a number of downsides to predicting sequences of water use versus a single pressure wave outright. For instance, it generates lag in the system. I need to wait for a full sequence of water use (possibly an entire day) before predicting the labels. This is not a wholly detrimental concern however. I can, of course, use a sequence classifier in an online mode, where I decode part of a sequence and update

the decoding beliefs as more instances fill out the sequence. However, sequence classifiers may require more labeled data to properly train. Because labeled data is scarce, this is potentially a real problem.

To answer this question, I compare several vector and sequence classifiers with varying amounts of data. The results of this experiment also motivate the use of semi-supervised learning approaches, discussed in the next chapter.

I implemented several classifiers under varying approaches. I describe each classification approach in turn. For sequence classifiers, the segmentation of different sequences is chosen based on time. For all classifiers, except the Bayesian classifier, one day is considered a single sequence. The reason for this is because the Bayesian algorithm works by pruning an N-best list over time. As the sequences grow in length, this pruning cuts off more and more of the possible sequences and is therefore slightly more affected by larger sequences. To mitigate these problems, sequences are made smaller: if no water activity occurs for 30 minutes, then a new sequence is created. Also, recall that the events are already segmented from the pressure stream so no algorithm needs to classify a "start state" or "end state" as is needed in many speech recognition systems.

The following sections describe the implementation of each classifier listed below. However a short summary for each classifier is given as a "quick reference."

1. *Template Matching, nearest neighbor* **(TM-KNN)**: this approach uses a dictionary of raw pressure waveforms and transformations of the waveforms. It selects the most likely event using the K nearest neighbors, where K=3.
2. *Template Matching, with language model* **(TM-LM):** combining language models and template matching for predicting sequences. It builds upon the template-matching algorithm and combines it with techniques similar to Bayesian inference inspired by natural language processing (NLP) and speech recognition.
3. *Decision Trees with bagging* **(TB)**: this implementation uses Breiman's Random Forest algorithm using expertly chosen vector features.
4. *K-Nearest Neighbors* **(KNN)**: the traditional k-nearest neighbor algorithm using the expertly chosen vector features (*not* the raw pressure templates). K=3
5. *K-Nearest Neighbor Ensemble* **(KNN-ENS)**: this uses an ensemble of KNN classifiers where each classifier is created by randomly sampling the labeled data to compile templates. This is also known as random sub-space sampling.
6. *Support Vector Machine* **(SVM)**: this is a traditional support vector machine classifier using the one-versus-one multi-class implementation.

7. *Conditional Random Field* **(CRF)**: a linear chain conditional random field with feature functions for between state transitions and logistic regression used to parameterize the emission probabilities.
8. *Dynamic Bayesian Network* **(DBN-EM)**: a linear chain dynamic Bayesian network (*i.e.*, a hidden Markov model) trained with expectation maximization and using Gaussian mixtures to parameterize emission probabilities. State transitions are parameterized in a bigram model.
9. *Stacking with TB and CRF* (**TB+CRF**): a stacked machine-learning algorithm with a conditional random field that takes expert features as inputs and the class confidences from a bagged decision tree as inputs.
10. *Stacking with SVM and CRF* (**SVM+CRF**): a stacked machine-learning algorithm with a conditional random field that takes sparse features as inputs and the class confidences from a support vector machine trained with sparse features as input.
11. *Stacking with TB and DBN-EM* (**TB+DBN-EM**): a stacked machine-learning algorithm with a hidden Markov model that takes expert features as inputs and the class confidences from a bagged decision tree as inputs.

7.1.2 Template Matching and Language Model Implementation, TM-KNN and TM-LM

In this approach, I apply a probabilistic approach using techniques derived from Bayesian estimation. The approach is inspired by the dynamic Bayesian models used in natural language processing. Instead of recognizing *words*, it recognizes *valve events*. Like many of the Bayesian approaches, it incorporates a language model and grammar, which estimates the most likely *sequence of valve events* and defines permissible *valve event pairings*. This provides robustness against transient deformations that can occur during natural valve usage (*e.g.*, brief water usage events, low-flow, and compounds).

At a high level, the classification algorithm works as follows: First, an incoming water pressure data stream is buffered and the pressure transients are segmented. Second, the segmented pressure transients are each compared to a library of labeled templates using a set of similarity algorithms. Third, a language model determines the likelihood of a given sequence of valve signatures and links *open* and *close* valve events into *paired tuples.* Then, I extract features from these paired tuples and compare them with smoothed probability distributions. For example, by pairing a *bathroom sink hot open* with a *bathroom sink hot close*, one can extract the *duration* of that event and estimate the *total flow volume* used and then obtain probabilities for those features. Finally, the probabilities from the previous three steps are multiplied together for each sequence and the sequence with the highest probability is selected.

I now formally define the TM-LM model for classifying pressure transient sequences. In Equation (7.1) below, let **V** denote the pressure signature template library (*i.e.*, a vector of labeled

pressure transient signatures and their transforms) and **S** denote a sequence of *unknown* segmented pressure transients. Then, using Bayes' theorem, the most likely valve sequence is defined as:

$$\hat{V} = \arg\max P(\mathbf{V} \mid \mathbf{S}) = \arg\max \frac{P(\mathbf{S} \mid \mathbf{V})P(\mathbf{V})}{P(\mathbf{S})} \tag{7.1}$$

The conditional probability term $P(\mathbf{S}|\mathbf{V})$ describes the outcome of the *template-* and *feature-based comparisons.* The prior probability term $P(\mathbf{V})$ describes the likelihood of the valve sequence (using bigrams) and the likelihood of each pairing in the sequence. Note that *arg max* simply returns a specific valve sequence rather than a probability estimate, thus the normalization constant $P(\mathbf{S})$ can be discarded in practice. One can expand the numerator of Equation (7.1) to further highlight the four major components of the approach:

$$\overbrace{\underbrace{\prod_{r=0}^{R-1} f_r(\hat{\mathbf{S}}_r|\hat{\mathbf{V}}_r)}_{\text{(i)templates and signal features}} \underbrace{\prod_{n=0}^{N-1} P(v_n|v_{n-1})}_{\text{(ii)bigram language model}} \underbrace{\prod_{i\notin\beta} f_p(v_i)}_{\text{(iii)grammar}} \underbrace{\prod_{k=0}^{K-1} \prod_{\langle a,b\rangle\in\beta} f_k(\langle v_a, v_b\rangle)}_{\text{(iv)paired valve priors}}}^{P(\mathbf{V})}}^{P(\mathbf{S}|\mathbf{V})} \tag{7.2}$$

$P(\mathbf{S}|\mathbf{V})$ is now represented by the first term in Equation (7.2), which describes the set of R signal transformations (the templates described earlier) and comparison algorithms (where f_r is the comparison algorithm for the rth transformation). $P(\mathbf{V})$ is expanded into three terms: the bigram language model, a grammar, and water usage event priors. Each term is described in the following. If just using template matching, TM-KNN, then the one simply takes the *argmax* over $P(\mathbf{S}|\mathbf{V})$, term (i). If combining all terms, then the algorithm is TM-LM. I note also that equation 7.2 is no longer a valid probability because it is not guaranteed to sum to one. In my implementation, all the probabilities are converted to log probabilities and summed with the "scores" from other terms.

Term (i): Template- and Feature-Based Comparison: Term (i) compares the segmented unknown pressure transient *s* with *open* and *close* valve templates in the library. Each comparison is broken into two parts: a *signal transformation* on *s* to achieve $\hat{s}$ which is a generic representation for one of the templates described earlier $\{p_1(t), p_2(t), p_3(t), p_4(t), P_5(k)$, or $P_6(k)\}$, and a *similarity score* calculation between $\hat{s}$ and a corresponding valve template $\hat{v}$ in the template library. Each scoring function is represented by f_r in term (i). All of the transformations and similarity measures are used to produce a set of similarity scores for a given valve. These scores are transformed into values between 0 and 1, and multiplied together to form a single template-match score between *s* and every valve *v* in the template library.

I use two similarity scores as discussed earlier for template matching: The matched filter is used for time domain transforms $\{p_1(t), p_2(t), p_3(t), p_4(t)\}$ and Euclidean distance is used for frequency-based transforms $\{P_5(k), P_6(k)\}$. After every $\{s,v\}$ comparison has been made, I reinterpret the similarity scores as probability scores (I use the term "probability score" here because the scores are not normalized, they are simply mapped to values between 0 and 1). For the matched filter comparisons, this is trivial as the matched filter already returns a similarity score between 0 and 1. For the Euclidean distance between each transient in **S** and template in **V**, I define $f_{EucDist}(\hat{s}|\hat{v}) = e^{-|d_m|}$, where d_m is the Euclidean distance between the templates (an interpretation of Euclidean distance as a probability score in log-space, similar to that used in exponential families). I note that I have also tried other distance measures such as the L1 norm and Itakura-Sato [71] but their performance was slightly worse than standard Euclidean distance.

At this point in the algorithm there is an unknown transient *s* and the sets of comparisons for each template in the library. To form a single score for each template, I sum the log probabilities in a set together. These scores are then grouped by valve (*i.e.,* all scores from "kitchen sink open hot" templates are grouped together; all scores from "bathroom sink close cold" templates are grouped, *etc.*). I then take the *argmax* over each valve grouping to find the probability that a particular valve is the originator of *s*.

Because I now have a single probability score for each valve, I can combine the template comparisons with the previously discussed scalar features. I train probabilities for these features by calculating the values for all templates in the library and then using Gaussian kernel density estimation (KDE) [46] to assign probability distributions to each valve in a non-parametric way. This results in a look-up table between feature observations and valve-level probability estimates. These probabilities are multiplied with the template probabilities to complete term (i). Note that when multiple pressure sensor streams are available, such as when two installation points are used, the probabilities for each stream can be multiplied together to form term (i). To incorporate with a language model, I use the best valve probabilities to create a trellis (where each valve type is a separate state). Then, I use the bigram language model to traverse through the trellis, discussed next.

Term (ii): The Language Model: The *language model* assigns probabilities for possible valve sequences. This is performed using bigrams and is represented by term (ii) in Equation (7.2) (*N* represents the length of the sequence). Bigram analysis is commonly used in the statistical analysis of text to examine co-occurrences of words or letters. Here, the bigrams are groups of two sequential valve events; for example, *toilet open→bathroom sink valve hot open* comprises a single bigram. The language model consists of transition probabilities for every valve pair $\langle v_{n-1}, v_n \rangle$ and is trained by counting the number of co-occurring valve pairs in the library. These counts are smoothed using Katz smoothing, which is

commonly used in speech recognition and works to assign a non-zero probability to every sequence [77]. I also employ methods from Gale, *et al.* to ensure that Katz smoothing assumptions are met [30,61]. This is important for handling transition probabilities between two valves that rarely occur in the library.

Traditionally, language models use these transition probabilities to select the optimal word (valve) sequence from all possible word (valve) sequences. I maintain an *n-best list* of sequences using Viterbi-stack decoding [32]. This allows me to dynamically reorder the most probable sequences as new valve events occur. The *n*-best list is restricted to be no longer than 250 sequences so that the computation remains manageable. This can sometimes eliminate a sequence that begins with a very small probability and later on becomes more likely. By keeping the sequences shorter, I can mitigate this effect. Even so, the *n*-best list also allows me to reorder based on secondary knowledge sources—particularly term (iii) and term (iv) in Equation (7.2).

Term (iii): The Grammar: Term (iii) describes a grammar, which is typically used to define a set of structural rules that govern the composition of sentences, phrases, and words in a given language. Here, the grammar defines the possible ways in which valve sequences can be constructed. The grammar rules are: (1) an opening of valve v_x must be followed by a closing of valve v_x; (2) a valve's closure must be preceded by its opening; (3) and the temperature state of a valve must be consistent—*e.g.*, a *kitchen sink hot open* event cannot be closed by a *kitchen sink cold close* event. Rather than eliminating impossible valve sequences (such as a close before an open or an open with no close), I use a *soft grammar* which applies a penalty to any valve sequence that violates a rule. In this way, sequences which contain grammatical errors but have the likeliest probabilities from the other terms can still be selected as correct. The grammar is applied to each sequence in the *n*-best list, resulting in a set of *paired valve tuples* β. In Eqution (7.2), the term f_p penalizes all unpaired valves (those not in β).

These paired tuples in β bind together specific valve open and close events. For example, given the valve event sequence $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow v_4$ where v_1=*toilet open,* v_2=*bathroom sink open,* v_3=*toilet close, and* v_4=*bathroom sink close*, the pairing algorithm might link the two toilet events into $\hat{\beta}_1 = \langle v_1 | v_3 \rangle$ and the two bathroom sink events into $\hat{\beta}_2 = \langle v_2 | v_4 \rangle$. These linkages are critically important because they allow me to compute an additional feature set (term (iv)) that is dependent on knowing the beginning and ending of a water usage event. I note that the language model combined with pairing is a novel aspect of the system.

Term (iv): Paired Valve Tuple Priors: By pairing valve events, I not only have the ability to link open and close transients together but also to compute classification features, such as *water usage duration* and relative *estimates of water volume*, which are not possible without a pairing methodology. For every

paired valve tuple in β, I compute K features over the entire water usage event, denoted as f_k in Equation (7.2). Similar to the *transient features* used in term (i), a probability density is calculated using KDE and the example water usage events in the library. For example, given a particular draw length for an unknown tuple, I can use the usage durations for all kitchen sinks in the dataset to lookup the probability that the usage event is a kitchen sink. Once all paired prior probabilities have been multiplied together, the n-best list is reordered and the likeliest valve sequence is chosen. I use the paired valve priors previously discussed. Namely, usage duration, volume used, max flow rate, most common flow rate, and mean flow rate.

The performance of this classifier is evaluated at the end of this chapter in comparison to other approaches.

7.1.3 Bagged decision tree implementation, TB

I use an implementation of a bagged decision tree that is similar to that from Breiman et al. [20]. The bagged decision tree has been shown to generalize well in a variety of applications [20,87,88,94]. Each decision tree is a classification tree that splits the data one feature at a time. Leaves are merged together based on risk from the parent node and the minimum observations in a leaf are set to one. Each tree randomly gets a set of F features where $F=\sqrt{N}$ where N is the total number of features. 250 total trees are used in the ensemble and observations are sampled with replacement for each tree that is grown.

A final confidence in each class is ascertained by multiplying the confidences of each class from each tree. The class with the maximum confidence is then chosen as the predicted class label. The bagged decision tree is implemented only using the expertly chosen vector features.

7.1.4 Support vector machine implementation, SVM

I use an implementation of a support vector machine from Chan et al [55]. The implementation uses a squared loss term (typically called L2-SVM) and has a cost weight of 0.1. For the multi-class implementation, the one-vs.-one model is used with tournament rounds. That is, each class is compared to another using the between class model and the winners are compared until finally a single comparison is left between two classes [109]. Note that I also implemented a one-vs.-all method but the results take far longer to compute and are about the same accuracy.

7.1.5 K-Nearest Neighbor implementation, KNN

For the k-nearest neighbor approach I used a scaled Euclidean distance measure on the expertly chosen vector features. That is, each feature is normalized by the standard deviation of that feature in the training data. This helps to weight each feature equally in the distance calculation. Ties among training observations are broken using the distance of the smallest observation to the test instance.

7.1.6 K-Nearest Neighbor Ensemble implementation, KNN-ENS

For the k-nearest neighbor ensemble approach consists of 500 different KNN classifiers. Each classifier is comprised of a random subspace of the observations [69] according to the number of features and the number of observations. Five features are chosen for each learner and the size of the KNN vocabulary is set to $\sqrt{N}$ where N is the total number of classes. Observations are sampled with replacement and a majority vote is used to select the final prediction.

7.1.7 CRF Implementation, CRF, TB+CRF, SVM+CRF

For the CRF implementation, I again use the segmented sequence of valve events, $\boldsymbol{S}$. However, no template matching is used in the CRF implementation. Instead I only use vector features. Therefore I define the set $\tilde{\boldsymbol{S}}$ which contains the same pressure waves as in $\boldsymbol{S}$, but can be described as a matrix with a number of rows equal to the number of extracted features, $\boldsymbol{O}$, and columns equal to the number of events in a particular sequence, $\boldsymbol{T}$.

I have given a dense discussion of CRFs previously and now provide the details of the implementation so that others can reproduce these results. Namely, I discuss optimization methods, regularization, the feature function chosen, and stacking with the SVM and TB.

Optimization method: Training iterations require the calculation of a gradient, but how this gradient is used to update the model can be quite different. Because the space is convex, any implementation will eventually converge. I chose to use a quasi-Newton method based on the approximation of the Hessian, limited memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS) [25]. Instead of directly calculating the Hessian (such as in Newton-Raphson), L-BFGS starts using gradient descent, but gradually incorporates an approximation of the Hessian based on the current gradient and past Hessian approximations. Note also that the space is convex, not quadratic. This means more than one iteration is needed for convergence. An exact bound on the number of iterations is impossible to know, but typically is less than 300. This provides a marked decrease in the training time. For ten folds, the TM-LM method requires days to train (including template comparisons), whereas the CRF implementation takes less than an hour.

This optimization is performed efficiently for both sparse and expertly chosen features. If a feature is "sparse" then only non-zero indices are traversed in the calculation of the gradient. The gradient of each sequence is calculated in parallel to the others. Once all sequences in a single iteration are completed the gradients are then summed together to form the final gradient and a line search is performed using golden section search [79].

Regularization: I use regularization on the magnitude of the weights to avoid over-training. Many regularization functions are possible, but I found the L2 norm to work well. Note that I have tested a

number of methods, including the L1 norm, and found them to work marginally worse, on average, than the L2 norm. None of the regularizations were statistically worse based on a two tail T-test at 95% confidence. This speaks to the robustness of CRF training. I use a factor of 10 to weight the regularization term, as recommended by [140]. The final equation for the regularization term is (using the same notation from Chapter III):

$$l_{reg}(\boldsymbol{\theta}) = \lambda \sum_{k=1}^{K} {\theta_k}^2 \tag{7.3}$$

Where the gradient is augmented with the gradient of the regularization:

$$\boldsymbol{\nabla_{\theta}} l_{reg}(\boldsymbol{\theta}) = 2\lambda\boldsymbol{\theta} \tag{7.4}$$

Which can be added directly to equation (3.19), $\boldsymbol{\nabla_{\theta}} l(\boldsymbol{\theta}) + \boldsymbol{\nabla_{\theta}} l_{reg}(\boldsymbol{\theta})$ and optimization can take place as it normally would with the gradient calculation.

Feature functions: The features chosen mirror those from an HMM. Namely, a feature function and corresponding weight exists for each pair of classes in the state space (*i.e.*, the transition probabilities) and for each observation and state pair (*i.e.*, the transmission probabilities). Each observation vector contains 30 observed features (from the non-generalizing features and the generalizing features) and the size of the state space depends on the number of valves in a home, typically 17 (but as high as 60). This means that the total number of weights is 17×17+17×32=833 when I incorporate all the observations. If I only include observations from the generalizing features space (8 observation features), then the number of weights is reduced to 17×17+17×8=425. These can actually both be considered fairly low dimensional problems from traditional CRF applications. Many named entity recognizers that use CRFs can have thousands of features. The advantage comes from a smaller state space.

Stacking: When stacking is employed, the per-class confidences of another classifier are appended to the features for each instance. The CRF is then trained as normal. The advantage of stacking, intuitively, is that it combines the advantages of a decision tree or SVM with the sequence labeling power of a CRF. Decision trees can model complex relationships (with enough training data) except for relationships that occur over time. In the same manner, with enough support vectors, an SVM can model nearly any complex relationship. This is a simple concept, but stacking is a powerful tool employed to solve many difficult problems [49,162].

7.1.8 Hidden Markov Model Based Implementation, DBN-EM

I use a linear chain hidden Markov model trained with expectation maximization and Gaussian mixtures as emission probabilities [72,126]. The implementation is taken from Bilmes and Zweig [11]. The state transition and initial state probabilities are set to uniform values at the start of training. Gaussian mixtures are initialized to have a single mixture and are set to be diagonal covariance. Then Gaussians are then "grown" via a scheduling of splitting and vanishing. The process is as follows:

- Gaussian mixtures are trained with a single mixture until the log-likelihood of the model changes by less than 4% (*i.e.*, 4% convergence).
- All Gaussians in each class are split in half such that the means of each are randomly perturbed by the standard deviation of the Gaussian. Then the model is trained again to 4% convergence. This is repeated four times such that the Gaussians are grown from 2->4->8->16. Spurious singularities are avoided using variance limiting (1e-6).
- After this, the Gaussian vanishing ratio is set to a smaller number, such that if the coefficient of the mixture is less than 700, it is vanished. The model is then trained to 2% convergence. This is repeated 5 times.
- More strict splitting and vanishing is enforced, where a round of splitting is followed by a round of vanishing with a splitting coefficient of 1, and vanishing coefficient of 10. The model is then trained to 2% convergence.
- Finally, the weakest Gaussians are vanished and the entire model is trained to 0.3% convergence.

The splitting and mixing of Gaussians is somewhat of an art, but this process tends to work well for the data. Note that splitting or vanishing occurs between training iterations because the log-likelihood is not guaranteed to increase after an iteration of splitting or vanishing. At the end of the schedule typical numbers of Gaussian mixtures range from 1 to 12 depending on the class and feature.

Stacking: The stacking implementation is identical to that used in the CRF. I only use the bagged decision tree with the DBN as the generative model has a more difficult time using the confidences from the classifiers to improve classification. This might be because there are many dependencies between features that are difficult to model with diagonal covariance Gaussians. Because these confidences are so inter-related, the $P(\mathbf{x},\mathbf{y})$ is difficult to model, whereas the $P(\mathbf{x}|\mathbf{y})$ is slightly simpler because the confidences are incredibly affected by the class value (assuming the vector classifier performs decently).

7.2 Supervised Training Results

Using the various implementations and classifiers above, I am now ready to explore the questions I set out to answer in the beginning of the chapter:

- Can pressure wave templates be used for classification during natural water use? And can a classifier be built that generalizes across homes?

To understand how the algorithms perform at different granularities, I conduct *valve* level, *fixture* level, and *fixture category* level classification. For valve level, the algorithm must identify the correct *fixture* responsible for the pressure transient, whether it is an *open* or a *close*, and its *temperature state* (hot, cold, or mixed). Fixture level ignores temperature state. For the fixture category level, I use the same categories as flow-trace analysis (*e.g.,* [6,44]). The algorithm must correctly classify open/close events as *bath, laundry machine, dishwasher, faucet, shower* or *toilet.* I add the constraint that a *shower* event must include an additional classification as open, close, or diversion. Recall that the dataset includes five residences consisting of three homes and two apartments. I refer to the residences as H1, H2, H3, A1, and A2 to distinguish whether the residence is a home or apartment.

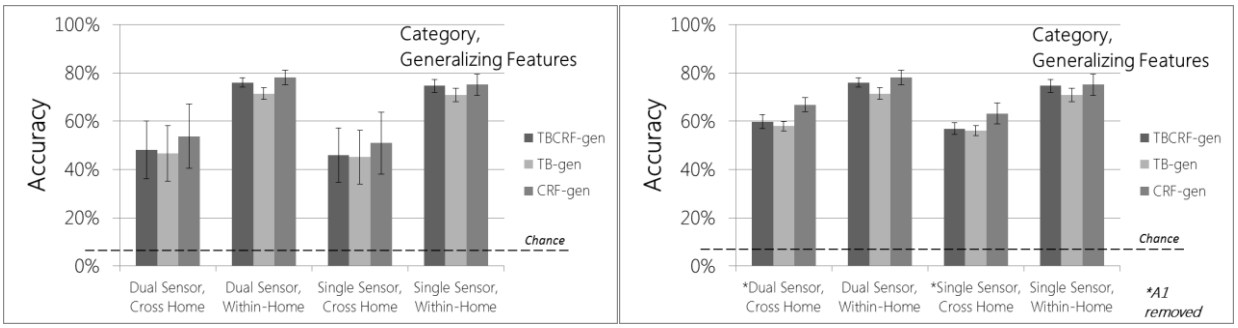

Figure 7-1. Comparison of the generalizing features performance within residence using 10-fold cross validation (an upper bound on performance), and across the residences using leave-one-out validation across the residences.

7.2.1 Generalizing features, upper bound and actual performance

I first turn my attention to building a model that can generalize across homes. To explore this, I use three different classifiers that do not rely on any type of template matching: *TB*, *CRF*, and *TB+CRF*. **Explanation**: Figure 7-1 shows the performance of *TB*, *CRF*, and *TB+CRF* using only features that might generalize across homes. The results are shown at the fixture-category level, meaning that all toilets are lumped into a single class, all faucets are lumped into a single class, *etc*. Algorithms are color-coded and error bars are the standard error across residences. Each series is separated into dual sensor and single sensor, and separated by "within-home" and "cross-home." The within home results are trained using ten-fold cross validation for a single residence and can be considered the "upper bound" of performance of the generalizing features. They are trained and tested on data from the same home. The "cross-home"

results are trained using a leave-one-out-across residence approach. That is, four of the residences are used to train the algorithms and then it is tested upon the left-out residence.

Results: The results upon A1 are not included in the rightmost graph. This was done because the generalized performance on A1 was drastically different from the other homes. A1 never had results greater than chance, while the others consistently performed higher than 50%. The results in the leftmost graph are included for completeness, but the only conclusion that can be drawn is that I can expect some residences to be outliers, with sufficiently different water usage behaviors such that generalization is not possible. It is unclear how often these "outliers" would appear, as the number of residences in the dataset is too small to make generalizations about this observation. The rightmost graph shows that the actual performance across homes is about 10-15%[2] (absolute) less than the upper bounds. The best performing algorithm is consistently the *CRF*, and the improvement is statistically significant.

Implication: It is apparent that using a generalizing model has a significantly diminished performance compared to using labeled data within the same home. The generalizing model is not very compelling. The performance of the generalizing model is only 15-20% greater than the majority[3] classifier, which would have an accuracy of about 40% because of the frequency of the kitchen sink. If I directly monitor the kitchen sink, then a generalizing classifier may have more usefulness. I also note that these algorithms are on the *fixture category* level. The results for the fixture and valve level are hardly worth reporting and are within 5% of the majority classifier. As such, it appears that a generalizing model cannot be trained reliably.

Moreover, training a generalizing model has limited utility. For instance, A1 has patterns of water usage that differ drastically from the other homes and a model cannot be attained that is greater than chance. This is possibly predictable based solely on the demographics of A1—they have an onsite laundry machine, they do not have a dishwasher, use hot water more than they use cold water, and their fixtures are almost exclusively dual handle. This is in contrast to the other residences that have mostly single handle faucets and use cold water more than hot water. As such, a generalizing model may not even be possible.

I therefore turn my attention to using labeled data within a home and using all the features at my disposal (not just features that generalize). I seek to answer if the approach of using pressure waves is

[2] In the entirety of this dissertation, all percentages refer to the absolute percentage improvements, not relative improvements.

[3] Note that this is sometimes referred to as a plurality classifier because it simply chooses the most frequent class, not necessarily something that is greater than 50%.

feasible and whether template matching has a distinct advantage over expertly chosen features or sparse codes.

7.2.3 Aggregate accuracy of algorithms, 10 fold cross validation

To explore the difference in classification between sparse codes, expert features, and templates, I chose two representative classifiers that use each feature set for comparison. For template matching, I chose an instance classifier and a sequence classifier, *TM* and *TM-LM*, respectively. For expertly chosen features I also chose an instance and sequence classifier, *TB* and *TB+CRF*. Finally, for sparse codes I chose the *SVM* and *SVM+CRF*.

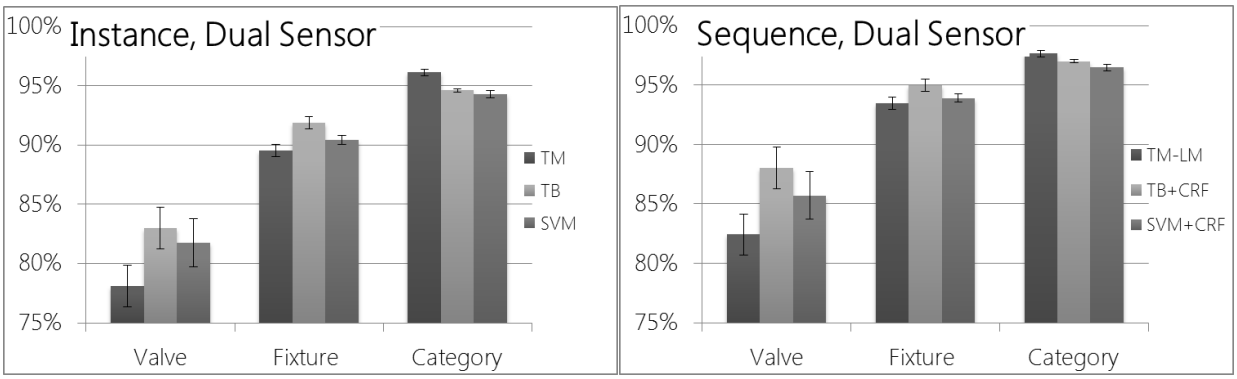

Figure 7-2. Aggregate accuracy of different algorithms across all residents in the dataset.

Explanation: Figure 7-2 shows the accuracy of each classification algorithm on a scale of 75%-100%. Error bars are the standard error across all folds and all homes. They are meant to show the confidence interval in the mean accuracy of each algorithm. Each classification algorithm is represented by a single color and the level of classification separates each series. Two plots are shown, the leftmost uses instance classifiers; the rightmost uses sequence classifiers. **Result**: With instance classifiers, *TB* is consistently better at the valve and fixture levels, but *TM* is the best performer at the category level. This trend continues when looking at the sequence classifiers. *TB+CRF* performs better than at the valve and fixture level, but *TM-LM* outperforms all algorithms at the category level. In terms of statistical significance, I look at a 2-tailed t-test at the 95% confidence level. At the valve level, *TB+CRF* and *SVM+CRF* are statistically the best performers. At the fixture level, *TB+CRF* is statistically the best performer. At the category levels, *TM-LM* is statistically the best performer.

Implication: The first result to note is that all of the algorithms are performing well, above 75% even at the valve level. In terms of overall accuracy *TB+CRF* is always one of the best performers and mainly competes with the *SVM+CRF* approach at the valve level. *TB+CRF* may be the best choice in terms of accuracy at the fixture and valve level, while *TM-LM* is a slightly better choice at the category level. In

terms of features, it would seem that the expert feature set and sparse codes have a real advantage over template matching at the valve and fixture level. For template matching, even though it is significantly the best performer at the category level, all of the algorithms are highly accurate. As such, I can conclude that template matching is not the most ideal approach in terms of overall accuracy. I cannot yet conclude, however, that it is not a viable approach because it may still require less labeled training data than the other approaches—an experiment I carry out at the end of this chapter.

Finally, I can say undoubtedly conclude that sequence classification is preferable to instance based classification in terms of accuracy, regardless of the features set employed. However, I also still need to investigate whether sequence based classification requires more training data.

Another factor to note is that all algorithms are within 70-90% accuracy at the valve level, 85-95% at the fixture level, and 90-100% at the category level. For a system like HydroSense, it may be that these accuracies are acceptable at the category level and fixture level. Recent studies have shown that for everyday context-aware applications there is an assumption that the system will need to be about 80-90% accurate [95]. However, these studies were for non-critical systems. It is likely that the homeowner will consider a system like HydroSense to be critical, especially since it may be related to utility billing and rebate programs. As such, a more appropriate assumption might be to expect 85-100% certainty. From this perspective, a better goal for the accuracies would be greater than 85%. As such, I now breakdown the performance barriers for the different algorithms and assess the major areas for improvement.

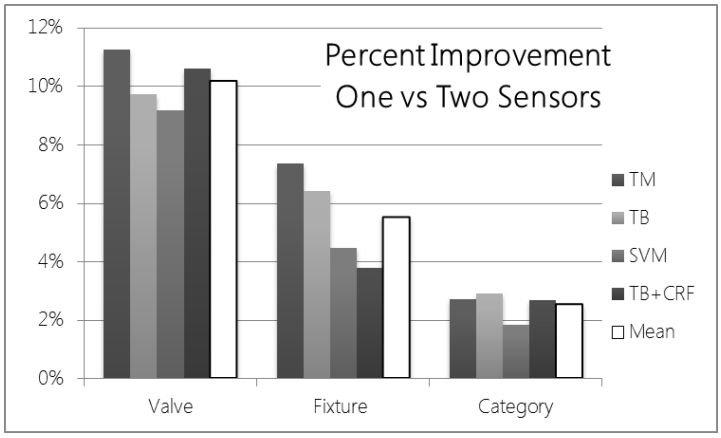

Figure 7-3. A comparison of the percent improvement for adding an additional sensor to the HydroSense system.

7.2.4 Dual sensor versus single sensor

From the previous result, it is apparent that sequence based classification with expert features is a good overall choice of classifier. However, I do not yet know how performance degrades when I use a single

sensor for classification rather than two sensors. Another sensor adds cost and complexity to the system (marginally), and therefore I investigate the relevance of its inclusion next.

Explanation: Figure 7-3 shows the percent improvement, averaged across each home, when using dual sensors, versus a single sensor. Again, series are split by classification level and color coded by algorithm. To simplify the figures and discussion, I chose four algorithms from the previous section, *TB*, *TM*, *TM-LM*, and *TB+CRF*, however classifiers based on sparse features have a similar conclusion. **Result**: All of the within algorithm improvements with a second sensor are statistically significant at the 95% confidence level. At the category level, these improvements are typically about 2% improvement. The valve level improvements are on the order of 10%. **Implication**: The additional sensor increases cost to the HydroSense system and adds a possible barrier to the installation. Thus the improvement in the category level is not warranted. However, the improvements in the fixture level and valve level are warranted, with an average of 5-10% improvement. For instance, without another sensor at the valve level, all the algorithms are below 80%. Although the final design of the HydroSense system should take cost into consideration, the addition of a second sensor greatly improves performance and therefore could drastically increase the confidence a homeowner has in the system.

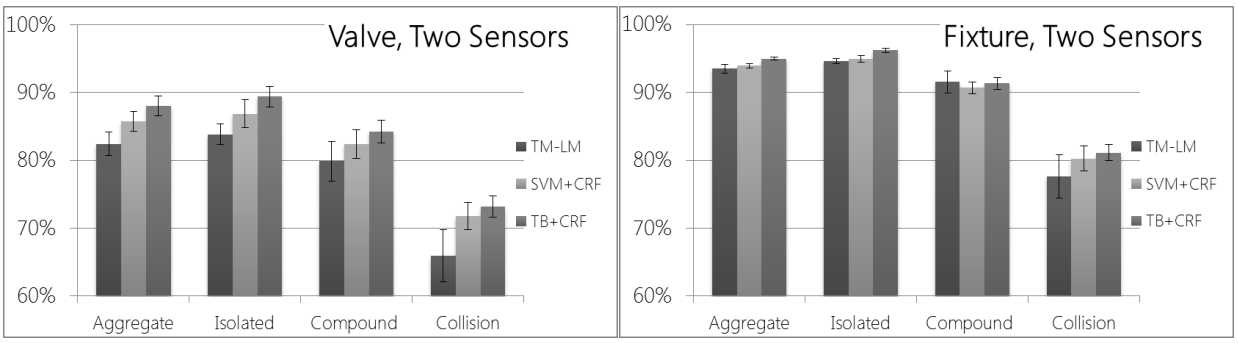

Figure 7-4. A comparison of algorithms across different classes of pressure waves, isolated, compound, and collision.

7.2.5 Performance on compound and collision

I now focus my attention upon how the algorithms perform on pressure waves that are isolated, compound, or involved in a collision (defined in Chapter VI), in order to find points for improvement or potential problem areas. I explicitly look at the dual sensor, fixture and valve level with *SVM+CRF*, *TM-LM, and TB+CRF*, but all results have similar conclusions. **Explanation**: Figure 7-3 shows the accuracy of different algorithms for different groups of pressure waves. Error bars are the standard error across residences. **Result**: All algorithms have a significant decrease in performance going from isolated to compound to collision. The *TM-LM* algorithm has increasing variance in performance going from isolated to collision, while the other algorithms have more consistent performance. Consistently, the best

performing algorithm is the *TB+CRF*. **Implication**: On average, the *TB+CRF* can improve the compound event classification by about 5% in order to bring it on the same level of performance as the isolated case. From Table 5-3, I know that compound event occur about 17% of the time, so improving classification of compounds should have a large impact on performance and be a priority. Collision events occur much less often, about 4.5% of the time. As such, efforts made to increase the accuracy of collisions would likely not result in significant improvements.

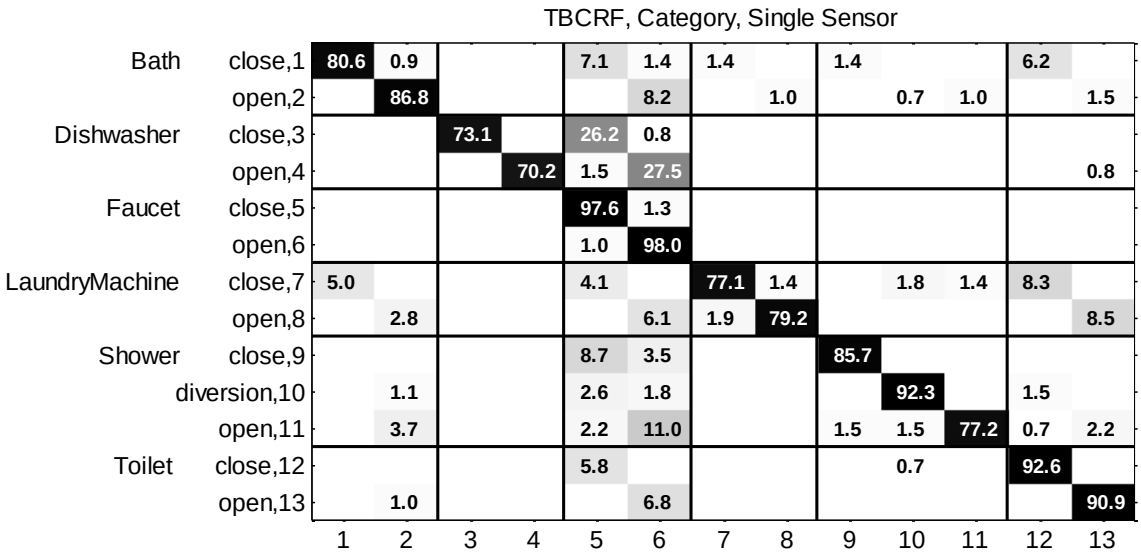

Figure 7-5. Confusion matrix for algorithm TBCRF across all residences, by fixture category.

7.2.6 Confusion analysis

Explanation: Figure 7-5, Figure 7-6, and Figure 7-7 show the confusion matrices for algorithm *TB+CRF* at the category, fixture, and valve levels. The ground truth labels are shown per row, the classified labels are shown per column. Therefore each row will add up to be 100%. Squares with no confusions do not have the percentage labeled inside the square. For each row, the fixture name is labeled, and then subclasses of that fixture are given (*i.e.*, open/close/diversion or Hot/Cold/HotAndCold). There is also a number assigned to each row, which corresponds to the same text label for each numbered column. The dark lines separate fixtures that are in similar portions of the residences (*i.e.*, in the same bathroom, kitchen). Note this cannot be done for the category level because multiple fixtures lie in different rooms. The lighter gray lines separate each fixture. **Result**: At the category level, there are a number of confusions of fixtures as a Faucet. This makes intuitive sense. The kitchen and bathroom faucets are used

more often than any other fixture in the home. The most consistent confusion is the dishwasher as a faucet. This makes sense also, and it is verified by the fixture level confusion matrix.

Result: Figure 7-6 shows the fixture level confusions. I see now that the dishwasher is confused fairly consistently for the kitchen sink. There are also a number of confusions among fixtures in the master bathrooms of the residences and a number of confusions of fixtures in the secondary bathroom for fixtures in the master bathroom. This also makes intuitive sense as many of the faucets in each bathroom are similar fixtures with similar pressure drops and usage durations. The "other" category consists of two fixtures from H2 that the other residences did not have: a refrigerator water dispenser and a laundry basin sink. Finally, note that the S. Bathroom Bath is confused for many other fixtures in the residences. This is largely because the number of examples from the secondary bathrooms was small. None of the families actively used this bathroom.

Result: Finally, Figure 7-7 shows the valve level confusions. The checkerboard patterns in the kitchen and bathroom sinks reveal that the highest number of Hot/Cold confusions occurs in these fixtures and this is the main area for improvement at the valve level. Because these fixtures are used so often, even a small gain in the hot/cold disaggregation for the faucets could drastically improve the aggregate performance of the algorithm.

Implication: the confusions in the home could be reduced by having an additional sensor attached to the kitchen sink. For instance a small flow sensor attached to the cold line of the kitchen sink would help to reduce not only the number of confusions from other fixtures as the kitchen sink, but also reduce the number of hot/cold confusions at the kitchen sink. If the homeowners are willing to install HydroSense underneath the kitchen sink, then the addition of a cheap flow sensor would be warranted.

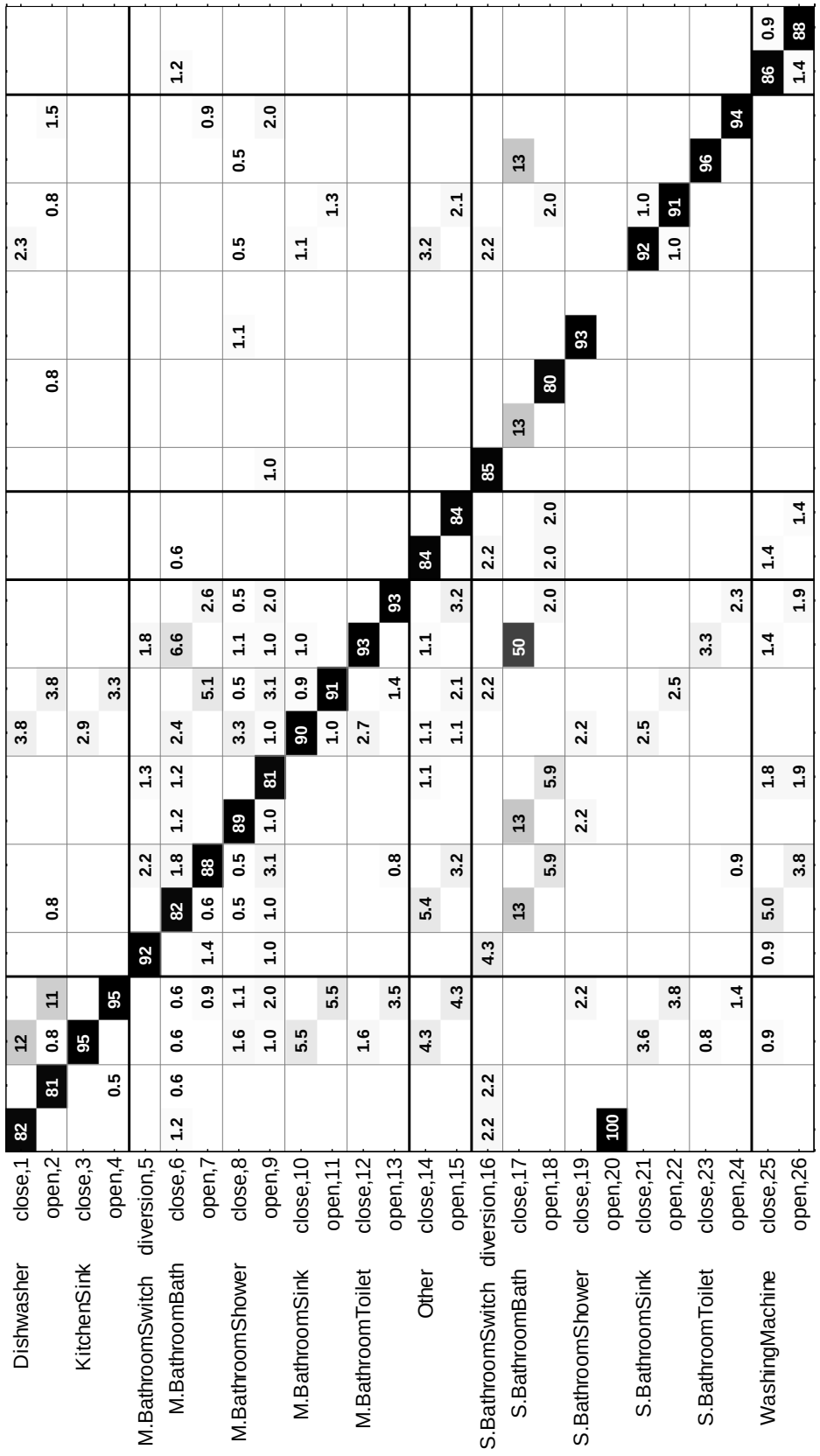

Figure 7-6. Confusion matrix for TBCRF across all residence at the fixture level.

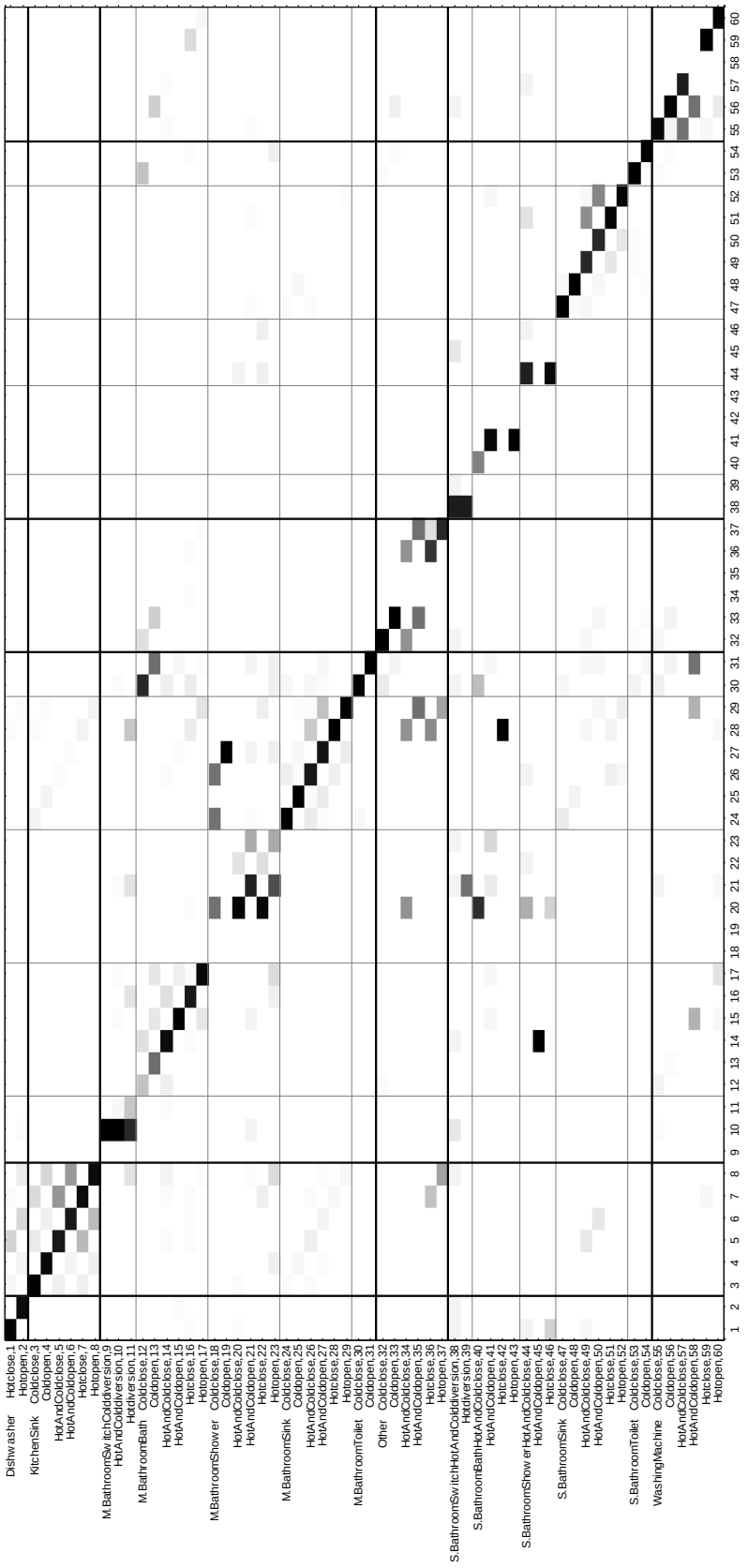

Figure 7-7. Confusion matrix for TBCRF across all residence at the valve level.

7.2.7 Comparison of all algorithms, 10-fold cross validation

Thus far I have begun to answer many of the questions this chapter put forth. I know that template matching has less accuracy than expert features and sparse codes. I also know that sequence classifiers perform significantly better than their instance level counterparts when enough training data is available. I know that a second sensor is a significant improvement and I know that there is room for improvement in detecting compound events. However I do not yet know how different classifiers are able to leverage the features for classifying natural data. Could I be using a simpler *KNN* approach rather than the more complicated *TB*? How does generative versus discriminative modeling work? If I had unlimited training data, would there be any preference for a particular classifier?

In this section I present all of the different algorithms comparatively at different classification levels. I also introduce two new classification levels: *lumped fixture* and *category+temperature*. In the lumped fixture level, I lump together the bath, shower, and diverter from a single bathroom into a single class. The idea here is that confusions within these classes are not detrimental to the algorithm—I still know that a shower was taken. The *category+temperature* level is the same as the category level from before, except I add in the constraint that the temperature of the fixture must also be ascertained as *Hot*, *Cold*, or *HotAndCold*. This could be important for applications such as a smart water heater, which needs to know fixture type and temperature, but not necessarily where in the home it is going.

Explanation: Figure 7-8 shows the results of aggregate level accuracy for each classifier with 10-fold cross validation. Results are presented at different levels as marked via the plot. Results are ordered from the least accurate to the most accurate in each plot to facilitate comparison. **Result**: *TB+CRF* and *SVM+CRF* are consistently in the top three performing algorithms. *TM-LM*, *CRF*, and *TB* sometimes make it into one of the top three, but are always in the upper half. *KNN* and *NN* are consistently the worst performing algorithms. In the middle is consistently *SVM*, *DBN-EM*, and *KNN-ENS*. **Implication**: Interestingly, the *DBN-EM* algorithm is consistently worse than the *CRF* giving rise to the conclusion that discriminative approaches are slightly preferable to generative approaches, at least when using a linear chain. When using stacking, the *CRF* is greatly preferred over the *DBN-EM*. Not shown in the graph are the results from stacking with *DBN-EM*. They consistently were worse than just using *DBN-EM* and many times were the worst performing algorithm overall. The most consistent performer, as has been in previous experiments is *TB+CRF*. Interestingly, however, all the algorithms tend to produce good results. That is, there is always a window of about 5% where one can expect the top performing algorithms to be within, and all algorithms are within a 10% of each other. From this point of view, it would be easy to conclude that the HydroSense system is no different than many other systems: with enough training data

just about any algorithm can produce good results. So I cannot yet discard any of the classifiers in the analysis, with the exception of possibly *KNN*. Instead, I need to now consider the effect of the number of training examples when selecting the algorithm. I also have an upper bound on classification performance at the (1) Valve Level~88%, (2) Category+Temperature~90%, (3) Fixture~95%, (4) Lumped Fixture~95%, and (5) Category around~97%.

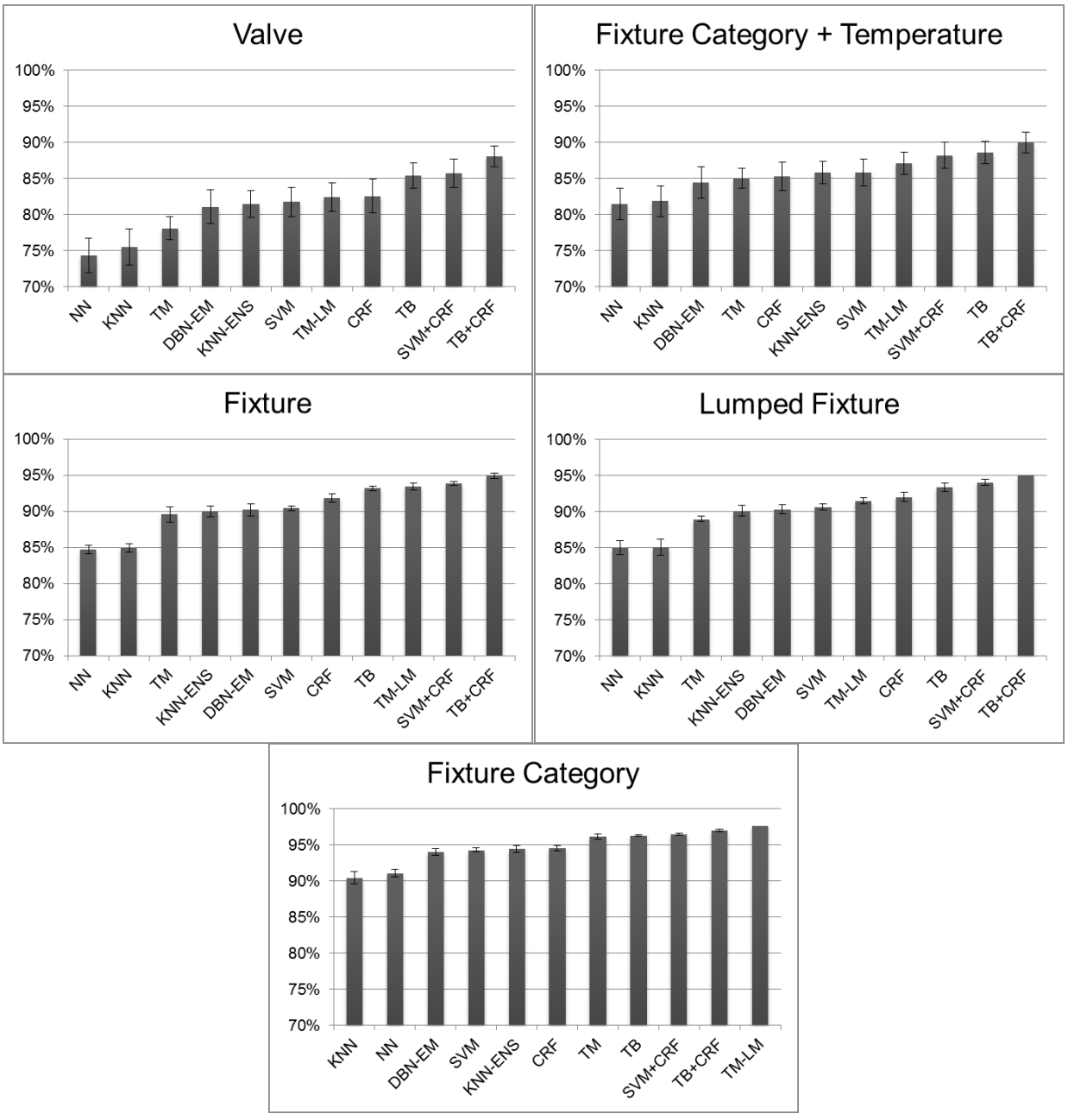

Figure 7-8. Comparison of algorithms under 10-fold cross validation

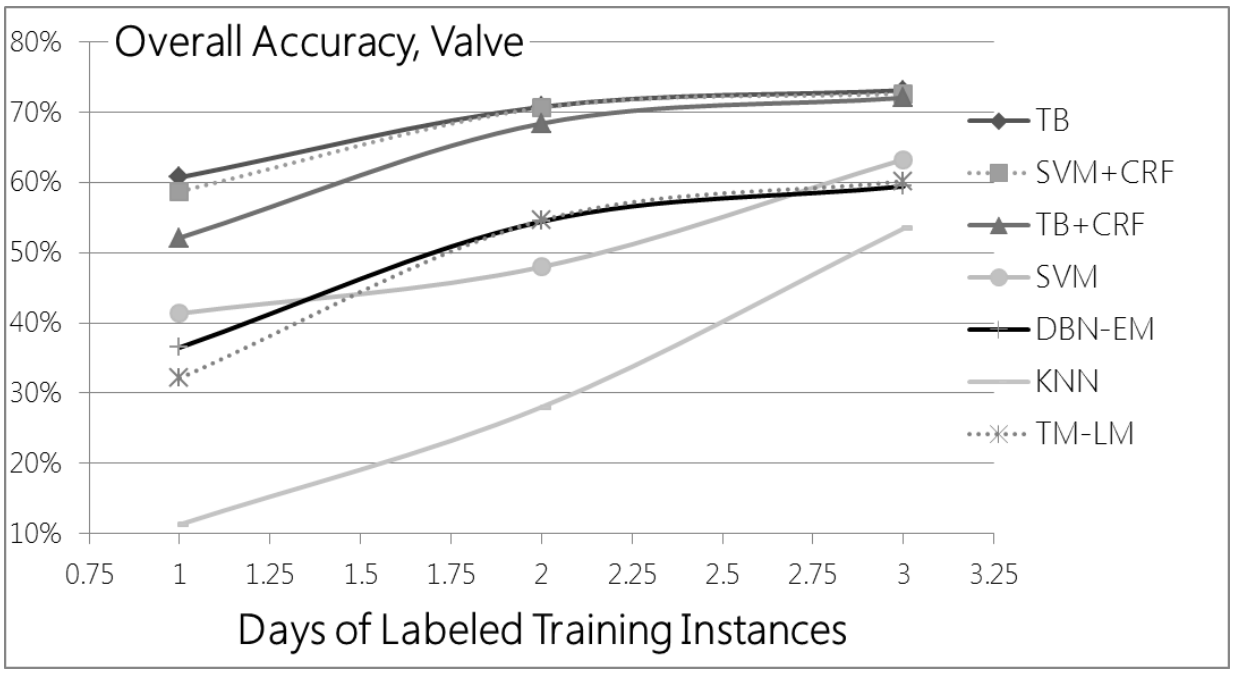

Figure 7-9. Training different algorithms with different levels of training data

7.2.8 Comparison of algorithms, limited training examples

The final and arguably most important limitation of the algorithms is the amount of calibration. All of these algorithms require labels upon pressure waves. The method of collection might be an electronic journal or a smartphone app that asks the resident to label an event when it is detected. My aim, then, is to ascertain how many days of water usage would need to be entered in order to bootstrap the classifier. **Explanation**: Figure 7-8 shows the accuracy of the algorithms as I increase the number of days of training data from 1 to 2 to 3 for all the residences. For each number of training days, 4 random trials were selected and tested. For example, in H1 one day was selected at random for training and the remaining days were used as test data. This was repeated three more times using 3 different days of training data and the results were averaged together. Error bars are standard error across residences and trials. The accuracies for each classifier are connected with a line so that a trend can be ascertained. All accuracies are shown at the valve level.

Result: A representative set of algorithms is shown in Figure 7-9. *TM* and *NN* are not shown, but have similar performances to *TM-LM* in the middle of the graph. Statistically, the best performers are *TB*, *TB+CRF*, and *SVM+CRF* across all days. *KNN* is the worst performing algorithm over the entire set of days. For comparison, the upper limit of accuracy is 88% at ten folds.

Implication: Recall that passing the 80% level has a large bearing on confidence in the system's performance. None of the algorithms come close, even with 3 days' worth of training data. The results of this experiment must be wrestled with the overhead of labeling data in a system. Even the most eco-

conscious person is unlikely to label their usage for more than a day [6,159]. For water end-use studies, it may be feasible to pay a family to journal a week of data, getting clusters of water use labeled such as around the bath and shower, and some kitchen sink activity. This may be the easiest and most systematic way of priming the HydroSense system, but it only has utility in well-funded studies and places a large burden on the residents of a home. For instance recall that in the dataset the average number of events per day ranged from 70-160, all of which would need to be labeled including automatic water events such as in the laundry and dishwasher cycles. From a consumer perspective, labeling a few events during installation might not be a large burden, but continuously logging usage over many days might alienate a large portion of adopters interested in sustainability. With this in mind, I now investigate the performance of ascertaining just a few labeled examples from someone during one day of installation, about half a days' worth of water usage.

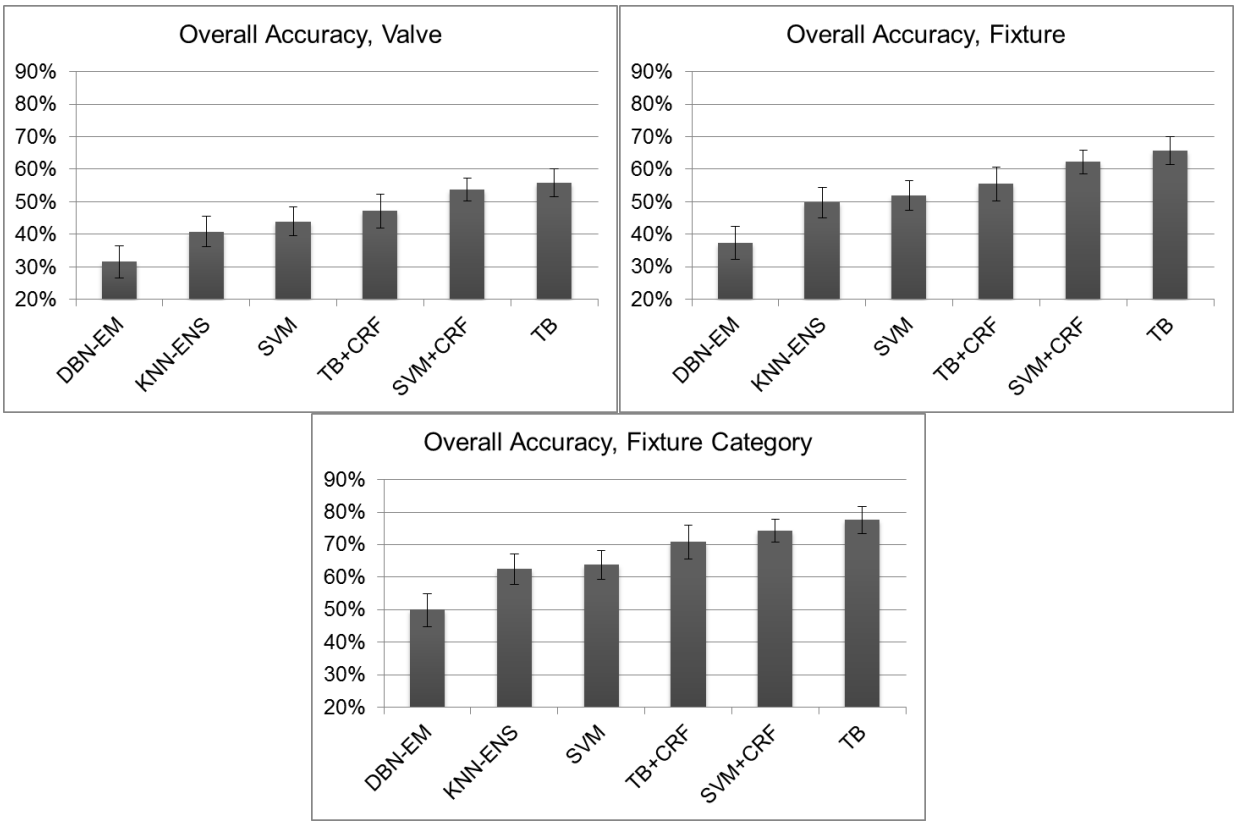

Figure 7-10. Accuracy of different algorithms using minimal amount of labeled data over all classes.

Explanation: Figure 7-10 shows the results of using a few clusters of labels throughout a single day to prime the system. A representative set of algorithms are used. Three different levels of accuracy are shown, valve, fixture, and category. The training data for each algorithm comes from "clusters." That is, in a single day I label each shower event (up to two) and I label any dishwasher or washing machine

events (only a single run). I also label two toilet events per bathroom and one bathroom sink events per bathroom, per temperature. For instance, if a bathroom sink has two handles, I get an example of cold water and hot water. If the sink is single handle, I get an example of cold, then hot, then mixed water use. Finally, I get two examples of the kitchen sink at different temperatures. For the largest home, this resulted in 20 labels from the homeowner. For the smallest residence, this resulted in 8 labels from the homeowner.

Result: The most consistent performer when only a few labels are available is *TB* followed by *SVM+CRF*, and *TB+CRF*. The worst performing algorithm is *DBN-EM* which is only slightly better than the majority classifier at the category level. In each graph, *TB* and *SVM+CRF* are statistically the best performers.

Implication: Despite TB being a consistent performer, the results are fairly lackluster. At the valve level, especially, none of the algorithms are doing better than ~55%. If one looks at the confusions in the home, I see that nearly everything is confused as a faucet. Almost all showers, dishwashers, and laundry machines are incorrectly labeled. The hypothesis is also confirmed that adding sequence labeling is hurting the performance when only a few labels are available. Clearly a semi-supervised approach is needed that incorporates more prior knowledge.

At the fixture category level, about 76% of the events are correctly classified. A number of confusions, however, are occurring from other fixtures being confused as the kitchen sink. If the kitchen sink has a dedicated flow sensor, these could be avoided. I conclude that a dedicated flow sensor on the kitchen sink cold line will help eliminate a large number of hot/cold errors in the home, but automatic labeling would still be required for more fine grained sensing. Even so, if the homeowner is willing to label a few events from each fixture upon installation and a small flow sensor is installed with the system, a system like HydroSense may be consumer ready at the category level. Outside of this scenario, I conclude that the calibration overhead is too great—another method must be used for widespread adoption.

And the bar is set: I need a system that can take 8-20 labeled events and perform better than ~55%. Ideally I could get back to the 88% accuracy at the valve level. Perhaps a more realistic goal is to get within the 80-90% confidence range proposed by Lim and Dey [95].

Although I am establishing a new training methodology with semi-supervised techniques, other methods of automatic labeling exist. For instance, crowd sourcing is a commonly employed labeling tool in machine learning. In these scenarios a human is presented with data and he/she is paid a small amount for each label successfully applied. This has been a wildly successful strategy for labeling data which are

easy for a human—speech to text or image labeling, for instance. However, there are many cases where a human is not the ideal interpreter, such as HydroSense. Humans were never meant to be able to easily ascertain what a pressure wave should or should not look like. Many pressure transients are extremely similar to one another. Moreover, having personally reviewed and labeled a large portion of the existing dataset, I can say without a doubt that labeling is a time consuming and difficult art. Without an array of separate sensor streams from the ground truth network, labeling would have been completely impossible. There were many times in labeling that three or four annotators looked at the same sensor streams and could not agree on a label—and if the ground truth sensor was malfunctioning, the visual cues in the pressure lines were not enough for a human to disaggregate the label. Because of my experiences in labeling, I conclude that crowd sourcing would not be a viable solution.

Selective journaling is another labeling strategy. If I prompt users for labels over a long duration, I can eventually build up the equivalent of many days of training data—however I need a system which automatically prompts the homeowner in a strategic way. There are a number of heuristics I can apply. If I see a long water usage event, I can ask the resident if they just took a shower and where in the home the shower was taken. If I see many repeated duration cycles, I can ask the resident if they ran the dishwasher or laundry machine, *etc*.

At first glance this seems viable. However, this does not necessarily help with valve level disaggregation. Other questions would arise such as: about how hot was your shower? Did you use the hot/cold or cold/cold washing machine cycle? The burden of labeling starts to increase as I want more detail about the event. As such, a journaling system needs to be *highly selective* about the events it asks the homeowner for, and it needs to ask the homeowner for details about the event right after it occurs, to ensure that I get as much correct detail about the event as possible (*i.e.*, if it was hot, cold, or mixed water flow?). This perspective is complementary to the idea of using active learning which is explored in Chapter VIII.

This is a cognitively complex task—do you know how much hot water you usually turn the bathroom sink to? Or how many times you adjust the temperature in the shower? When was the last time you used the kitchen sink? These problems are not insurmountable with the proper interfaces and proper tradeoff between “annoyance” and “labeling detail.”

7.3 Summary

In summary, this chapter explained the features used in HydroSense and how I evaluated the utility of classification feasibility in a longitudinal 5-week study. A properly trained CRF with features from a bagged decision tree yielded the highest performing and most consistent results. I have further broken

down the results to highlight the main areas for algorithmic improvements and the limitations of using labeling data for calibration. I conclude that a selective journaling approach might be feasible in situations where the resident can be motivated to journal water activity, but places a large burden on the resident. A more elegant approach would be to incorporate semi-supervised training to ease or eliminate this burden.

CHAPTER VIII

8. Chapter Eight:
Semi-Supervised Training for Disaggregated Water Sensing

This chapter summarizes the semi-supervised algorithms evaluated in this thesis for disaggregating hot and cold water use with a minimal set of labels. I begin the discussion by injecting prior knowledge in CRFs via other households. Second, I discuss the use of multi-view classification. Third, I propose and evaluate several rule based classifiers for classifying different water usage events automatically. I discuss a combined learning system that incorporates multi-view classification and the rule based classifier in a probabilistic manner, using virtual evidence to inject the information. Finally, I investigate using this model to selectively find water pressure events in real time that the system is unsure about. I simulate the process of asking the homeowner for two to four labels, every other day, and investigate the performance of the classifier, showing it to be 81% correct at the valve level, 89% at the fixture level, and 93% at the fixture category level.

8.1 CRF knowledge injection from other households

In the previous chapter I found that CRFs were slightly less accurate than bagged decision trees when labeled data was scarce. However, I also found that CRFs had the best and most consistent performance when training data was freely available—in the limit, they produced the most accurate results. As such, my first priority is to inject prior knowledge into the CRFs to boost their performance.

The most straight forward method of injecting knowledge is through the transition feature functions of the CRF. When labeled examples are scarce, one cannot expect the transition probabilities between states to be reliable—most of the labels I receive are not from juxtaposed events, so their transition probability is not leveraged properly. Instead, I can rely on the trained transition feature functions from other homes in the analysis. I train two CRFs: one global CRF on four of the five homes, and one minimally trained CRF on the test home, with the minimal label set. The feature transition weights from the global CRF are then used to replace the weights of the transition features for the minimal CRF. This is analogous to

training a global language model based on other homes. The equation for the CRF sequence probability then becomes:

$$p(\boldsymbol{y}^{(i)}|\boldsymbol{x}^{(i)}) = \frac{1}{\mathrm{Z}(\boldsymbol{x}^{(i)})} \exp\left(\sum_{t=1}^{T_i} \boldsymbol{\theta_E}^{\mathrm{T}} \cdot \boldsymbol{f_E}\left(y_t^{(i)}, \boldsymbol{x}_t^{(i)}\right) + \sum_{t=1}^{T_i} \boldsymbol{\theta_T}^{\mathrm{T}} \cdot \boldsymbol{f_T}\left(y_t^{(i)}, y_{t-1}^{(i)}\right)\right) \quad (8.1)$$

Where the subscript "***E***" denotes that the feature and weight are from the emission probability of the minimally trained CRF and the subscript "***T***" denotes that the weights come from the global transition CRF. In essence, the emission probabilities from the minimally trained CRF are simply the weights from a logistic regression classifier (note that the feature function only takes the current state value and the sequence features as inputs). The weights are combined after each minimal CRF has been fully trained and therefore only affects the decoding operation.

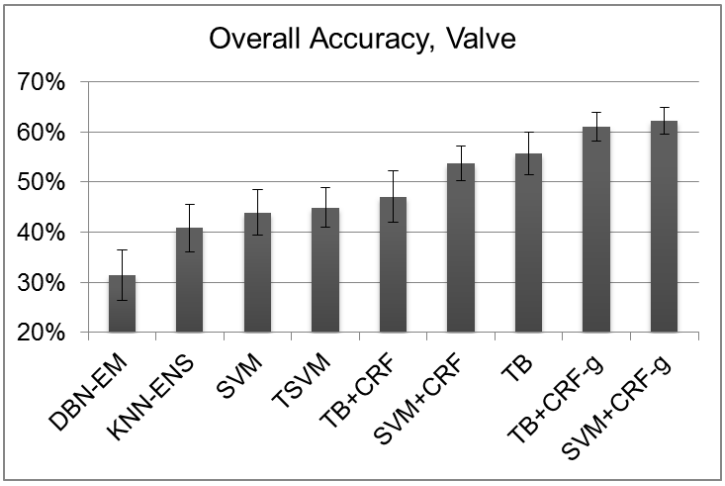

Figure 8-1. Adjusting the weights of the global transition feature weights, compared to the other algorithms.

Explanation: Figure 8-1 shows the performance at the valve level of the representative algorithms from the previous chapter using a minimal set of training examples (8-20 examples per home). Two new algorithms are shown, *TB+CRF-g* and *SVM+CRF-g*. These correspond to CRFs that use the global transition feature weights. Error bars are standard error. **Result**: The injection of prior knowledge through the global transition feature weights improves performance, on average, over the best algorithm by 5-7%. The results are statistically better than TB and each are greater than 60%.

Implication: The CRF was able to successfully leverage the global transition weights. The improvement of 5% accounts for about 750 pressure waves that were previously being misclassified. However, the accuracies are still somewhat low and an analysis of the confusions reveals that, again,

many fixtures are still confused for faucets in the home. Figure 8-2 shows the confusion matrix at the *fixture* level, highlighting this assertion.

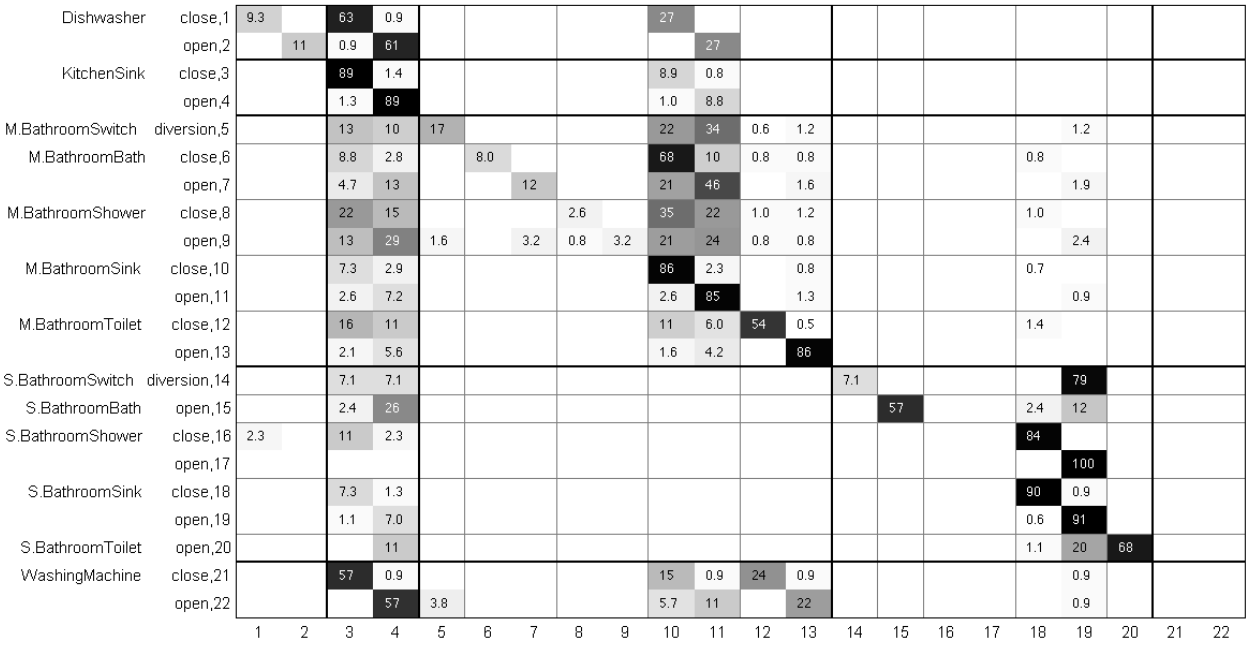

Figure 8-2. Confusion analysis for SVM+CRF-g and TB+CRF-g

Explanation/Result: Confusions are shown for the combined results of *TB+CRF-g* and *SVM+CRF-g*. Almost all classes are confused as either faucets or toilet events, despite the 60% overall accuracy of the classifiers. **Implication**: The performance of the global classifier across classes extremely oriented towards the faucets. If this system were deployed, the homeowner would be assumed to never take showers, wash their clothes, nor run the dishwasher. From this perspective, a "good" classifier is one that not only correctly classifies as many overall events as possible, but also correctly classifies as many events within different classes as possible. To generically capture this, I define a new performance metric which takes the average classification accuracy across classes. That is, I average the percentages across the diagonal of the confusion matrix. I also place a threshold on the number of events in a class category as five or more. This is because, at the valve level, some events only occur 2 or 3 times in the entire database. This occurs for events such as a when the hot and cold water tap at a dual handle faucet are turned on at the same time (corresponding to a *HotAndCold* activation, as opposed to *Hot* and then *Cold*, or, *Cold* and then *Hot*). This is an extremely rare event and is excluded from the averaging because of its rarity.

The new performance metric is referred to as *Across Class Accuracy*, and is graphed at the valve level for the all classifiers in Figure 8-3. In essence, these numbers help to summarize the confusion matrices of all the homes without the need for an in-depth survey of the confusions. As you can see, none of the algorithms perform greater than ~40%, on average, across classes. Interestingly, even though the global *CRF* weights helped to increase the overall accuracy by 5%, they have statistically the same results as *TB* when one averages the classes. None of the algorithms are adequately capturing the rarer classes, such as showers, washing machines, and dishwashers. Also, there are still a high number of confusions between temperatures of the activated fixtures.

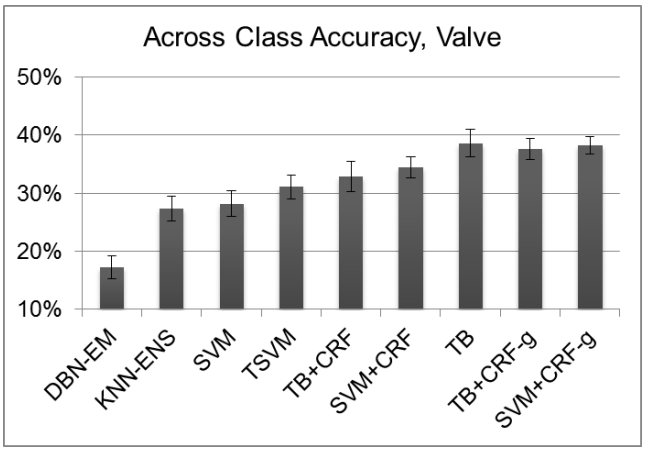

Figure 8-3. Across class average accuracy as a summary of the confusion matrices

8.2 Multi-view classification in IMS

In this section I explore the use of adding unlabeled instances of pressure waves through self-labeling and multi-view classification. I show empirically, however, that such an approach fails to provide distinct information about rarer labels in the dataset, such as the shower and washers.

Multi-view classification (Chapter III) seeks to train different classifiers on the same labeled examples, but using different feature sets that capture different information about the problem. Co-training is a form of multi-view classification where "new" labels are added to a "self-labeled" set based on the agreement between two or more different classifiers [15]. The main idea is that if each classifier is truly independent of the other, then when they agree on a label, that label is probably correct [144]. This is also an example of PAC learning or *p*robably *a*pproximately *c*orrect learning. In this framework, I have a hypothesis from two different classifiers. With high probability (the "P" in PAC), when the two

classifiers agree, the resulting label is true (the "approximately correct" part of PAC), and the results are likely to generalize.

PAC learning has some limitations, however. The "hypotheses" should be generated by "weak" learners. That is, a complex learner like a bagged decision tree or *SVM* is more affected by "label noise," *i.e.,* when an incorrect label is inserted into the pool of self-labeled instances. If a wrong label is inserted, these learners tend to exploit this relationship in the training process. The learner starts associating and confusing similar classes with increased regularity, and eventually learns that these two classes are actually the same class (incorrectly). One is less likely to have this occur with a weak learner, which has trouble making more complex associations between classes. However, label noise can also affect weak learners if the associations between classes are "similar enough" for a weak learner (such as *KNN*) to associate. When this happens, however, it is usually because the two confused classes really are quite similar in the feature space, rather than the learner finding a complex relationship between the classes.

I explore the use of co-training using single classification trees as weak learners. The algorithm is as follows:

Algorithm 8-1. Co-training

1. The features are broken into two sets, F_1 and F_2
2. Randomly, a set of unlabeled instances, X_U are selected from the pool of unlabeled instances, U
3. Two classifications trees, T_1 and T_2 are trained using the initial labeled set, L, each with the features F_1 and F_2, respectively
4. Each classifier predicts the label for each instance X_U
5. For each class, the most confident examples from each classifier are compared
6. Where each classifier agrees on the output label, R classes are added to the labeled set, L
    a. The number of instances, R, is selected based upon the expected number of labels for that class based on their occurrence in L. That is, if class A occurs 25% of the time in L, then R=0.25×(*number of instances in* X_U)
7. Any instances left unlabeled are returned to the pool, U
8. Steps 2-7 are repeated until U is empty or until no instances from X_U are added to L in an iteration
9. The final classifiers, T_1 and T_2 are returned.
10. The confidences from each classifier are combined to form the final prediction accuracy by multiplying them together and taking the class with the maximum confidence.

This is a common method of co-training [15,48]. For breaking the features into disjoint sets, I explored three different scenarios. First the instances are chosen solely on whether they come from the hot sensor or whether they come from the cold sensor, a natural split. Second the features are split by generalizing versus non-generalizing features. Lastly, the features are randomly split at each iteration, before step 2.

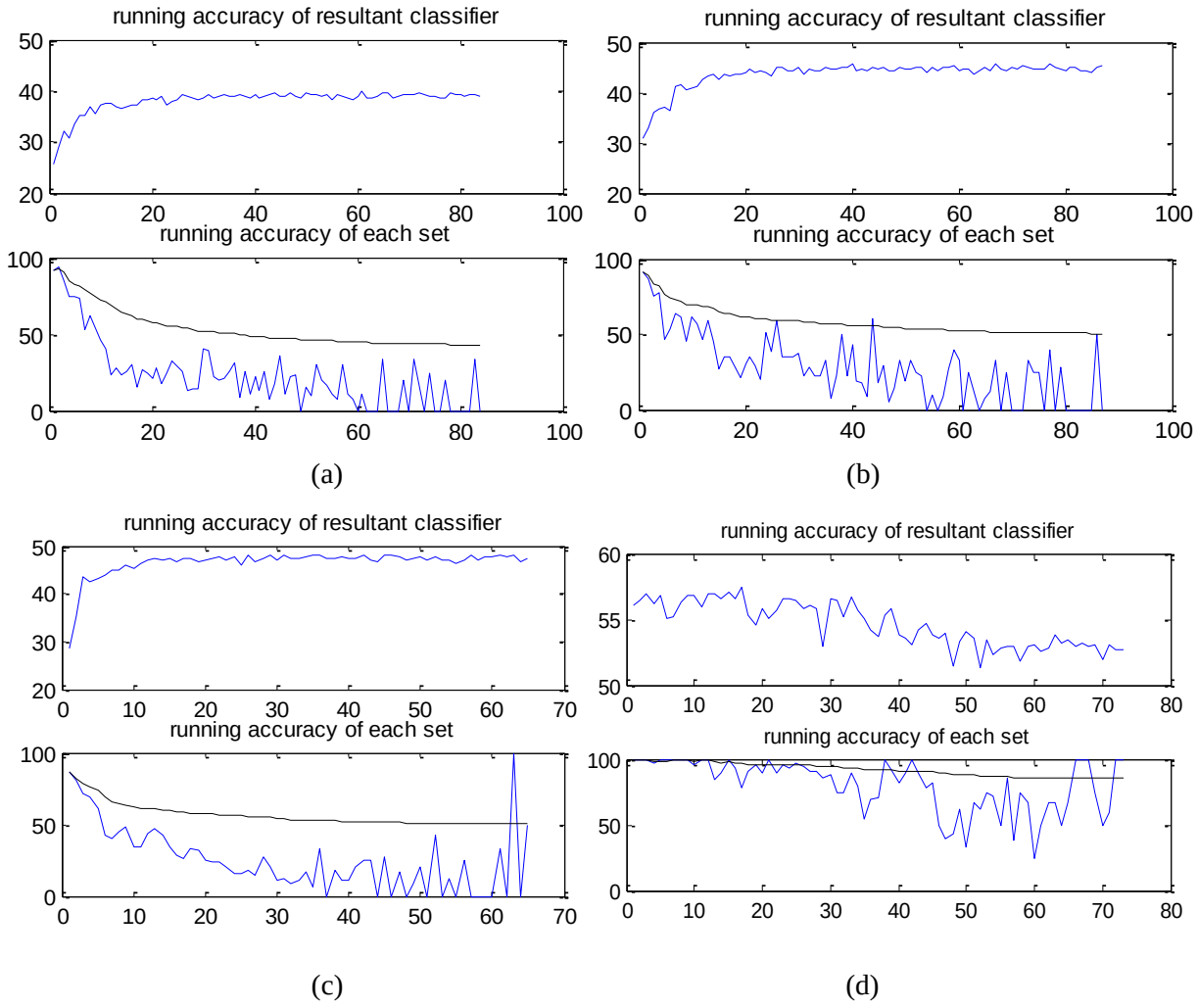

Figure 8-4. Four rounds of co-training for H3. (a): features split by temperature. (b): Features split by whether they generalize or do not. (c): features split randomly at each iteration. (d): a bagged decision tree

Explanation: Figure 8-4 shows the results of co-training with multiple views using three disjoint sets of features: temperature, generalization, and random. The labeled examples that seed the co-training are from the minimal training set (discussed earlier). All features are collected from the expertly chosen feature set. The last plot in the bottom right shows what happens when trying to use co-training on the same data with a bagged decision tree, instead of weak learners. The top of each plot shows the accuracy of the resultant classifier versus the iteration of training. The lower plot shows (1) the *total resultant accuracy* of the examples in the labeled pool, *L*, in black and (2) the *running accuracy* per set that is added to the pool at each iteration in blue. **Result**: Each feature split results in successful co-training where the final accuracy is greater than the original accuracy, except when using the bagged decision tree. Note that the number of labeled examples added by the bagged decision tree is about ~900 at the final iteration. The number of labeled examples added by the weak learners is actually in 2700-3000 range. The

starting accuracy of the bagged decision tree is about 55%, which is greater than the end accuracy of all of the other classifiers. **Implication**: For the data, co-training weak learners is not highly compelling. The resultant accuracy of the weak learners is substantially less than that of the starting accuracy of the bagged decision tree. Moreover, the label noise added by each weak learner quickly degrades the accuracy of examples add to *L* (the bottom of each plot). This is not true, however, for the bagged decision tree. Amazingly, the accuracy of the examples in *L* stays near 100% for many iterations and never drops below 80%. This phenomenon is further explored in Figure 8-5.

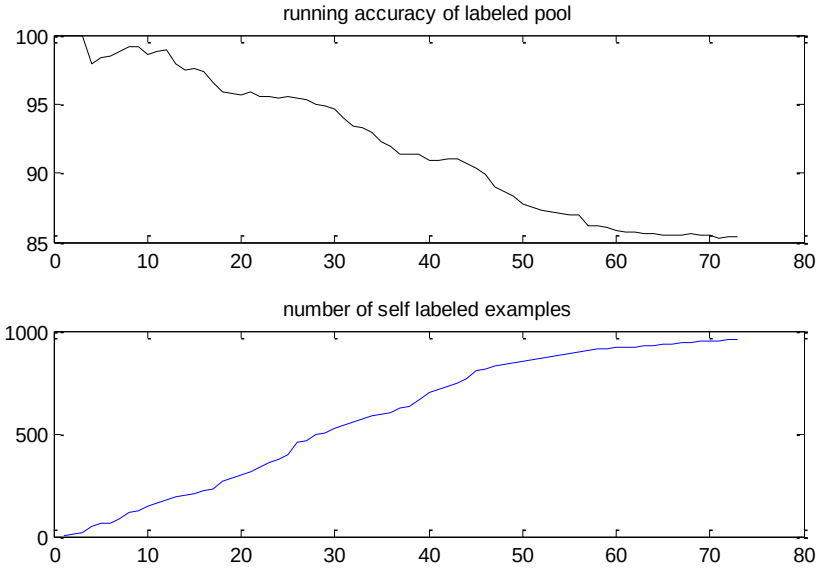

Figure 8-5. The accuracy of the labeled pool and the number of labeled examples versus the iteration of co-training

Explanation/Result: The same information from Figure 8-4d is shown in Figure 8-5 except the number of self-labeled examples is also shown in the bottom plot. The accuracy of the entire self-labeled pool is shown on top. **Implication**: The multi-view classification results in a pool of nearly 1000 examples that is ~85% accurate. If I was to stop the co-training early I could achieve a pool of 400-500 labeled examples that was 95% accurate. Clearly this is advantageous, but the label noise is clearly causing the overall classifier to form incorrect relationships. Moreover, classes added to the labeled pool are not divers. Only about 5 classes are in the labeled set. Even so, co-training with a complex classifier is able to sufficiently capture 5 classes with high accuracy—usually the different faucets in the home. I now investigate whether the feature split of classifiers affects the performance of the bagged decision tree.

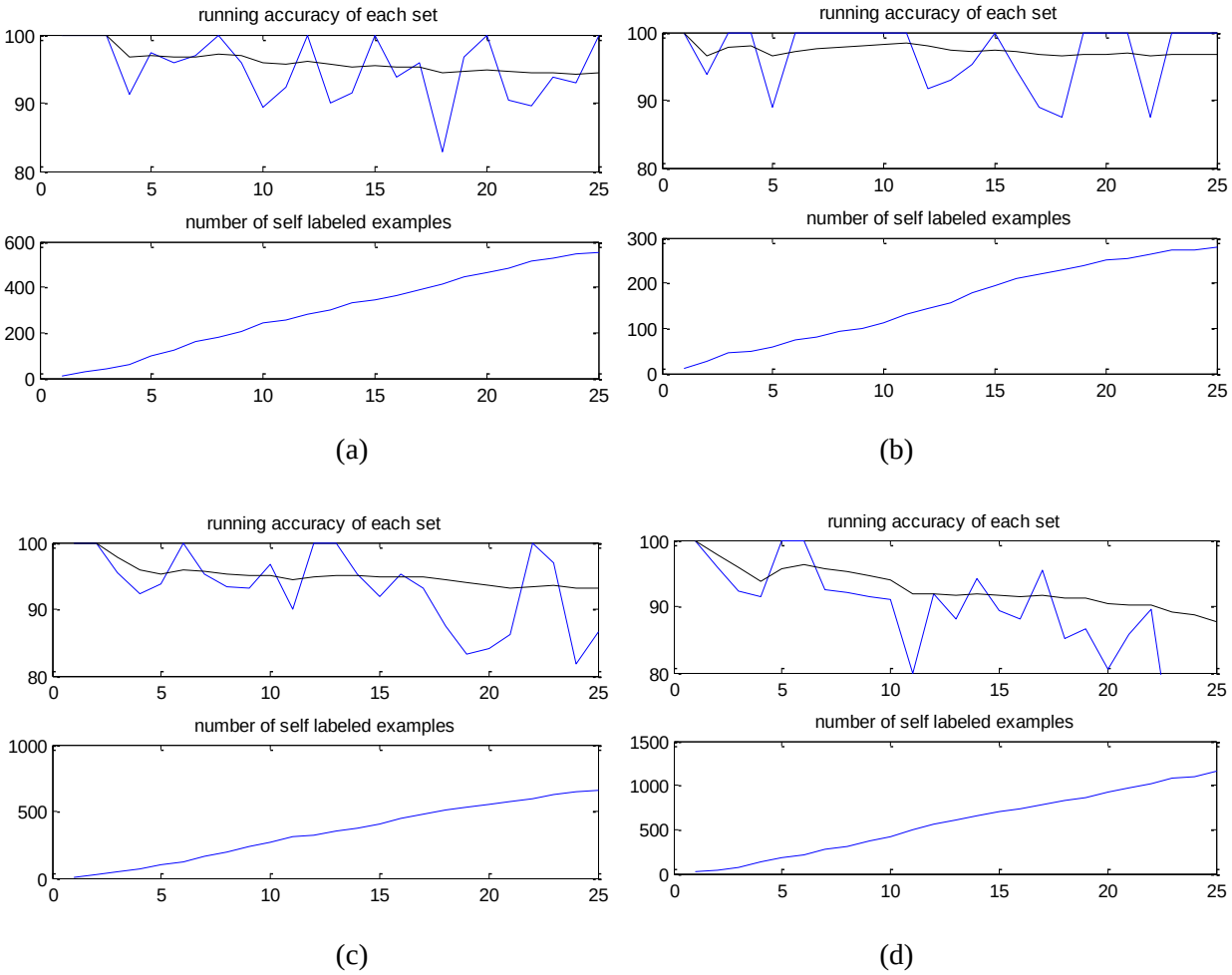

Figure 8-6. The results of co-training four different bagged decision trees

Explanation: Figure 8-6 shows the accuracy of self-labeled pool and number of examples in the self-labeled pool for different feature splits, corresponding to classifiers with different views of the data. The top left, top right, and bottom left are the hot/cold, generalized, and random feature splits, respectively. The bottom right graph shows when no feature splits occur, but when labels are added to *L* based on the confidence of the classifier (self-training instead of co-training). **Result**: The natural feature splits, generalizing and hot/cold, result in a labeled set > 95% accurate. The random feature split and self-training splits result in lesser accuracies, but somewhat larger pools of labeled instances. **Implication**: There appears to be a distinct advantage to using natural feature splits in the data. By using natural groupings like hot/cold sensor data, I can attain a pool of instances which I am highly confident to be correct. Co-training has a clear advantage over self-labeling in the dataset. Up until this point, however, I have only explored the idea of splitting features in the expertly chosen feature set. I now explore the idea of combining this approach with the sparse codebook features.

The sparse features and expertly chosen features are another natural splitting of the feature space. They theoretically provide distinct information about the pressure waves, so co-training may provide a means of attaining even more accurate results [144]. I first explored the idea of randomly splitting the sparse vocabulary and splitting the sparse features into hot and cold as I did the bagged decision trees. However, the results suffered from the same phenomenon I saw before—weak learners had successful co-training rounds, but were not as accurate as running a *SVM* upon the initial labeled instances. Again, the *SVM* was able to capture more complex relationships in the waves and was extremely susceptible to label noise. Co-training resulted in a decrease in the overall accuracy. This was also true for the *SVM+CRF-g* classifier. For the sake of repetition, these results are excluded but follow the same pattern as the weak learners versus bagged decision trees.

I then explored the idea of combining co-training of the bagged decision trees with expertly chosen features and the co-training of the SVM+CRF with sparse features. In this scenario, I get two self-labeled pools, L_{TB} and L_{CRF}—one from each classifier. L_{TB} corresponds to examples from co-training bagged decision trees with expertly chosen features. L_{CRF} corresponds to labels from the *SVM+CRF-g* that are co-trained using the sparse feature set.

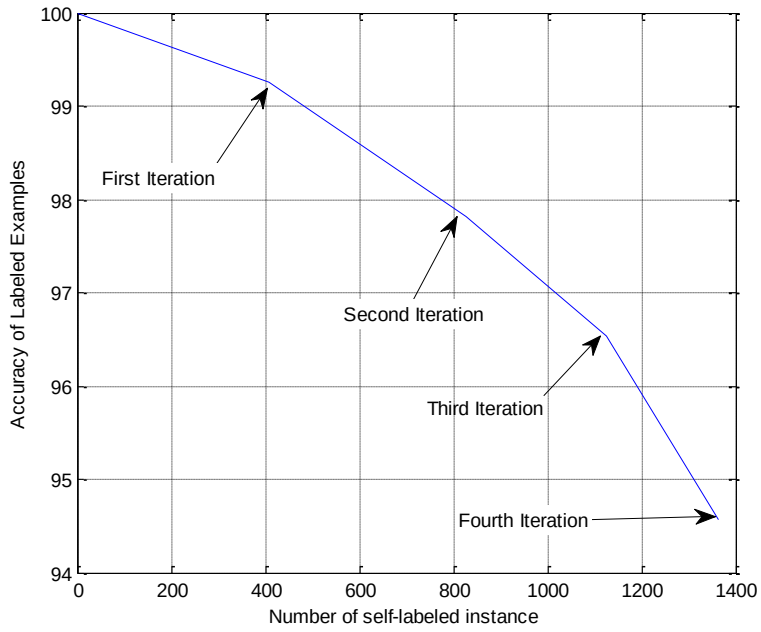

Figure 8-7. Accuracy of combined co-training with expertly selected and sparse features

Explanation: Figure 8-7 shows four iterations of combined co-training between the bagged decision tree and the *SVM+CRF*. At each iteration, the set of self-labeled examples is combined by looking for

instances from each pool where the predictions agree with each other. The full size of the unlabeled pool is ~4000 instances. **Result**: The accuracy after a single iteration is above 99% accuracy and contains 400 examples. This means I have essentially added 396 instances with correct labels and only 4 instances are incorrect. At the second iteration this becomes 792 correct examples and 8 incorrect examples. The trend rapidly drops at the third and fourth iterations, but remains above 94% accuracy even when the unlabeled pool is as large as 1300 instances. **Implication**: Combining co-trained pools from different views resulted in a significant boost to the accuracy of the self-labeled instances. It is amazing that after two iterations I have gone from a minimal set of labeled examples to a set of examples that is over 98% correct, with only eight incorrect labels. However, there is no free lunch—this process is simply grabbing the classes that are "easy" to classify based on our training examples. If I combine the confidences of the different co-trained classifiers (*i.e.*, multiply the confidences to get a new estimate for each class) then the overall accuracy at each iteration is 58.1%, 56.2%, 53.14%, and 52.8%. This is no better than using just a single classifier such as *TB* or *SVM+CRF-g*, in terms of overall accuracy. This is because I have not added any labels to the pool of instances that gives me new information—I have added examples that are similar to the labeled examples. This is also apparent in the diversity of classes added to the self-labeled pool. Even at 800 instances, there are only 7 classes out of 34 for this particular residence (H3). All of the classes are from kitchen and bathroom faucets, activated at different temperatures.

Even so, all is not for naught. Even though I have not increased the diversity of the class assignments and I have not increased the overall classification accuracy, I now have a labeled pool from which I can be very confident that the labels are correct. I also know the pressure waves in the data which have poor generalization given the minimal set. This is important information that I can leverage in other models. In the next sections, I use these labeled examples to come up with a set of heuristics for clustering and gleaning information about surrounding water activity. Later on, I combine this with a probabilistic model using virtual evidence. It is also the basis by which I am able to selectively ask the homeowner for examples that I know are difficult to classify.

8.3 Leveraging prior knowledge

In this section I explore the idea of using custom, rule based classifiers and the pool of self-labeled examples to increase classification performance. I also evaluate the performance of the rule based classifiers in finding classes that are used sparsely such as the dishwasher and washing machine.

8.3.1 Dishwashers and washing machines

Perhaps the most obvious fixtures to classify using rules are the dishwasher and washing machines. The predictive cycling of each have been shown to be good indicators of when they are running [57]. For

classifying these washers, I use one known example from the homeowner of an entire dishwasher run and laundry machine run in order to compare it to a sequence of pressure waves. I save the fill durations, $\boldsymbol{t}_{fill}$, and the time between fills, which I call the cycle duration, $\boldsymbol{t}_{cycle}$. Both are arrays of the durations for fills and cycles throughout the dishwasher/laundry machine run. I also save the magnitude of the pressure drop for each fill, $\boldsymbol{p}_{fill}$. Although I use the labeled examples to get this information, one can easily imagine a system that asks the homeowner to run the dishwasher or washing machine once before going to bed so that it can learn $\boldsymbol{t}_{fill}$, $\boldsymbol{t}_{cycle}$ and $\boldsymbol{p}_{fill}$.

At the end of each day, I look at the entire sequence of segmented pressure waves for that day. For each segmented pressure wave I come up with a hypothesis that it is the start of the dishwasher cycle using a greedy algorithm. The intuition behind the algorithm is to compare the pressure drops and durations of each pair of pressure waves in an unknown sequence, $P(n)$, and compare the time differences and pressure drops in $P(n)$ to $\boldsymbol{t}_{fill}$, $\boldsymbol{t}_{cycle}$ and $\boldsymbol{p}_{fill}$. The algorithm is outlined below:

Algorithm 8-2. Dishwasher and Laundry machine cycle classification

1. For each pressure wave, $P(n)$ in a day long sequence of N pressure waves, assume it is the start of the dishwasher. Initialize three values to zero, T_{fill}, T_{cycle}, P_{fill}. These values will accumulate errors in the duration and pressure drops compared to $\boldsymbol{t}_{fill}$, $\boldsymbol{t}_{cycle}$ and $\boldsymbol{p}_{fill}$. The current pressure wave is denoted as $P(n_0)$
    a. Set $n=n_0$ and $f=0$. Set P_{fill} to the difference between the pressure drop of $P(n_0)$ and $\boldsymbol{p}_{fill}(f)$. Create an empty array $path(n_0)$
    b. For each pressure wave starting at $t= n+1 : N-(n+1)$, calculate the duration between $P(n)$ and $P(n+t)$ and compare it to $\boldsymbol{t}_{fill}(f)$. Save the difference into an array $diff(t)$. If the pressure drop between the $P(n+t)$ and $\boldsymbol{p}_{fill}(f)$ is greater than 50%, set $diff(t)=infinity$
    c. Take the argmin $diff(t)$ to find the closest example to a washer cycle, t_{best}.
    d. Accumulate errors, $T_{fill} \mathrel{+}= diff(t_{best})$, $P_{fill} \mathrel{+}= |P(n+t_{best}) - \boldsymbol{p}_{fill}(f)|$
    e. Increment $n= n+t_{best}$ and add n to the $path(n_0)$ array
    f. Repeat steps b, c, d, and e except use $\boldsymbol{t}_{cycle}$ in place of $\boldsymbol{t}_{fill}$. Accumulate the errors in T_{cycle}. Increment $f=f+1$
    g. Repeat step f until there is no longer any durations or fills in $\boldsymbol{t}_{fill}$, $\boldsymbol{t}_{cycle}$ or $\boldsymbol{p}_{fill}$.
2. Save the accumulated errors, T_{fill}, T_{cycle}, P_{fill} and save the $path(n_0)$. Increment $n_0= n_0+1$
3. Repeat steps 1 and 2 until no pressure waves remain in the sequence $P(n)$.

For dishwashers and washing machines, the idea is that the accumulated errors will be small when I detect a cycle. However, I explicitly want to avoid setting a threshold because it will almost certainly be dependent on the house. Instead, I look for outliers in the accumulated time errors and the accumulated pressure drop difference for each $P(n)$. Throughout the day, any accumulated error that is less than 1.5 times the interquartile range of all the accumulated errors is flagged as the start of the dishwasher or

washing machine cycle. All events saved in the *path*(*n*) are flagged as possible dishwasher or washing machine cycle events. Figure 8-8 shows an example of when the dishwasher is successfully detected and when there are no dishwasher events for a given day. The accumulated error is shown for T_{fill} and the units are in seconds.

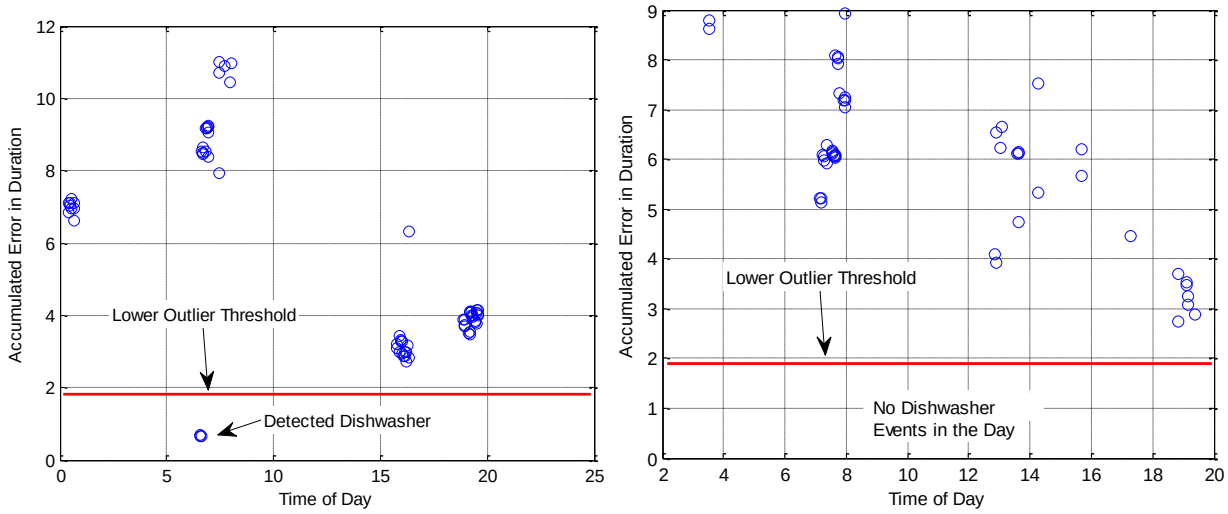

Figure 8-8. Examples of detected dishwasher start cycles and when no dishwasher events occur in a day.

Using this approach I can also detect clothes washers, however I relax the bounds on the pressure drop to be the minimum of the pressure drop on the hot or cold water line. This is because the clothes washer can be run using multiple temperatures so pressure drops might be different on the hot or cold pressure lines. The fill cycles can also be different depending on the amount of water selected at the washer (*i.e.*, small versus large). Therefore I ignore the duration of fill cycles for the clothes washer and only look at accumulated error in cycle durations, T_{cycle}.

Using this approach I am able to detect 73% of all dishwasher cycles in the database, with only 15 false positives (*i.e.*, 15 pressure waves incorrectly flagged as belonging to the dishwasher). For washing machines, I found 70% of all cycles with about 16 false positives pressure waves (also see Table 8-1). The majority of the events not detected come from H3, where the cycles of the dishwasher or washing machine are interrupted by the homeowner (*i.e.*, opening the lid or door to the appliance to add more dishes or clothes). Simply opening the appliance quickly does not affect the algorithm, but leaving the system idle for longer periods (5-10 minutes) confounds the algorithm. Even so the precision of the classifier is quite good so that I can be confident about when the clothes washer or dishwasher is being run.

Because I also know when the dishwasher is being run, I can look at events that occur up to five minutes before the cycle begins. If I flag these events as kitchen sink events (*i.e.*, rinsing dishes before running the dishwasher) I will have a precision of 70% for 115 pressure waves. Although I note that this is only a modest fraction of the number of events that occur at the kitchen sink.

8.3.2 Showers

At first glance, detecting showers seems an easy task. I simply need to look for the longest events in the home. However, in the dataset, median shower time is only 4 minutes. The longest showers can last beyond 10 minutes, but these events are rare. Only about 5% of all showers last longer than 6 minutes. Even more worrying is that the flow rate of today's showerheads are about the same flow rate as the kitchen sink and many bathroom sinks—so the pressure drops are also similar. And the duration of sink usage is erratic. Less than 1% of sink usage is above 5 minutes—but since there are so many faucet events it's easy to assign a long face wash at the bathroom sink or a long draw of water at the kitchen sink as a showering event. To be sure, duration is useful. However, I can only assign about 15% of showers using duration alone without a substantial false positive rate. Clearly, a more clever approach is needed.

Instead I leverage the fact that many showers have a diverter valve. If the diverter is initially closed, then turning on the water activates the bath faucet, and after the diverter is activated, changes the flow to the shower. This results in a large pressure drop followed by a jump in pressure. However, there are many ways in which this diversion can occur. The diverter can initially be set to the shower, in which case the bath faucet is never used. Or, the homeowner can turn the water to the bath before shutting off the shower, in which case there is a large pressure drop at the end of the bath. Or the diverter valve can be used at the beginning and end of the shower.

Even so, humans are habitual. In the dataset, certain individuals use the water in predictable ways. For instance, in H1, the process is always *bath->diversion->shower*. In A1, one occupant always uses the *bath on->diversion->diversion->bath off* process while the other occupant always goes from *shower on* to *shower off*. In H3, which has the most number of occupants there is an incredible diversity in ways the showers are used. H2 does not have diverters, but instead has walk in showers.

Whenever the diverter is used, it is typically easy to tell that the bath was turned on or off because the pressure drop is uncharacteristically large. However, if diversion is not used, the start or end shower event is difficult to classify. This is because a number of other water fixtures are typically being used around the same time as the shower—such as the faucet, toilet, and many times the kitchen sink by the other occupant. In the midst of all these pressure waves, the beginning or end of the shower is hard to judge. I

may know that a shower occurred, but I may not be able to say exactly which pressure wave was the open or close event. I formalize the finding of showers through the following algorithm.

First I ask the homeowner for an example of them turning the shower on so that I can ascertain whether they used the diverter or not and whether it occurs at the beginning or the end of the shower. I save the information about when the diversion(s) occurred and the pressure drops from the hot and cold pressure line for each *diversion*, *shower*, and *bath* event. These are saved to an array $\boldsymbol{p}_{bath}$. Once I have the examples, I classify shower events in a day long sequence of pressure waves, denoted by $P(n)$ where $n=1{:}N$. For each pressure wave,

Algorithm 8-3. Shower classification

1. Initialize the accumulated error as $P_{fill}=0$ for the current pressure wave, $P(n)$. Initialize an array $path(n)$ to contain the current value of n.
2. Assume, $P(n)$ is the initial opening of the shower or bath. Record the difference in expected pressure drop versus the actual pressure drop, $P_{fill} \mathrel{+}= |P(n)- \boldsymbol{p}_{bath}(1)|$
3. If the next expected event is a diversion from bath to shower,
    a. find the event within the next 30 seconds with the most similar pressure drop to $\boldsymbol{p}_{bath}(2)$.
    b. Denote the pressure wave that is most similar as $P(n+k)$. Update the accumulation error $P_{fill} \mathrel{+}= |P(n+k)- \boldsymbol{p}_{bath}(2)|$. Update $path(n)$ to include the value $n+k$. Proceed to step four.
4. If the next expected event is shower to bath diversion (else go to step five),
    a. look for the end event of the bath that has the most similar pressure drop to $\boldsymbol{p}_{bath}(end)$ and is greater than 3 minutes from $P(n)$, but not more than 15 minutes.
    b. Denote the pressure wave that is most similar as $P(n+k)$. Update the accumulation error $P_{fill} \mathrel{+}= |P(n+k)- \boldsymbol{p}_{bath}(end)|$. Update $path(n)$ to include the value $n+k$. Proceed to step six.
5. If the next expected event is a shower close,
    a. look for the end event of the shower that has the most similar pressure drop to $\boldsymbol{p}_{bath}(end)$ and is greater than 3 minutes from $P(n)$, but not more than 15 minutes.
    b. Denote the pressure wave that is most similar as $P(n+k)$. Update the accumulation error $P_{fill} \mathrel{+}= |P(n+k)- \boldsymbol{p}_{bath}(end)|$. Update $path(n)$ to include the value $n+k$. Return the accumulated error, P_{fill}.
6. If the previous expected event is a shower to bath diversion,
    a. look for the event within 30 seconds of the close event that has the most similar pressure drop to $\boldsymbol{p}_{bath}(end\text{-}1)$.
    b. Denote the pressure wave that is most similar as $P(n+k)$. Update the accumulation error $P_{fill} \mathrel{+}= |P(n+k)- \boldsymbol{p}_{bath}(end\text{-}1)|$. Update $path(n)$ to include the value $n+k$. Return the accumulated error, P_{fill}.

The intuition behind the algorithm is that I accumulate the error in pressure drops based upon the where I think a bath open or close event occurs and where the corresponding diversion occurs. I denote the entire set of accumulated errors by the array $\boldsymbol{P}_{fill}(n)$. For detecting the showers, I again employ outlier

detection. When I detect an outlier in the sequence of distances, I flag all the events in $path(n_{outlier})$ as being a shower, bath or diversion, depending on the specific behavior I have seen before. Figure 8-9 shows an example of when this algorithm correctly finds a showering event, and when it misses an event. Black circles show the accumulated pressure error, P_{fill}, versus the time of day. The dotted line shows the threshold for outliers for that particular day of data. The blue circles denote an actual shower event (bath, shower, or diversion) occurred at those times. When the shower is found, a diversion was performed at the opening of the shower. When the shower is missed, it is because it did not contain a diversion event. In this particular example, the beginning of the shower is also misclassified, but the final *diversion* and *bath off* pressure waves are found. Note the amount of other water activity also occurring around the times that the showers occur.

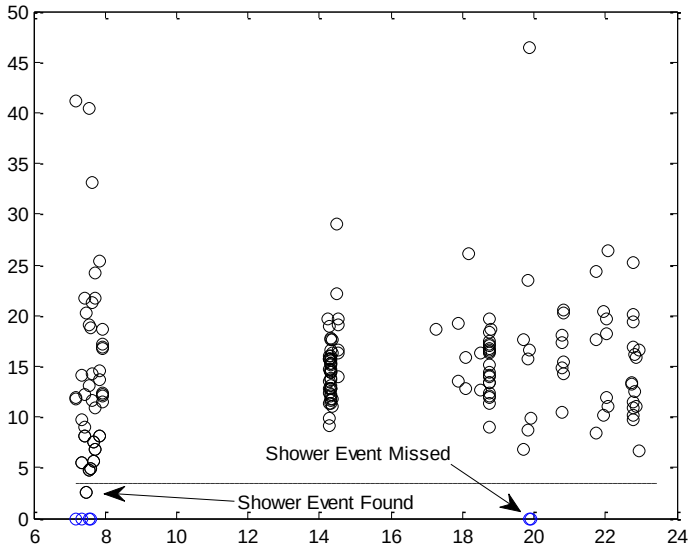

Figure 8-9. The variable $P_{fill}(n)$ for a particular day of data, graphed versus the time of day at which the event occurred.

I also apply a number of different heuristics:

- two shower events cannot overlap each other's *path* arrays.
- the number of shower events cannot exceed the number of occupants.
- when events are detected exceeding the number of occupants, I sort $\boldsymbol{P}_{fill}(n)$ and take the minimum values, up to number of occupants.
- If no shower events are detected and it is a weekday where I have seen clusters of water use, I force the system to select the minimum value of $\boldsymbol{P}_{fill}(n)$ and assign it as a shower.

- Lastly, if a sustained pressure drop is observed that lasts longer than 6 minutes, I automatically flag the beginning and end pressure waves as shower events, regardless of the value of $\boldsymbol{P}_{fill}(n)$.

Applying this to the dataset results in a positive detection of 43% of all shower, bath, and diversion events. The main missed events occur in H2 where the showers do not have diverter valves (walk in showers) and in A1 where the diverter is not used often. Surprisingly, this algorithm also resulted in 58 false positives. The main culprit here is no correctly detecting the shower on or shower off event in noisy clusters of water use. Also, in H1, there are a number of events where the toilet is flushed (a high flow event that looks similar a bath open) and the bathroom sink is opened directly when the toilet closes, making it look like a diversion event. The water is then ran for a long period—perhaps someone washing their face or leaving the water on while brushing their teeth. It is also of note that even though I miss many of the shower open and close events, I am still detecting the bath open or bath close event of the shower 73% of the time. So I may not know the exact pressure wave to assign for the showerhead, but I know that a shower occurred 73% of the time.

8.3.3 Toileting

The toilet, also, may seem at first glance to be an easy classification. It has a specified duration and usually a large pressure drop. However, one must also weigh that sinks close to the pressure sensor also have large pressure drops. Moreover, the sheer number of times faucets are activated means some events also have durations similar to that of a toilet. Also, the toilet duration is not extremely consistent. The filling time of the tank depends upon how long the lever was depressed for many toilets, which controls how much water flows from the tank into the basin. For the toilets in the database, they can fluctuate in fill time by as much as 35 seconds, but typically only fluctuate by about 10 seconds. However, simply applying duration heuristics does not achieve good performance. If I flag all events that occur in the specified duration (10 seconds) of a toilet, I can select 78% of all toilets, but with an inordinate amount of false positives from the kitchen and bathroom sinks, about 1200 events. As such, I also apply a threshold that the sustained pressure drop must be within 10% of expected. This reduced the true positive rate to 55%, but also reduces the number of false alarms to 338 events. Finally, I can further filter the events by requiring that they also have a compound event in the middle of them, corresponding to a bathroom sink use (washing hands) after the toilet is used. This reduced the true positive rate to 39%, but also reduces the number of false alarms to 119 events. As such, I flag events that meet all criteria (duration, pressure drop, and compound events in between) as “probable toilets.” I assign events that meet the criteria of duration and pressure drop as “possible toilets.” These flags are used later on by the dynamic Bayesian network.

8.3.4 Clustering from the self-labeled pool

Finally, I use the self-labeled pool of instances to help assign whether clusters of water use should be flagged as coming from the different bathrooms in the home. For example, if I have classified the master bathroom sink as being used during co-training, then I flag events within 2 minutes of the event as "likely coming from the master bathroom." Similarly I flag events from the secondary bathroom if they occur within 2 minutes of a self-labeled secondary bathroom fixture. If an event occurs within 2 minutes of both labels from a master and secondary bathroom, then I flag it as "likely from the bathroom," but do not assign a specific label to which bathroom it came from. Applying this clustering across the residences in the dataset resulted in a precision of 84% when detecting bathroom versos non-bathroom, and precision of 62% at distinguishing the master bathroom from the secondary bathroom. That is, the percentage of total events flagged that were correctly flagged as bathroom, master bathroom, or secondary bathroom. This is encouraging, because it helps to further distinguish bathroom events classified above, such as toilets and showers. For a large percentage of events, I can say *which* toilet or shower was activated from in the home (*i.e.*, fixture level versus category level classification).

Summary of rule based classification performance:

	True Positive Rate	False Alarms	Precision
Dishwasher	73%	15 pressure waves	90%
Laundry Machine	70%	16 pressure waves	89%
Kitchen Sink	-	19 pressure waves	70%
Showering	43%	58 pressure waves	85%
Toileting	39%	119 pressure waves	82%
Generic Bathroom Clustering			84%
Distinguishing Bathrooms			62%

Table 8-1. A summary of the performance of the different rule based classifiers for detecting different water usage events

8.4 Virtual Evidence

I now turn my attention to incorporating the rule based classifier and co-training in a probabilistic framework. I chose to use virtual evidence (discussed previously) because it naturally lends itself to incorporating different beliefs from a wide array of information sources. All of the virtual evidence I employ is employed only in the decoding procedure. Trying to incorporate it during the training procedure actually decreased performance. This is not surprising because I have very few labeled examples. It is difficult to tune the virtual evidence probabilities with a small number of examples.

The virtual evidence nodes I employ are based primarily on the heuristic classifiers and the co-trained labels, which are discussed first. However, I also use a number of virtual evidence nodes based upon the demographics of the residents in the home and the type of fixtures in the home.

The underlying graphical model I use is a hidden Markov model that is identical to the model used in previous results sections, *DBN-EM*. The model has a conditional probability table that parameterizes the state to state bigram and emission probabilities that are parameterized by Gaussian mixtures. The number of mixtures per state is variable and is determined during training from a splitting and vanishing procedure, as discussed. Similar to the *CRF*, the transition probabilities are not trained from the minimal training examples, but are taken from a global model trained on other residences. That is, a global transition matrix is determined from four residences in the dataset and used to test on the remaining home. In essence, the Gaussian mixtures are the only terms parameterized by expectation maximization. Variance limiting is used to prevent spurious singularities in the training data.

The model is shown in Figure 8-10 using shorthand notation. The dashed lines around the chunk and "T" mean that the chunk is rolled out along the entire sequence of pressure events. Recall from previous experiments that one sequence is equal to a days' worth of pressure events. All observations are always fully observed, denoted by solid gray circles. The hidden nodes, denoted by an empty circle, represent the fixture which generated a given pressure wave. During training by expectation maximization, the labeled instances become observed state nodes. During training I keep unlabeled nodes between labeled instances hidden, rather than removing them. This allows the EM procedure to regularize its training using the unlabeled instances from the given sequence—this marginally increases accuracy (about ~1.5%, on average). I do not regularize with unlabeled data from other sequences, however, because I have found this to drastically reduce the classification accuracy of the system. The sheer number of unlabeled sequences tends to make all the Gaussian mixtures converge to an area in the feature space that is not representative of the actual class/observation joint probability that I am attempting to parameterize.

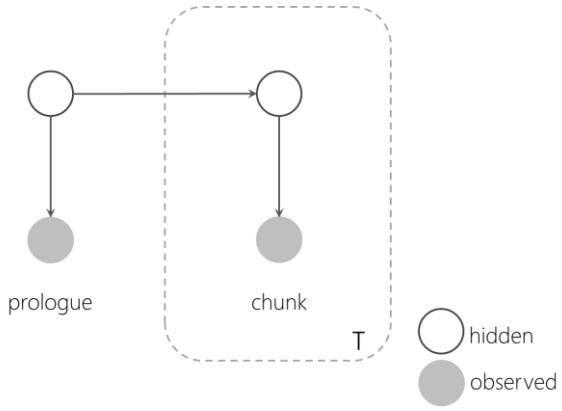

Figure 8-10. The initial DBN graphical model I train on the minimal label set

Recall that the valve level accuracy of the DBN-EM system is about 30%. Using the global transition matrix helps to increase the accuracy to 42%, bringing it onto the same performance level as the *SVM* and *KNN-ENS* algorithms. However, there is still tremendous room for improvement as the *SVM+CRF-g* algorithm has an accuracy of about 60%. I now explain the different forms of virtual evidence employed in decoding to increase the accuracy of the classifier.

8.4.1 VE: Self-labeled examples with high and medium confidences

The first injection of knowledge with virtual evidence occurs by using the self-labeled examples from the co-training algorithm. The self-labeled examples are divided into three categories: high, medium, and low confidence labels. The high confidence instances are those in the self-labeled pool. I can be near certain that these instances are labeled correctly. The medium confidence examples are the examples from the self-labeled pools of co-trained *TB* and co-trained *SVM+CRF-g* instances that did not make it into the final self-labeled pool. These instances were not added to the final pool because only one algorithm, TB or *SVM+CRF-g*, was confident enough to add the instance to its co-training pool, but not both. The low confidence examples are all other instances in the unlabeled pool.

This knowledge is used with virtual evidence by inserting a node into the *DBN* model that is always equal to one, but has a parent whose value is determined based upon the class from the self-labeled pool, and its confidence. This is shown in Figure 8-11. The prologue is no longer shown, but has the same form as the chunk, except there is no input transition. The dark node is the "pivot" for the virtual evidence and has a value that is always "1." The node forms a clique with the hidden fixture state and two other nodes which denote the self-label of the current instance and whether label is high, medium, or low confidence. The conditional probability table (CPT) for this clique is formed using a decision tree that:

1. If "confidence" is low, the value is of the CPT is 1, regardless of the input from either parent

2. If the confidence is medium, the value of the CPT is 1 when the fixture state and self-label state are equal to each other and ½ otherwise.
3. If the confidence is high, the value of the CPT is 1 when the fixture state and self-label state are equal to each other and 1/20 otherwise.

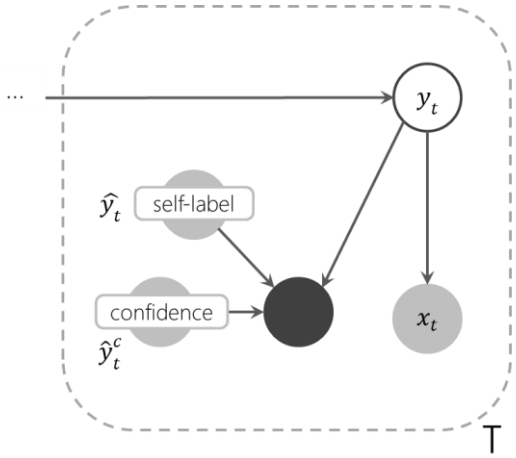

Figure 8-11. The addition of a single virtual evidence node.

This CPT has the effect of lessening the probability of certain fixture states when the confidence from the self-training is high or medium confidence and has no effect when the confidence is low. Specifically, the node says that I believe the self-trained label to be 20 times more likely than any other label when I have high confidence and only twice as likely when I have medium confidence. It is important here that I do not assign a "0" probability when the fixture state does not match a high confidence self-label. This is because this would have the effect of splitting the sequence. All the probability mass from all sequences would converge in a single node during the decoding process.

This virtual evidence node changes its value over time and reflects the belief about the actual label based upon the current self-label. I can denote the probability of the clique as $P(y_t|\hat{y}_t, \hat{y}_t^c)$ where $\hat{y}_t$ represents the self-label and $\hat{y}_t^c$ represents the confidence of the estimate.

8.4.2 VE: Less likelihood for already labeled events

The next way in which I inject knowledge from the self-labeled pool is through the distribution of labeled examples. This virtual evidence node is informed by how many times the self-labeled instance appears in the pool, compared to how often I expect the fixture to be used. The distribution of each fixture is determined from the collected data from other homes. For example, the expected distribution of labeled master bathroom events is ~12%. If this node has been labeled more than 12% in the self-labeled pool,

then I penalize its probability over the entire sequence. The node is shown in Figure 8-12 and is a decision tree with five possible values.

1. If the parent, y_t, is equal to a class that has been labeled more than 120% of its expected distribution, then the CPT returns 0.01.
2. If the parent, y_t, is equal to a class that has been labeled between 80% and 100% of its expected distribution, then the CPT returns 0.1.
3. If the parent, y_t, is equal to a class that has been labeled between 50% and 80% of its expected distribution, then the CPT returns 0.5.
4. If the parent, y_t, is equal to a class that has been labeled between 20% and 50% of its expected distribution, then the CPT returns 0.75.
5. If the parent, y_t, is equal to a class that has been labeled less than 20% of its expected distribution, then the CPT returns 1.

This CPT boosts the probability of under-labeled classes based on the previous knowledge of other homes water use. The probability can be given by $P(y_t|\boldsymbol{Y}_{emp}, \boldsymbol{Y}_{self})$ where $\boldsymbol{Y}_{emp}$ is the expected empirical distribution of state assignments and $\boldsymbol{Y}_{self}$ is the self-trained distribution of labels. The value of this conditional probability does not change over time. It is static over the entire sequence.

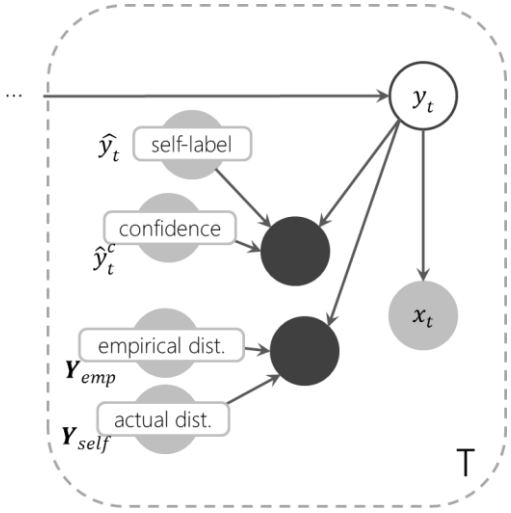

Figure 8-12. The combination of two virtual evidence nodes

8.4.3 VE: Clustering and fixture assignment from the rule based classifier

I now inject knowledge into the model using knowledge from the rule based classifiers. Recall that these classifiers can be a class category, such as shower, diverter, bath, dishwasher, probable toilet, maybe toilet, unknown, *etc*. I denote these assignments as $\tilde{y}_t$. Also recall that the clustering node can take on

values such as bathroom, master bath, secondary bath, and unknown. I denote these assignments as $\check{y}_t$. The CPT for this virtual evidence node has three parents, $y_t, \check{y}_t$, and $\tilde{y}_t$. Formally the conditional probability, $P(y_t|\check{y}_t, \tilde{y}_t)$, is given by:

1. If the fixture state, y_t, matches both the cluster and the category then the CPT returns 1.
2. If the fixture state, y_t, matches the fixture category but not the cluster then the CPT returns 0.5.
3. If the fixture state, y_t, is a not a toilet, but the category is a "probable toilet" then the CPT returns 0.01.
4. If the fixture state, y_t, is a not a toilet, but the category is a "maybe toilet" then the CPT returns 0.1.
5. If the fixture state, y_t, matches the cluster and the category is unassigned the CPT returns 1.
6. If the fixture state, y_t, does not match the cluster and the category is unassigned the CPT returns 0.25.
7. If the cluster and category states are unassigned, then the CPT returns 1.
8. Otherwise, if the fixture state, y_t, does not match the category, the CPT returns 0.01.

8.4.4 VE: Temperature preference

Next I inject knowledge into the model based upon the knowledge of labeled examples from the homeowner. Specifically I seek to enhance the ability of the model to distinguish between hot water, cold water, and mixed water use. I perform this using a mix of multi-view classification and bagged decision trees on the minimal set of labeled instances. I first label the minimal set of instances only by its temperature state, *Hot*, *Cold*, or *Mixed*. I then use co-training with a bagged decision tree as before, where I separate the features based upon the given sensor. However I only run co-training for a single iteration. This results in two classifiers that each has a separate hypothesis about the temperature state of the valve. I combine the two classifiers by multiplying the confidences together and taking the class with the highest confidence. On average, this results in a classifier that is ~73% accurate across the different residences (chance is 33%).

However, I do not want to inject temperature information when the classifiers are not certain about the temperature state. The multi-view classifier affords me that option because I can have a greater trust in the confidence levels. I therefore devise a virtual evidence node that takes the class confidences from the multi-view classifier as input. When the max margin between the most probable class and next most probable class is greater than 30%, I give that temperature state more probability. When the margin is less than 30% I do not penalize any temperature assignment. If the max margin is given by T_{margin} and the

temperature state from the classifier is given by T_{temp} then the conditional probability, $P(y_t|T_{margin}, T_{temp})$, is given by:

1. If the T_{margin} is less than 30%, return 1.
2. Otherwise, if the temperature of the fixture state matches T_{temp}, return 1.
3. Otherwise return 0.1.

8.4.5 VE: Dual versus single handle and left handed versus right handed

This virtual evidence node is meant to leverage information about the type of faucet and, when the faucet is dual handle, leverage information about the handedness of the primary user of that faucet. This node is fairly simple. When the faucet is dual handle, preference is given for *mixed* water use opposed to *just hot* or *just cold* water. This is based on empirical observations. A single handle faucet is almost always activated by pushing the lever straight up to get water. Only when there is a preference for hot or cold water does the valve get turned "up-and-to-the-left" or "up-and-to-the-right."

For dual handle faucets, the preference for activated temperature is given by the handedness of the operator. When no preference is needed for hot or cold water, individuals often use their free hand. For a right handed person, this means they use their left hand to activate the hot water. For a left handed person, they tend to activate the cold water more. In the dataset, three residences used dual handle faucets, A1, H1, and H3. The individuals in A1 and H1 are all right handed and the preference is given for hot water. In H3 a bathroom is shared between a right handed person and a left handed person. As such, the distribution of water used is much more balanced, with about as many hot events as cold events.

The conditional probability of the clique is given, then, by the type of faucet F and the average handedness of the primary operators, H_{ave}. $P(y_t|F, H_{ave})$:

1. If the fixture state is not a faucet, return 1.
2. If the fixture state is a faucet and F is single handle return 1 for mixed water use, and 0.5 otherwise.
3. If the fixture state is a faucet and F is dual handle return 0.01 for mixed water use.
4. If the fixture state is a faucet, F is dual handle, and H_{ave}=*right* return 1 for hot water and 0.75 for cold water use.
5. If the fixture state is a faucet, F is dual handle, and H_{ave}=*left* return 1 for cold water and 0.75 for hot water use.
6. If the fixture state is a faucet, F is dual handle, and H_{ave}=*no preference* return 1 for cold water and 1 for hot water use.

Figure 8-13 shows all of the virtual evidence cliques used in the model thus far and the shorthand representation of the five different virtual evidence nodes. I now turn the discussion to incorporation a valve pairing clique.

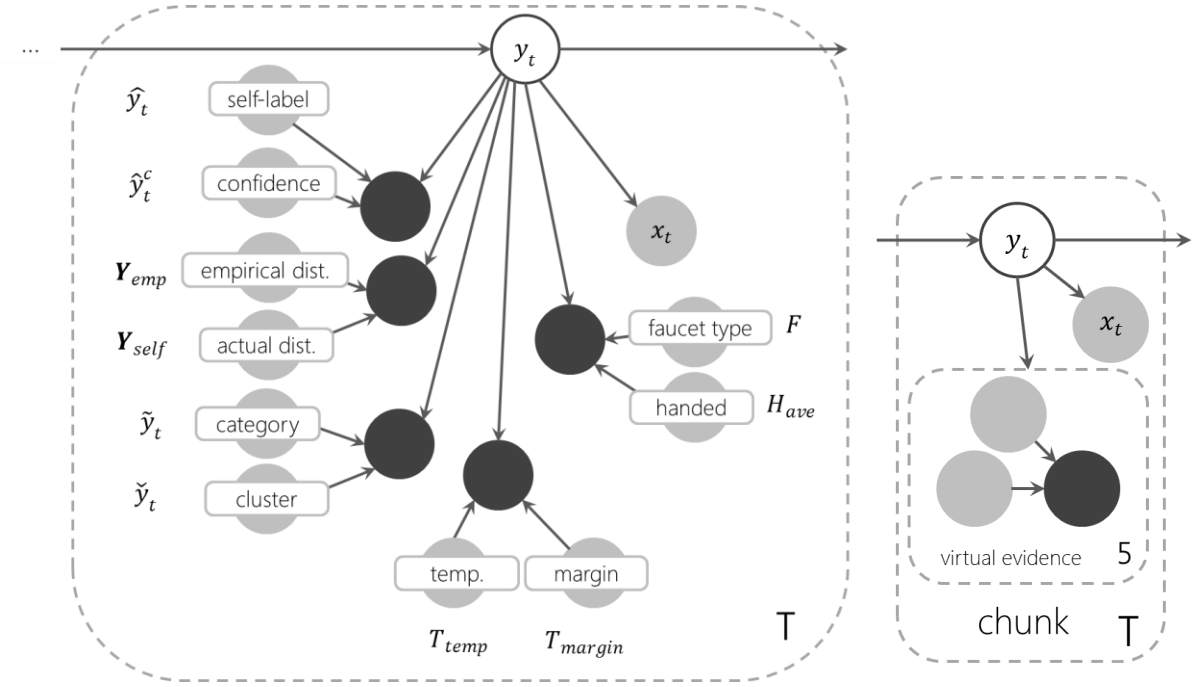

Figure 8-13. A virtual evidence chunk that incorporates all but the valve pairing clique (left) and its shorthand notation (right)

8.4.6 VE: Possible valve pairings (grammar)

The last form of prior knowledge takes into account the way that valves may be permissibly used in a residence. Similar to the grammar imposed by the *TM-LM* model, I formalize the pairing of valves by making permissible valve assignments more likely and impermissible assignments less likely. Whereas in previous work I incorporated this knowledge by reordering N-best lists, I now investigate how a four state pairing clique performs. The parents of the clique are the current fixture state and the past three fixture states. This structure is denoted in Figure 8-14. I again use shorthand to represent the different virtual evidence cliques, of which there are five.

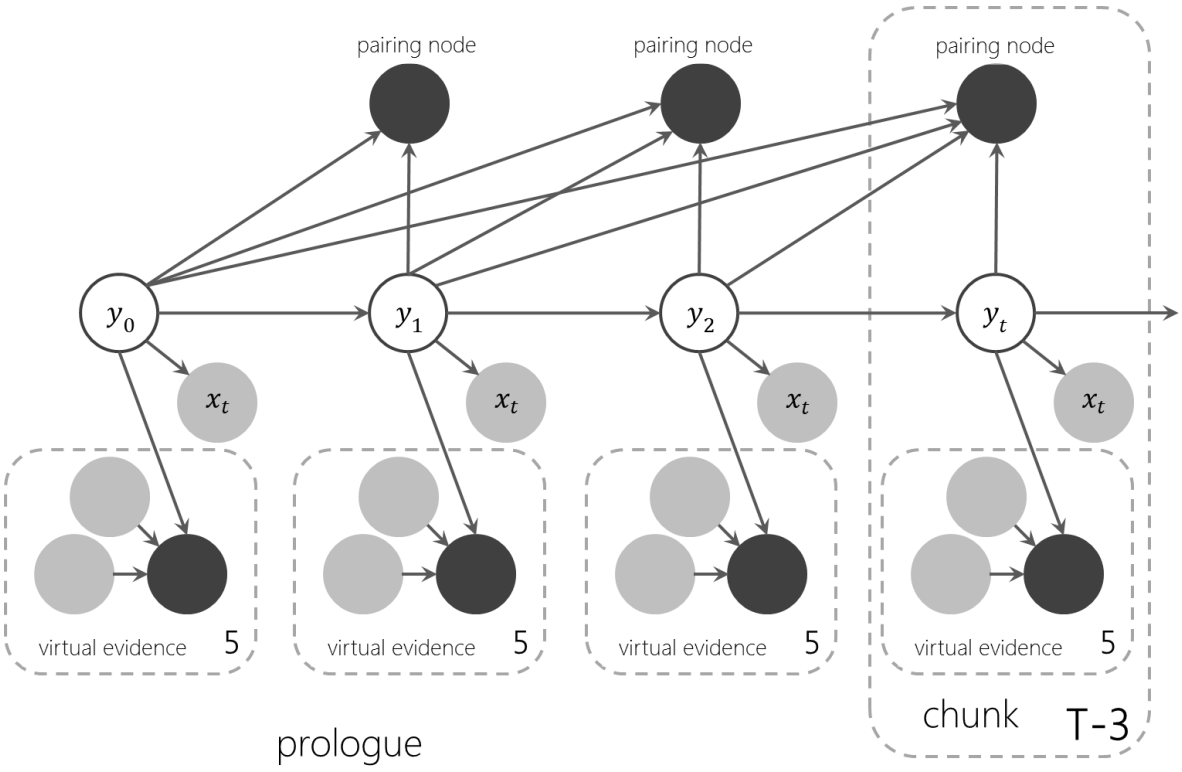

Figure 8-14. The pairing structure of the final graphical model used in the VE analysis.

The quad state pairing node takes in four different nodes as input, but its function is very simple. The conditional probability $P(y_t|y_{t-1}, y_{t-2}, y_{t-3})$ is given by:

1. If the current state is a close event, then the corresponding open event for that fixture must occur within one of the last three states. If the fixture is there, but the temperature of the valve does not match, return 0.5. If there is no pairing, return 0.01.
2. If the current state is a shower close event or bath close event, return 1 if a bath close or shower close does not appear in the previous three states, otherwise return 0.01
3. If the current state is an open event, return 1 if at least one close event appears in the previous three states and none of the previous events are also the same fixture open event. Otherwise return 0.01.

Shower and bath event are special cased because they tend to have longer durations than the other events. Therefore, I want to avoid penalizing them if the previous state assignment does not include a diversion of bath/shower open. Many other compound events could occur between the open and close, such that the shower might not be paired in four states. I note that the addition of the pairing structure adds considerable computation time to decoding. Whereas the previous injections of virtual evidence did

not enlarge the state space trellis that needs to be traversed during decoding (*i.e.*, all probabilities for a given state could be lumped into a single number and passed to the node during message passing), the pairing node greatly enlarges amount of message passing that must be done over hidden states. Even so, the decode time goes from about one second for most sequences to about a minute per sequence (on a 4GHz PC with four cores). The memory requirements also dramatically increase as the intermediate probabilities for all the pairing node combinations must be saved.

8.5 VE: Comparison of accuracy

I now turn my attention to evaluating the performance of the *DBN* with virtual evidence. I run the classifier with and without co-training to investigate the improvements of just using prior knowledge about water use, versus leveraging the self-labeled instance pool. I use *DBN-VE* to denote the network without co-training and I use *Co-DBN-VE* to denote when the self-labeled pool is used. I also add another semi-supervised algorithm, the Transductive Support Vector Machine (*TSVM*) to compare as a baseline. As discussed, the *TSVM* uses the unlabeled data to place support vectors in areas of low entropy. I allow the *TSVM* to use instances from the highly confident self-labeled pool.

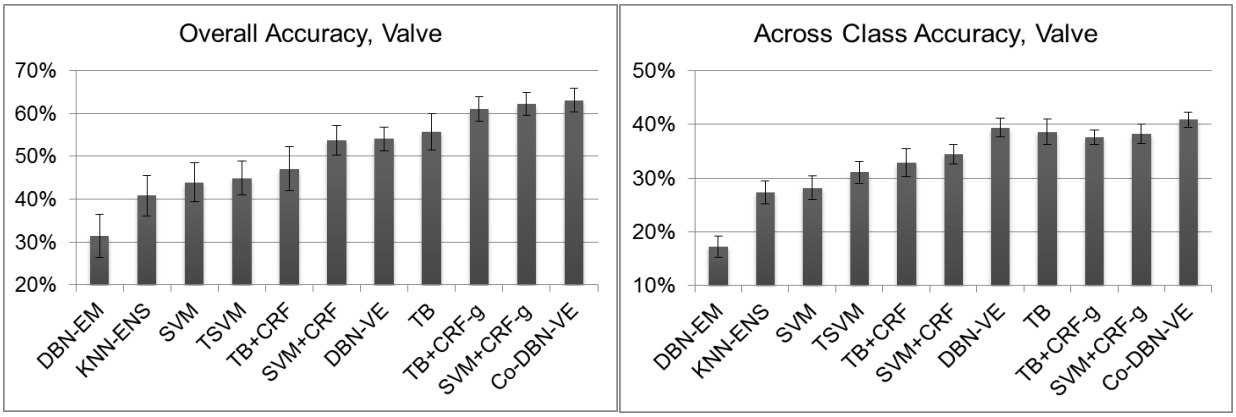

Figure 8-15. Baseline performances of overall and across class accuracy with the addition of DBN-VE and Co-DBN-VE

Explanation: Figure 8-15 shows the results at the *valve level* and the *across class accuracy* level for different algorithms. **Result**: Overall, the *DBN-VE* performs slightly worse than the bagged decision tree, *TB*. The best performing algorithm is *Co-DBN-VE*, but the accuracy is only 63% and is statistically the same as *SVM+CRF-g* when looking at overall accuracy. The across class accuracy of *DBN-VE* and *Co-DBN-VE* are the highest among all the algorithms. In terms of statistical significance, *Co-DBN-VE* is the best performing algorithm across classes. **Implication**: The first item to notice is that virtual evidence significantly boosts the accuracy of *DBN-EM*, bringing it on par with the overall accuracy of other classifiers. Next, the addition of the labeled instances is successfully leveraged with virtual evidence,

resulting in the best performing algorithm, especially when looking across classes. However, the overall accuracies are still low. Much lower than the 80-90% that I set out to achieve. Even so, the results from Figure 8-15 are deceiving. The across class accuracy of *DBN-VE* and *Co-DBN-VE* may only be about two percent better, but the fixtures and appliances within that 2% are by far the most water consuming valves in the home—namely the washing machine and showers. This is more apparent when I start to look at the fixture level and fixture category levels, Figure 8-16.

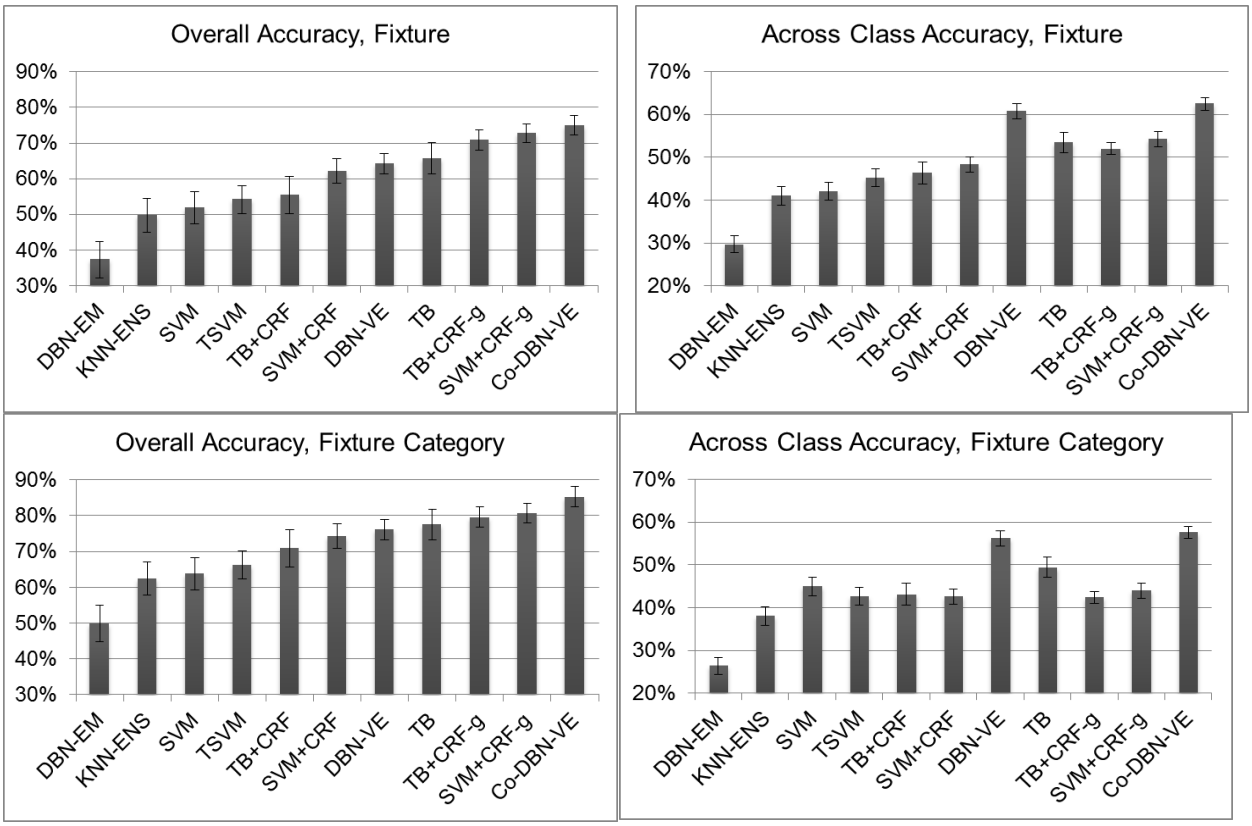

Figure 8-16. The overall and across class accuracies at the fixture and category levels

Explanantion/Result: The overall accuracy and across class accuracy are given at the fixture and category levels. *DBN-VE* and *Co-DBN-VE* are always statistically the best performers when looking at across class accuracy. At the fixture category level, *Co-DBN-VE* is statistically the best performer and has an across class accuracy that is nearly 10% higher than any other algorithm, except for *DBN-VE*.

Implication: The true advantage of using virtual evidence is that the across class accuracies at the fixture and category level are more distinguished than at the valve level. This is because, at the valve level, there are still a large number of temperature confusions. The information introduced by the temperature base bagged decision tree was not enough to sufficiently boost the performance of classifying valves. Analyzing the confusions from *Co-DBN-VE* (Figure 8-17) at the lumped fixture level, I can see

that the previous problems in not detecting the dishwasher, washing machine, or showers are starting to be mitigated, but are not completely solved. There are still a large number of confusions for the kitchen sink and "within bathroom" confusions. There are also a number of confusions between the dishwasher and kitchen sink, as well as the master bathroom shower and secondary bathroom shower.

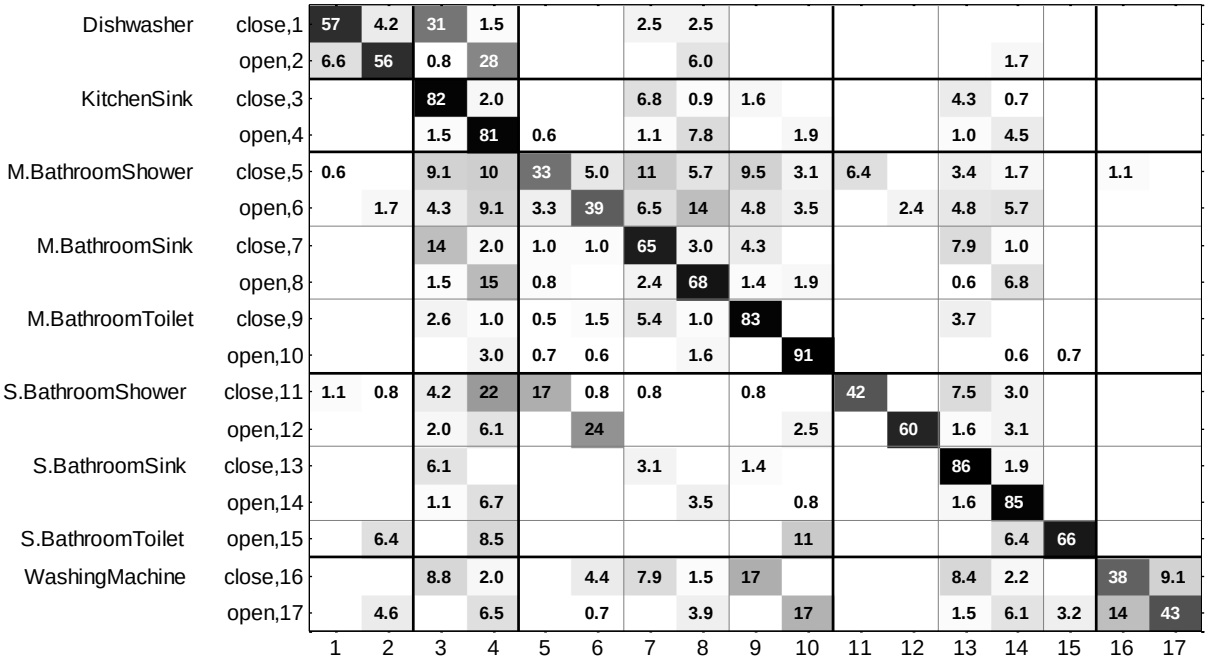

Figure 8-17. The lumped fixture level confusions for Co-DBN-VE using the minimal set of training instances

Implication: Despite the progress that virtual evidence has achieved in recognizing and labeling sparse classes, it is still not in the 80-90% range I set out to achieve. There are still many temperature confusions and confusions within the rooms where fixtures reside. I now investigate ways to leverage the behavior of the co-training classifier, together with virtual evidence to get into the 80% range. This includes adding a dimension from the home owner—selective journaling.

8.6 The human component: active learning through cooperative labeling

Up to this point, I have primarily relied on a few selected labels from the homeowner during one day of water usage. This, as explained, was to reduce the overhead of calibrating the system. However, there is no need for these labels to come from the same day—I have used sequence learning methods that learn their state transition probabilities from a global model, not a specific home. In this way, the training has been reduced to parameterizing the emission probabilities of the *DBN* and the *CRF* models. Labels can

come from any time during the installation. Moreover, the sparse codebook requires unlabeled instances in order to create its codebook—so the system would need to be primed with about a week of data before it could provide anything more finely grained than fixture category classification (or classification from the rule based classifier).

However, once I have the minimal set of examples, could I selectively ask the homeowner for examples of water usage using active learning? To be sure, leveraging the user should not be overly relied on. Asking too often about water usage can be perceived as "annoying" or, worse, may actually decrease the user confidence in the system because it exposes when the system is unsure about water events. If I ask too much, the homeowner is likely to be annoyed. If I don't ask at all, then the system can only achieve about 63% valve level accuracy, on average. The task, then, is to find a "good" method of asking for examples of natural water use, where I only choose examples that will truly help prime the classification. It is also important that I choose labels from natural water usage, not canned examples, so that I can capture the variability that the classifier has trouble exploiting.

Instead, I can leverage the ideas of active learning in stream based uncertainty sampling (as discussed in Chapter III). In particular I can combine methods from query by committee and confidence sampling with our semi-supervised co-trained classifiers. I know when the co-trained classifiers are not confident about a particular pressure wave and when they do not agree. Moreover, because one of the co-training classifiers is an instance classifier (the bagged decision tree), I can immediately know if the pressure wave classification is low confidence (rather than needing to wait for the sequence classifier to label an entire sequence). This means that I can message the homeowner on their mobile device seconds after the water is used. With such immediacy, it is likely that I can ask not only *what* water fixture was activated, but also if they know the *temperature setting* of the valve. In this way, the trust in the oracle for active learning can be maximized.

In the next experiment, I simulate the process of selectively asking for labels from the homeowner. It is not known what the perceived annoyance level of asking for labels will be; therefore I take a conservative approach. I never ask for more than four labels in a single day and, if I have asked for a label in the previous day, I do not allow the system to ask for examples in the current day. This is mild assurance that I am not over-asking for labels or asking for too many labels at a time. Moreover, I reduce the overhead of labeling by only asking for labels twice in the day—once in the morning and once in the evening. If the event I ask for in the morning is in a cluster of events, I will also randomly ask for a label to the preceding water usage event. In the same way, the event I ask for in the evening may be in a cluster

of events and, if so, I will ask for a label to preceding water usage. I define a preceding event as water usage within 30 seconds of the uncertain event.

To select the "low confidence" events, I first employ uncertainty sampling on each co-trained bagged decision tree. As discussed in Chapter III, this is known as margin sampling. In particular, I set a threshold on the confidence margin for each co-trained bagged decision. The first event in the day (used after 8AM) that has a margin between the most probable class and next most probable class that is less than 20%, may be queried. I also employ the query-by-committee approach because each classifier used in co-training must also disagree about the label. When the margin is greater than 20% and the committee disagrees, I ask the homeowner for a label. The same applies in the evening, where the event occurs after 2PM but before 9PM. This is to simulate that the system is asking for labels in a time frame that is not annoying to the homeowner. I then can retrain the *Co-DBN-VE* algorithm using these labels. I denote this particular process of active learning as "Co-Labeling" and the resultant classifier is denoted as *CoLabel-DBN*. Figure 8-18 shows the results of this simulation for H1.

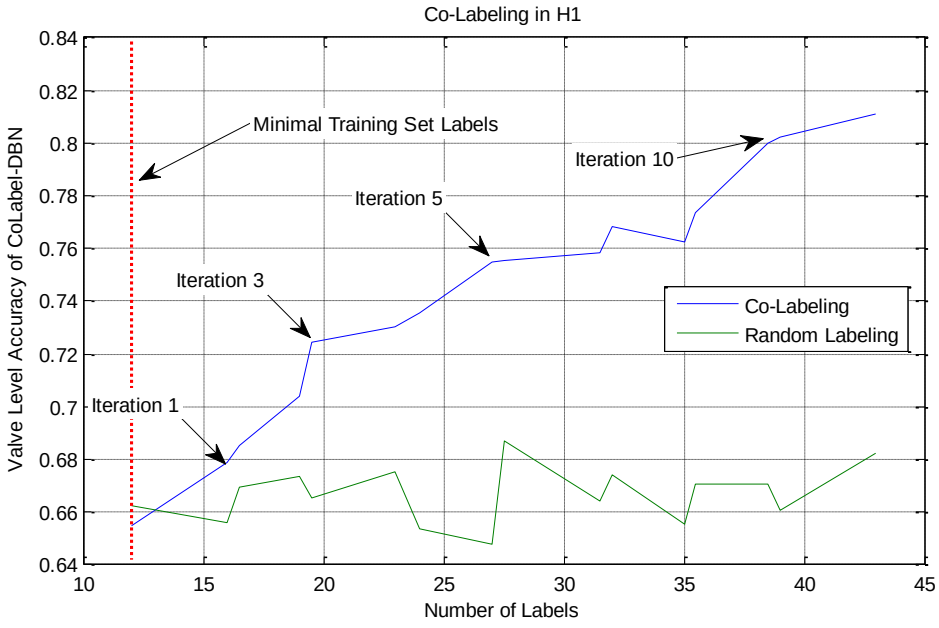

Figure 8-18. The accuracy of the system as I selectively ask the homeowner for more examples every other day for 22 days. At each iteration 2-4 labels are asked of the homeowner.

Explanation: Figure 8-18 shows the accuracy of the *CoLabel-DBN* trained with an increasing number of labels from the homeowner. The initial size of the labeled pool is 13 fixture events—which is the size of the minimal label set. At each iteration, 2-4 labels are asked of the homeowner on different days, as discussed. Also shown are the results for asking randomly for events from the homeowner instead of

using active learning to prime the system. **Result**: There is a constant and sustained improvement using active learning. However, random labeling does not significantly increase the accuracy of the overall system. After two iterations, the results for co-labeling have improved by 5% and after ten iterations; I have surpassed the 80% confidence mark I originally set out to achieve. **Implication**: Co-labeling with the DBN shows a good ability to prime the system quickly. Amazingly, with only 37 labeled fixture uses, I am able to achieve 80% valve level accuracy in H1, which has over 2,000 test instances.

It is of note that the active learning approach is independent of using virtual evidence in the DBN. One can use active learning with just a bagged decision tree, but the across class accuracy suffers greatly because the classifiers without virtual evidence tend to miss washing machines, dishwasher, and showers. Next, I evaluate the performance of *CoLabel-DBN* across all the residences.

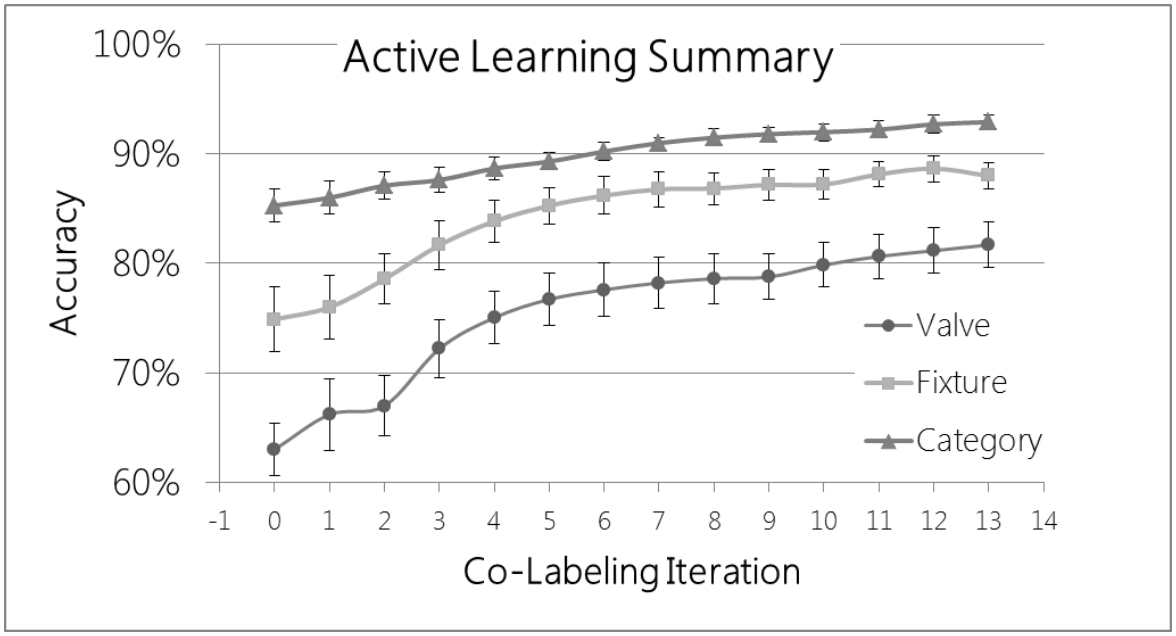

Figure 8-19. Accuracy of using active learning versus the co-labeling iteration. At each iteration, 2-4 labels are added to the pool

Explanation: Figure 8-19 shows the performance of *CoLabel-DBN* as I increase the labeling iteration. For each home, co-labeling simulation is carried out three times with a different set of minimal labels to seed the classification. At each iteration, 2-4 examples are added to the label pool from the homeowner. On average, ~3.2 labels are added at each iteration and it is most common to add four labels. Accuracies are shown at the valve, lumped fixture, and category levels. Error bars are the standard error across all homes and all trials. **Result**: All levels improve dramatically using active learning. For each level it takes three rounds of co-labeling for the results to be statistically better than the baseline performance (iteration 0). For valve level accuracy, it takes 10 iterations for the results to surpass the 80% marker. For fixture

level, it takes 3 iterations, and for the category level results are always above 80%, but surpass 90% at six iterations. **Implication**: Through active learning, I have finally reached the goal of 80% accuracy at the valve level. The ten iterations needed to surpass this marker correspond to 30-40 queries answered by the homeowner over a period of 20 days. This can be considered truly selective labeling. Moreover, after only three iterations, corresponding to 9-12 labels, the fixture level accuracy surpasses 80%. The most common label for the system to ask for is a hot or cold event from the kitchen or bathroom faucet, followed by the toilet. This advocates the importance of using the *DBN-VE* as a baseline classifier, because active learning only marginally increases the *diversity* of class examples. Even so, a shower example is typically asked for in the first two iterations, but washing machines and dishwashers are not asked for until typically the fifth or sixth iteration (if at all). This is not a problem, however, because the rule based classifier and *DBN-VE* are able to leverage prior knowledge in classifying these fixtures and appliances.

A final investigation of the confusions reveals that the system, after 10 iterations of co-labeling, is highly accurate among each fixture, although temperature confusions still exist (Figure 8-20). The most common confusion is the secondary bathroom shower for the master bathroom shower.

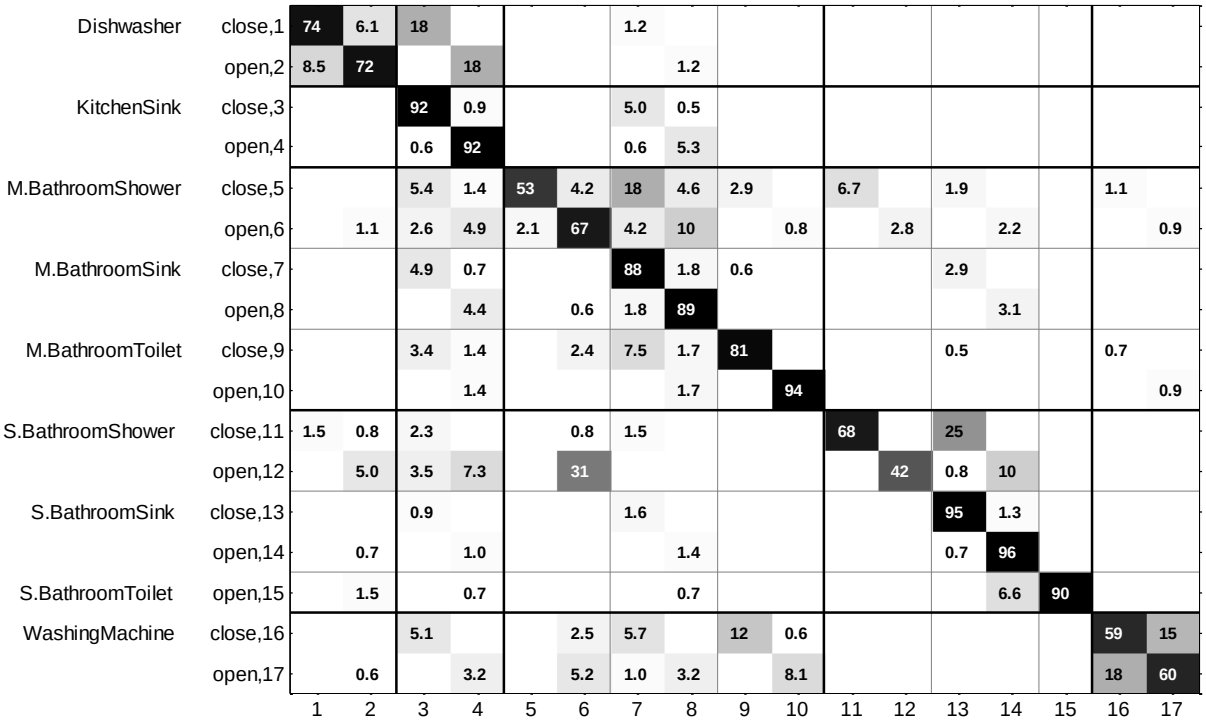

Figure 8-20. The final confusion matrix for the CoLabel-DBN algorithm

For comparison to the other algorithms, I also show the improvement in the across fixture accuracy at the valve, lumped fixture, and fixture category level, shown in Figure 8-21.

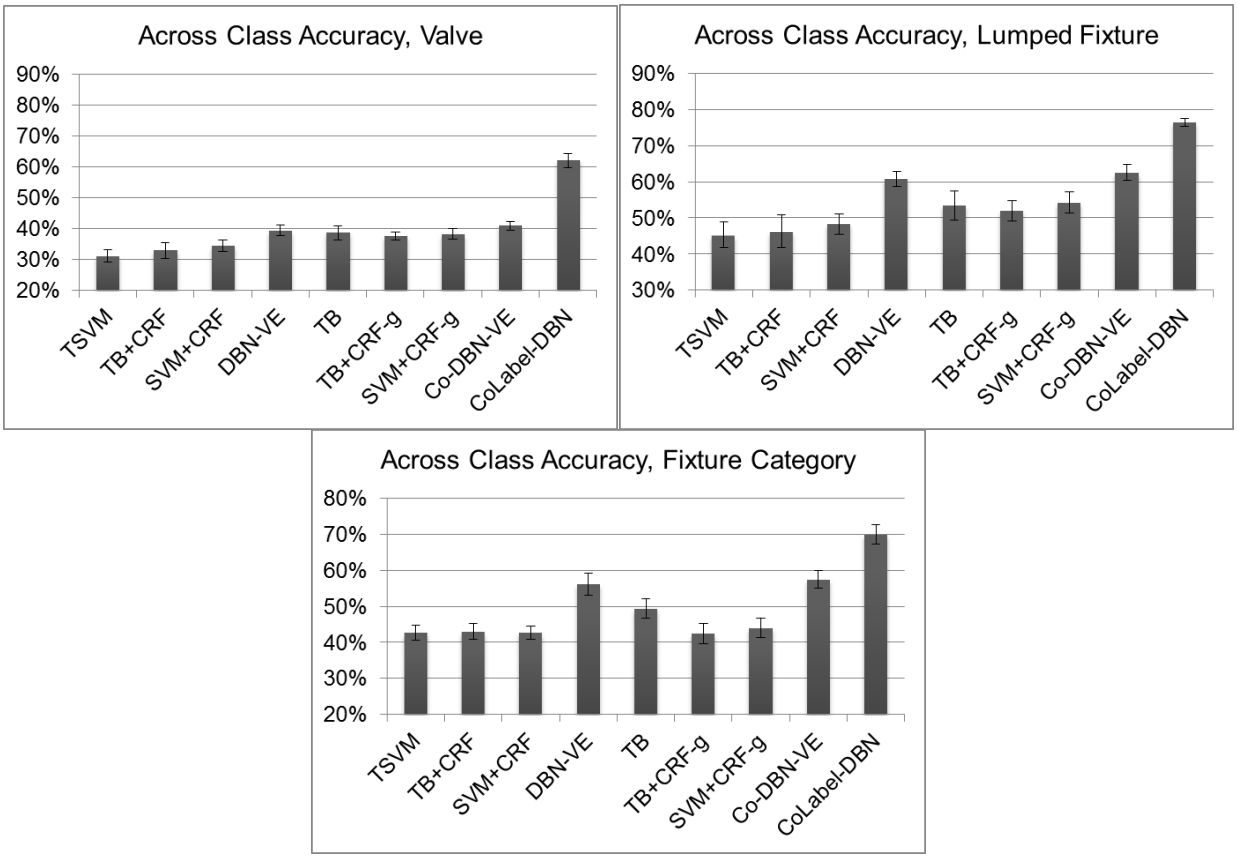

Figure 8-21. The comparative increase in across class accuracy that co-labeling provides over baseline classifiers

The biggest improvement occurs at the *valve level*, *across class accuracy*, which is truly starting to learn how to leverage labeled information to disaggregate hot versus cold versus mixed water usage over the entire array of fixtures and appliances in the home. For comparison, 10-fold cross validation results in across class accuracies of 75%, 91%, and 89% at the valve, lumped fixture, and fixture category levels. The co-labeling approach achieves 62%, 78%, and 70% across class accuracies, despite having $1/100^{th}$ of the amount of training data.

With these accuracies I can now characterize the entire calibration protocol for the HydroSense system and the feedback level that the homeowner can expect at different time periods. This is summarized in Table 8-2. The final Chapter of this thesis discusses the contributions and limitations of the approach presented here, placing perspective on the methodology in the context of different applications for different communities.

Time Period	Protocol	Expected Feedback
First Week	Homeowner provides 8-20 examples over the first week: • 1-2 Shower usages • 1 run of the dishwasher • 1 run of the laundry machine • 2 examples of each toilet • 1 example of hot and cold water use for each dual handle faucet • 1 example of hot, cold, and mixed water use for each single handle faucet (2 examples if in kitchen)	HydroSense relies on the rule based classifier for the first week. Pressure waves are saved in order to create a sparse codebook of features. Results are displayed at the fixture category for dishwashers, showers, and washing machines.
Start of Second Week	Homeowner provides 2-4 labels every other day when the system messages them on their mobile device	Results are displayed at the full fixture category level from the CoDBN-VE algorithm. Expected accuracy: • 85% at fixture category level
End of Second Week	Homeowner has supplied 9-12 examples that were flagged by active learning.	HydroSense now displays results at the Lumped Fixture level. Expected accuracy: • 82% at fixture level • 87% at fixture category level
End of Third Week	Homeowner continues to supply sparsely selected examples every other day. About 9-12 additional examples provided.	Valve level accuracy now provided. Expected accuracy: • 80% at valve level • 87% at fixture level • 92% at fixture category level
Fourth Week	Homeowner can optionally continue to provide examples to the system for increased accuracy.	Expected accuracy: • 81% at valve level • 89% at fixture level • 93% at fixture category level

Table 8-2. Expected feedback and calibration protocol for semi-supervised HydroSense system

8.7 Summary

In summary, I was able to successfully use co-training to attain a highly accurate self-labeled pool of instances from the home. Using this pool I was able to also create a number of rule based classification techniques for detecting sparsely used fixtures in the home. I then combined these approaches into a probabilistic framework using virtual evidence in a dynamic Bayesian network. Although this system vastly improved the ability of the system to classify many water usage events, the overall accuracy was not above the target of 80% valve level accuracy. I then used the multi-view classifier to selectively flag events for the homeowner to label. This active learning approach was able to drastically increase the accuracy of the system to above 80%. At this accuracy level, the system is ready for various field studies and for widespread adoption among interested homeowners trying to reduce their environmental footprint. Moreover, various other communities can leverage the system for activity detection and other sensing applications in the pervasive and ubiquitous computing fields, as discussed in the next chapter.

CHAPTER IX

9. Chapter Nine:
Contributions and Conclusions for Different Communities

This chapter summarizes and categorizes the contributions of this thesis in the broader context of different readership. In particular I discuss the advantages and limitations of using the methodology outlined in this thesis, using application areas from different communities. I also categorize the research questions I seek to answer in my career and the progress that this thesis makes toward answering those questions. The contributions of this thesis are outlined. I hope this chapter is a model for how machine learning, sensing, and the human-interaction communities can come together to solve a difficult problem with broad implications for other communities, such as health and sustainability.

9.1 Advantages and Limitations for Different Communities

This section breaks down the advantages and limitations of the approach for different interested communities. Specifically I discuss the methodology in the context of the water sensing community and general sustainability community, the activity detection and elder care community, and the machine learning and signal processing community. I also outline the core challenges remaining for this system or methods to be used widespread in different application areas.

9.1.1 The water sensing community and general sustainability community

The methodology for the system has the most concrete implications for the water sensing community. This community is largely made up of utilities, policy makers, regulators, and other government agencies. As discussed in Chapter I and II, these communities often need to assess the water usage behavior of large populations, especially as it pertains to the end-use of water and the end-use of hot water. For these communities the solution has tremendous advantages with little disadvantage because they are already devoting tremendous resources towards collecting the same type of data. The advantages are quite obvious in terms of cost and manpower. As such, I try to focus on the limitations of this approach for these communities and outline some of the potential challenges that need to be addressed.

The method I propose does require motivation from the homeowner in providing samples of water usage. However I have a distinct advantage over existing journaling techniques because I can automatically prompt the homeowner when needed. However, such a labeling infrastructure still needs to be created. I have shown through simulation that the process is promising; however I have not actually evaluated the reaction of consumers to this process. There are many research question still left to be researched. For instance, does asking for water usage labels actually affect the trust that users have in the system? If the system continually asks for examples of the same fixture, would the homeowner begin to believe that the system is "not good?" Or perhaps, the asking of labels would actually bring the homeowner closer to the system. Because I prompt the homeowner for examples, is it possible that they become more comfortable with the system in their everyday lives because they feel more comfortable with the inner workings of the system?

In my experience, many people are intrigued with the system when installed in their home. They tend to enjoy exploring what the system can or cannot sense. Asking for labels, then, could be a way of evolving people's trust in the system. For instance "the system missed my shower yesterday morning, but then it asked me what fixture I had just used—today it did not miss the event so it is obviously getting better." By the same token, continually getting an example of an event may also enhance people's understanding of the system. For instance "it said I used the kitchen sink way fewer times than I actually did. That's okay though, because I know it has trouble getting the sink—it always asks me for examples of it."

I have also not evaluated the effect of label noise from the homeowner. What happens if they periodically tell us an incorrect event? For instance, if I ask for a label and they tell us it was the kitchen sink hot instead of mixed water usage, will the label be detrimental to the system? What if they tell us the bath was a kitchen sink? From this perspective it may be more reliable to give the homeowner a prompt of what I think the event was they just used. Something perhaps like, "I think you just used the kitchen sink, is that right?" The label then could be "yes" or "no." If yes, then I add the template to the database. If no, then I ignore the event and move on without requiring the homeowner to provide a label. The idea here is that, if the homeowner does not need to navigate a menu, then they are less likely to provide an incorrect label. Moreover, the homeowner may be more likely to provide a yes/no through one click than they would be to provide a label via different menus. It also enables the technology to be used without a smartphone because the person could respond with a text message from a feature phone (perhaps a single "y" or "n" is all that would be needed).

Lastly, one aspect that I have not focused on is that the homeowner is not the only user of water in the home. The best approach is to ask the person using water what fixture they just activated—which I cannot know. To be sure, this problem is not insurmountable. I could ask all individuals in a home for a label when water is used—however this has some privacy concerns associated because it means a spouse, parent, or roommate would know when someone in the home used water. Another possibility is not to use a mobile device at all, but perhaps rely on a dedicated interface in the home. If water is used and the system is uncertain it rings out a quick ding or chime letting anyone near know that it needs a label.

9.1.2 The activity detection and elder care community: privacy

For individuals in this community, the main goal is not to know how much water is used in the home, but to know more about the habits and schedule of the inhabitants of a home. The idea is to learn when someone is cooking, entertaining, or going about their regular routine. The information provided by a water sensing system is certainly rich in terms of looking at someone's routine. However, it can only tell you so much about their actions. For instance it may be impossible to tell the difference between washing your hands versus washing your face—or it may not be possible to distinguish brushing teeth from shaving (*i.e.,* many individuals use the hot water tap repeatedly to rinse their toothbrush). In these situations the system may not provide the direct information that relates to activity. However, the system could be used in conjunction with other distributed sensing systems to provide higher level information.

For instance, if I combined HydroSense with the popular learning thermostat system, the Nest, I can immediately provide knowledge about the temperature of the shower in the morning versus the temperature in the home. Optimizations could then be applied to the hot water heater and thermostat to optimize the energy usage tradeoff between shower temperature and in home cooling or heating.

In the field of elder care, providing labels for the system to learn may be more difficult because older populations typically need specially designed systems that cater to their previous experiences with technology. For instance, assuming they have a mobile device is no longer warranted. When using a labeling system with geriatric populations, I may need periodic labels from a caregiver. This, however, highlights another privacy concern of the system—many individuals consider their water use private. Water use, of course, occurs around sensitive, private scenarios such as bathing and toileting. Even though the sensor is not a camera or more invasive technology, it may still be difficult for individuals to decouple the privacy of these moments with the sensing of these activities.

Even so, the value of this technology needs to be weighed against its privacy implications. For instance, the detection of right side congestive heart failure may be possible using a system like

HydroSense. This is because one of the first indications of heart failure is frequent urinations during the night time. By looking at toileting patterns, I may be able to flag the early detection of the disease.

Another issue for privacy is in apartment complexes. In the deployment, A1 was also able to sense water usage in surrounding apartments. Could I ascertain what these other participants were doing based on the pressure waves? If their pressure waves are similar enough to those learned in your apartment, it is a possibility. The privacy concern is not just for "spying" on one's neighbor, but could perhaps be more benign. Suppose an inference engine ascertains that you cooked dinner last night for six people, but, in actuality, it was inferring the dinner party of your neighbor. In this case, the homeowner never meant to collect this information, but received it nonetheless.

9.1.3 The machine learning and signal processing community

I now switch contexts from the application of the technology to the methods used. The methodology I have created combines the use of expertly chosen feature selection, sparse codes, co-training, and virtual evidence. Of particular interest to the community is the use of sparse codes and expert features with co-training. In many instances, the use of co-training is difficult because a natural split in the features is not possible to get two independent views of the problem. By combining expert knowledge with sparse codes, I get two complementary views of a problem—the most necessary requirement for PAC learning methods such as co-training. To the best of my knowledge, this has never been performed before.

Other application spaces may also be able to leverage this method. However, it is not guaranteed to produce independent views of the data. Even in this application, there was enough independence in the knowledge of each view to use co-training to add information to the problem space. That is, the new labels I added were, in some way, similar to the minimally labeled set of examples. The addition of new labels did not add new information about the problem. But the confidence in the labels that were added was quite high. To quote Supreme Court Justice Potter Stewart, "I can't define it … but I know it when I see it." The classifier I built is identical in its knowledge. I cannot define other examples as being close to the actual class label, but I know that label when it occurs.

Secondly, I am not aware of any other work combining virtual evidence and semi-supervised learning. The use of co-training to attain high confidence labels, then incorporating this knowledge with virtual evidence to inform the selection of other class labels near in the time series was a successful approach. Certainly other time series applications can benefit from this paradigm. For example, certain speech and language applications could benefit from knowing the class assignment of nearby words or phrases. The use of expert knowledge around these high confidence labels is also easy to apply.

9.2 Research Questions in my Career

Through my research I seek to answer three questions:

Can a framework using semi-supervised techniques be made sufficiently accurate and simple so that it can be adopted by non-experts in machine learning? Non-experts often follow the *path of least resistance* when employing statistical machine learning [120,121]. Tuning parameters and understanding methods for optimization can be giant hurdles for adoption. Having a straightforward, versatile method for time series classification with only a few labels is needed in a range of disciplines [102], and could result in a number of unforeseen advancements. This thesis contributes to this field by incorporating black box feature extraction methods, such as sparse coding, with expert knowledge through virtual evidence.

How can technology create and support a sustainable lifestyle? We are using resources faster than we can replenish them. Sustainability through efficiency is a necessary challenge and promises to be one of the grand challenges of our and future generations. Tools that support sustainable lifestyle, along with advancements in other fields like renewables, must be actively researched and evaluated. This technology is one building block in my research towards this goal.

Can we leverage human behavior with infrastructure mediated sensing to create an unobtrusive, context aware system? I want to understand more fully how to incorporate prior knowledge of human behavior with multiple sources of information, such as sensors and processed features over time. I ultimately want to do more than just classify devices and fixtures, and move towards understanding the context of activity and usage. Such advancements are the core vision of ubiquitous computing and have not yet been realistically realized [1].

Many of the specifics of these research questions will be answered throughout my career. However, this thesis is the first step in addressing the difficult problems associated with these research questions. I now review the contributions of this thesis, categorized into three sets: core advancements in machine learning and ubiquitous computing, core advancements in sustainability, and peripheral impacts. The core advancements pertain to immediate results and concrete innovations pertaining to (a) machine learning and ubiquitous computing, and (b) to water conservation and sustainability. Peripheral impacts pertain to advancements and contributions that are byproducts of the core research.

9.2.1 Core advancements in machine learning and ubiquitous computing

The core advancements are direct outcomes of this thesis and contribute to the ultimate goals of my research in machine learning and ubiquitous computing. They include:

1. An improved training method for semi-supervised learning for time series classification that is:
    a. able to leverage co-training, expert knowledge, and virtual evidence in its decision making process,
    b. can incorporate different knowledge sources from different data sets,
    c. allows changes in confidences from an expert in the field without retraining the entire system
    d. easily combined with techniques of active learning to further minimize the number of labeled training examples
2. An evaluation of performance of using a few labels on real world HydroSense data streams, specifically investigating:
    a. co-training methods,
    b. the tradeoff between required number of unlabeled instances and accuracy,
    c. convergence time of the system,
    d. the schedule of calibration for the system,
    e. the infusion of different knowledge sources through sparse coding and expertly selected features,
    f. a strategy for active labeling that quickly boost the performance of the classifier, without placing an undue burden on the homeowner

9.2.2 Core advancements in water conservation and sustainability

The other core advancements pertain to the sustainability community, providing

3. Larger and more comprehensive studies of the end uses of water. I consider these the "first adopters" of this type of technology. A semi-supervised, low cost framework allows institutions and companies to conduct larger studies, faster. This allows the evaluation of wasteful practices, rebate policies and incentives, outreach campaigns, and comparative studies upon different groups (*i.e.*, different socio-economics, geography, country, *etc.*) [7,27,160].
4. More comprehensive control and understanding of hot water use [134]. By incorporating a technology like HydroSense, one can more adequately control when and how to manage water heating, the second highest consumer of electricity in the US [38]. In a recent forum of hot water use, this was stated as the most relevant need for creating policies in support of sustainability and water management (Hot Water Forum, 2011, 2012).

5. A dataset of ground truth hot and cold water end uses available for the water resource community. Although I use this to evaluate my methods, the ground truth labels are of immediate utility to many researchers in sustainability and water management.
6. Awareness in everyday life, providing a scalable method for monitoring water usage through eco-feedback displays. HydroSense also has the ability to further research in the eco-feedback community by providing disaggregated feedback that can be used in deployments comparing different incentive structures, displays, and design goals.

9.2.3 Peripheral Impacts

The research carried out here has a number of advancements in fields outside of signal processing, machine learning, and sustainability. This research also has the promise to impact:

7. The health community, providing
    a. A means to monitor the everyday activities for detecting the onset of dementia [47]
    b. A means for elder care monitoring, such as logging daily activities, cooking, *etc*. without more intrusive sensors (*e.g.*, a camera)
    c. A means of monitoring hygiene
8. Other domains, providing
    a. A toolkit that can go well beyond classification of water usage. The technology could further advancements in many other domains. I see some of the first adopters for the toolkit as:
        i. Other IMS systems (such as disaggregated electricity and gas sensing [35,64]),
        ii. on-body sensing [36,37],
        iii. and analytics (*i.e.*, classifying types of events in a Facebook or twitter streams [40])

9.3 Conclusions and Future Work

This dissertation has provided an interdisciplinary link in the fields of sustainability, machine learning, and ubiquitous computing. I was able to take a simple idea from inception to a sensing paradigm that is ready for adoption. Large scale water end use studies are now possible with fewer costs. Wide scale adoption of eco-feedback initiatives can be launched with this technology underpinning. The practicality of the system pivoted on its ability to learn quickly from only a few labeled examples. Through semi-supervised learning, virtual evidence, and active learning, the approach is now ideally suited for

widespread use. These same techniques can also be applied in other sustainability applications—for instance in the fields of electricity and natural gas sensing—or in many other IMS systems not directly related to sustainability. Through this work, I hope that our environmental footprint can begin downsizing—perhaps one day it will no longer be inevitable that we will deplete our water supply.

There are a number of opportunities for future work. As with any research, the answering of questions has created new questions. Because of the interdisciplinary nature of this topic, there are a number of new questions across fields. The approach evaluated for active learning may not only be a good tool for minimizing calibration, but may also be an efficient means of accounting for sensor drift. The longitudinal deployment uncovered several features with values correlated with time. It may be helpful, then, to take this sensor drift into account when selectively asking for new labels. For example, if we have seen a particular feature in the master bathroom drift substantially, it may be more pertinent to actively ask the homeowner for examples from the master bathroom. The process of accounting for this drift, and effectively integrating with selective sampling, remains an open research topic.

In the health space, this technology could have far reaching impacts. However, it still needs further investigation to ascertain whether water sensing and health sensing can be effectively integrated. A good start would be linking daily activity logs and water sensing. Or, investigating how accurately activity inference can be achieved with a technology like HydroSense. Screening for congestive heart failure and overactive bladder in retirement communities may also provide significant benefit.

Finally, more eco-feedback water studies and water end use studies are the obvious next steps with this technology. It makes large scale deployments within the reach of a number of different communities. Although the core sensing research is mature, the selection of feedback mechanisms and evaluation of these mechanisms remains a pervasive topic: what is the best way to increase awareness and incentivize efficiency? With this technology, I hope that question can now be explored in depth.

10. Bibliography

1. Abowd, G. What next, Ubicomp? Celebrating an intellectual disappearing act. *Proceedings of the 14th international comnference on ubiquitous computing*, (2012).

2. Aguilar, C., White, D.J., and Ryan, D. Domestic Water Heating and Water Heater Energy Consumption in Canada. *Canadian Building Energy End-Use Data and Analysis Centre.*, (2003), 343–352.

3. Aharon, M., Elad, M., and Bruckstein, A. K-SVD: Design of dictionaries for sparse representation. *Proceedings of SPARS 5*, (2005), 9–12.

4. Atlas, L., Cohn, D., Ladner, R., El-Sharkawi, M.A., and Marks II, R.J. *Training connectionist networks with queries and selective sampling*. Morgan Kaufmann Publishers Inc., 1990.

5. Baraniuk, R.G. Compressive Sensing. *Signal Processing Magazine, IEEE 24*, 4 (2007), 118–121.

6. Beal, C., Stewart, R., and Huang, A. *South East Queensland Residential End Use Study: Baseline Results*. 2010.

7. Beal, C., Stewart, R. a., Spinks, A., and Fielding, K. Using smart meters to identify social and technological impacts on residential water consumption. *Water Science & Technology: Water Supply 11*, 5 (2011), 527.

8. Beal, C., Stewart, R.A., Huang, T., and Rey, E. SEQ residential end use study. March (2011), 80–84.

9. Beal, C.D., Stewart, R. a., and Fielding, K. A novel mixed method smart metering approach to reconciling differences between perceived and actual residential end use water consumption. *Journal of Cleaner Production*, (2011), 1–13.

10. Beckmann, C., Consolvo, S., and Lamarca, A. Some assembly required: Supporting end-user sensor installation in domestic ubiquitous computing environments. *Proceedings of the International Conference on Ubiquitous Computing (UbiComp 2004*, Springer-Verlag (2004), 107–124.

11. Bilmes, J. and Geoff Zweig. The Graphical Models Toolkit: An open source software system for speech and time-series processing. *Proc. IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing*, (2002).

12. Bilmes, J. *On Virtual Evidence and Soft Evidence in Bayesian Networks*. 2004.

13. Blasius, H. Das Ahnlickkeitsgesetz bei Reibungsvorgangen in Flussigkeiten Forshungheft. (1911), 131.

14. Blei, D.M., Ng, A.Y., and Jordan, M.I. Latent dirichlet allocation. *J. Mach. Learn. Res. 3*, (2003), 993–1022.

15. Blum, A. and Mitchell, T. Combining labeled and unlabeled data with co-training. *Proceedings of the eleventh annual conference on Computational learning theory*, ACM (1998), 92–100.

16. Bo, L., Ren, X., and Fox, D. Hierarchical matching pursuit for image classification: Architecture and fast algorithms. *Advances in Neural Information Processing Systems 24*, (2011).

17. Bo, L., Ren, X., and Fox, D. Learning Hierarchical Sparse Features for RGB-D Object Recognition. *International conference on pervasive computing*, (2012).

18. Bohlig, C. *Water Utility Overview, Water Smart*. San Fransisco, 2012.

19. Bouchard, G. and Triggs, B. The Tradeoff Between Generative and Discriminative Classifiers. *16th IASC International Symposium on Computational Statistics (COMPSTAT '04)*, (2004), 721–728.

20. Breiman, L. Random forests. *Machine learning*, (2001), 1–35.

21. Brown, J.C. Calculation of a constant Q spectral transform. *The Journal of the Acoustical Society of America 89*, 1 (1991), 425.

22. Brown, J.C. An efficient algorithm for the calculation of a constant Q transform. *The Journal of the Acoustical Society of America*, (1992), 2698–2701.

23. Brumitt, B., Meyers, B., Krumm, J., Kern, A., and Shafer, S. EasyLiving : Technologies for Intelligent Environments. *Proceedings of the International Symposium on Handheld and Ubiquitous Computing*, (2000).

24. Burges, C.J.C. and Platt, J.C. Semi-Supervised Learning with Conditional Harmonic Mixing. 2006.

25. Byrd, R., Lu, P., Nocedal, J., and Zhu, C. A limited memory algorithm for bound constrained optimization. *SIAM Journal on Scientific Computing*, (1995).

26. Candès, E.J., Romberg, J., and Tao, T. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *Information Theory, IEEE Transactions on 52*, 2 (2006), 489–509.

27. Carragher, B.J., Stewart, R. a., and Beal, C.D. Quantifying the influence of residential water appliance efficiency on average day diurnal demand patterns at an end use level: A precursor to optimised water service infrastructure planning. *Resources, Conservation and Recycling 62*, (2012), 81–90.

28. Chen, F., Dai, J., Wang, B., and Sahu, S. Activity analysis based on low sample rate smart meters. *KDD 2011*, (2011).

29. Chen, J., Kam, A., Zhang, J., Liu, N., and Shue, L. Bathroom activity monitoring based on sound. *Pervasive Computing*, (2005), 47–61.

30. Chen, S. and Goodman, J. An empirical study of smoothing techniques for language modeling. *Proceedings of the 34th annual meeting on language modeling*, (1996), 310–318.

31. Cheung, E. Municipal Water Meter Monitor. 2003. http://www.edcheung.com/automa/water.htm.

32. Chow, Y. and Schwartz, R. The n-best algorithm: An efficient procedure for finding top n sentence hypotheses. *Proceedings of the workshop on Speech and Natural Language Processing*, (1989), 199–202.

33. Cohen, W.W. *Stacked sequential learning*. 2005.

34. Cohn, D., Atlas, L., and Ladner, R. Improving generalization with active learning. *Machine Learning 15*, 2 (1994), 201–221.

35. Cohn, G., Gupta, S., Froehlich, J., Larson, E.C., and Patel, S. GasSense: Appliance-level, Single-point Sensing of Gas Activity in the Home. *Proceedings of the 8th International Conference on Pervasive Computing (Pervasive 2010)*, Springer (2010), 265–282.

36. Cohn, G., Gupta, S., Lee, T., et al. An Ultra-Low-Power Human Body Motion Sensor Using Static Electric Field Sensing. *Proceedings of the 14th International Conference on Ubiquitous Computing*, (2012).

37. Cohn, G., Morris, D., Patel, S., and Tan, D. Humantenna: Using the Body as an Antenna for Real-Time Whole-Body Interaction. (2012).

38. DOE. Water Heating. 2012. http://www.energysavers.gov/your_home/water_heating/index.cfm/mytopic=12760.

39. Dagan, I. and Engelson, S.P. Committee-based sampling for training probabilistic classifiers. *Machine Learning International Workshop and Conference*, (1995), 150–157.

40. Dann, S. Twitter content classification. *First Monday*, (2010).

41. Day, N.E. Estimating the components of a mixture of normal distributions. *Biometrika 56*, 3 (1969), 463–474.

42. DeOreo, W.B., Heaney, J.P., and Mayer, P.W. Flow Trace Analysis to Assess Water Use. *Journal of the American Water Works Association 8*, 1 (1996), 79–90.

43. DeOreo, W.B. and Mayer, P.W. Project Report: A Process Approach for Measuring Residential Water Use and Assessing Conservation Effectiveness. *Project Report: A Process Approach for Measuring Residential Water Use and Assessing Conservation Effectiveness*, (1994).

44. DeOreo, W.B. and Mayer, P.W. *The End Uses of Hot Water in Single Family Homes from Flow Trace Analysis.* 2000.

45. Dempster, A. and Laird, N. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society.*, (1977).

46. Deveaux, R. Applied Smoothing Techniques for Data Analysis. *Technometrics*, (1999).

47. Dewey, M.E. and Copeland, J.R.M. Diagnosis of dementia from the history and aetiology schedule. *International Journal of Geriatric Psychiatry 16*, 9 (2001), 912–917.

48. Didaci, L. and Roli, F. Using co-training and self-training in semi-supervised multiple classifier systems. *Structural, Syntactic, and Statistical Pattern Recognition*, (2006), 522–530.

49. Domingos, P. A few useful things to know about machine learning. *Communications of the ACM 55*, 10 (2012), 78.

50. Druck, G., Mann, G., and McCallum, A. Learning from labeled features using generalized expectation criteria. *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, ACM (2008), 595–602.

51. Druck, G. *Generalized Expectation Criteria for Lightly Supervised Learning*. University of Massachusetts Amherst, 2011.

52. Dziegielewski, B., Opitz, E., Kiefer, J., and Baumann, D. Evaluation of Urban Water Conservation Programs: A Procedures Manual. *Prepared for California Urban Water Agencies by Planning and Management Consultants*, (1992).

53. Edison Thomaz, Bettadapura, V., Reyes, G., et al. Recognizing Water-Based Activities in the Home Through Infrastructure-Mediated Sensing. *International Conference on Ubiquitous Computing*, (2012).

54. Fabio and Cohen. Risks of semi-supervised learning: How unlabeled data can degrade performance of generative classifiers. *Semi-Supervised Learning*, (2006).

55. Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. LIBLINEAR: A Library for Large Linear Classi cation. *Journal of Machine Learning Research 9*, (2008), 1871–1874.

56. Flint, R. The sustainable development of water resources. *Journal of Contemporary Water Research*, Adler 2002 (2010), 41–51.

57. Fogarty, J., Au, C., and Hudson, S.E. Sensing from the basement: a feasibility study of unobtrusive and low-cost home activity recognition. *Proceedings of the 19th annual ACM symposium on User interface software and technology*, ACM (2006), 91–100.

58. Froehlich, J., Findlater, L., Ostergren, M., et al. The Design and Evaluation of Prototype Eco-Feedback Displays for Fixture-Level Water Usage Data. *Proceedings of the 2012 Annual Conference on Human Factors in Computing Systems (CHI 2012)*, (2012).

59. Froehlich, J., Larson, E.C., Campbell, T., Haggerty, C., Fogarty, J., and Patel, S.N. HydroSense: Infrastructure-mediated Single-point Sensing of Whole-home Water Activity. *Proceedings of the 11th ACM International Conference on Ubiquitous Computing (UbiComp 2009)*, (2009), 235–244.

60. Froehlich, J., Larson, E.C., Gupta, S., Cohn, G., Reynolds, M.S., and Patel, S.N. Disaggregated End-use Energy Sensing for the Smart Grid. *IEEE Pervasive Computing*, (2010), 28–39.

61. Gale, W.A., Laboratories, T.B., and Hill, M. Good-Turing smoothing without tears. *Journal of Quantitative Linguistics*, (1995), 1–24.

62. Grandvalet, Y. and Bengio. Semi-supervised learning by entropy minimization. *CAP 2005*, (2005).

63. Gupta, S., Chen, K.-Y., Reynolds, M.S., and Patel, S.N. LightWave: using compact fluorescent lights as sensors. *Proceedings of the 13th international conference on Ubiquitous computing*, ACM (2011), 65–74.

64. Gupta, S., Reynolds, M.S., and Patel, S.N. ElectriSense: single-point sensing using EMI for electrical event detection and classification in the home. *Proceedings of the 12th ACM international conference on Ubiquitous computing*, ACM (2010), 139–148.

65. Helal, S., Mann, W., El-Zabadani, H., King, J., Kaddoura, Y., and Jansen, E. The Gator Tech Smart House: a programmable pervasive space. *Computer 38*, 3 (2005), 50–60.

66. Henze, G.., Tiller, D.K., Fischer, M., and Rieger, M. Comparison of Event Inference and Flow Trace Signature Methods for Hot Water End Use Analysis,. *American Society of Heating, Refrigerating, and Air-Conditioning Engineers 108*, 2 (2002), 467–479.

67. Hiller, C.C. New Hot Water Consumption Analysis and Water-Heating System Sizing Methodology. *American Society of Heating, Refrigerating, and Air-Conditioning Engineers SF-98-31-3*, (1998), 1864–1877.

68. Hirsch, T., Forlizzi, J., Hyder, E., Goetz, J., Stroback, J., and Kurtz, C. The ELDer Project : Social and Emotional Factors in the Design of Eldercare Technologies. *Proceedings of the ACM Conference on Universal Usability*, (2000), 1–8.

69. Ho, T.K. The Random Subspace Method for Constructing Decision Forests. *IEEE Transactions Pattern Analysis Machine Intelligence 20*, 8 (1998), 832–844.

70. Hwa, R. On minimizing training corpus for parser acquisition. *Proceedings of the 2001 workshop on Computational Natural Language Learning-Volume 7*, (2001), 10.

71. Itakura, F. and Saito, S. Analysis synthesis telephony based on the maximum likelihood method. *Proc. 6th of the International Congress on Acoustics*, (1968), C–17–20.

72. Jelinek, F. Continuous speech recognition by statistical methods. *Proceedings of the IEEE 64*, 4 (1976), 532–556.

73. Jiao, F., Wang, S., and Lee, C.-H. Semi-supervised conditional random fields for improved sequence segmentation and labeling. *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational*, (2006), 209–216.

74. Joachims, T. *Text categorization with support vector machines: Learning with many relevant features*. 1998.

75. Joachims, T. Transductive Inference for Text Classification using Support Vector Machines. *Proceedings of the Sixteenth International Conference on Machine Learning*, Morgan Kaufmann Publishers Inc. (1999), 200–209.

76. Kabal, P. and Ramachandran, R.P. The computation of line spectral frequencies using Chebyshev polynomials. *Acoustics, Speech and Signal Processing, IEEE Transactions on 34*, 6 (1986), 1419–1426.

77. Katz, S.M. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, (1987), 400–401.

78. Keerthi, S.S. and Sundararajan, S. *CRF versus SVM-Struct for Sequence Labeling*. 2009.

79. Kiefer, J. Sequential minimax search for a maximum. *Proceedings of the American Mathematical Society 4*, 3 (1953), 502–506.

80. Kim, Y., Schmid, T., Charbiwala, Z.M., Friedman, J., and Srivastava, M.B. NAWMS : Nonintrusive Autonomous Water Monitoring System. *Sensys 2008*, (2008).

81. Klein, D. and Manning, C.D. Corpus-based induction of syntactic structure: models of dependency and constituency. *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, Association for Computational Linguistics (2004).

82. Körner, C. and Wrobel, S. Multi-class ensemble-based active learning. In *Machine Learning: ECML 2006*. Springer, 2006, 687–694.

83. Ladd, G.O. and J.L. Harrison. Electric water heating for single-family residences: Group load research and analysis. *EPRI EA-40006.*, (1985).

84. Lafferty, J., McCallum, A., and Pereira, F. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. *Proc. Int. Conf. on Machine Learning*, (2001).

85. Larson, E.C., Froehlich, J., Campbell, T., et al. Disaggregated Water Sensing from a Single, Pressure-based Sensor: An Extended Analysis of HydroSense Using Staged Experiments. *Pervasive and Mobile Computing*, (2010).

86. Larson, E.C., Froehlich, J., Saba, E., et al. A Longitudinal Study of Pressure Sensing to Infer Real-World Water Usage Events in the Home. *Proceedings of the 9th International Conference on Pervasive Computing (Pervasive 2011)*, (2011), 50–69.

87. Larson, E.C., Goel, M., Boriello, G., Heltshe, S., Rosenfeld, M., and Patel, S.N. SpiroSmart: Using a Microphone to Measure Lung Function on a Mobile Phone. *Proceedings of the 14th ACM International Conference on Ubiquitous Computing (UbiComp 2012)*, (2012).

88. Larson, E.C., Lee, T., Liu, S., Rosenfeld, M., and Patel, S.N. Accurate and Privacy Preserving Cough Sensing using a Low-Cost Microphone. *Proceedings of the 13th ACM International Conference on Ubiquitous Computing (UbiComp 2011)*, (2011).

89. Lewis, D.D. and Catlett, J. Heterogeneous uncertainty sampling for supervised learning. *Proceedings of the eleventh international conference on machine learning*, (1994), 148–156.

90. Lewis, D.D. and Gale, W.A. A sequential algorithm for training text classifiers. *SIGIR 94*, (1994), 3–12.

91. Li, X. On the use of virtual evidence in conditional random fields. *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing Volume 3 - EMNLP '09*, Association for Computational Linguistics (2009), 1289.

92. Liang, P. and Jordan, M.I. An asymptotic analysis of generative, discriminative, and pseudolikelihood estimators. *Proceedings of the 25th international conference on Machine learning*, ACM (2008), 584–591.

93. Liao, L., Choudhury, T., Fox, D., and Kautz, H. Training conditional random fields using virtual evidence boosting. *Proc. of the International Joint Conference on Artifical Intelligence*, (2007), 2530–2535.

94. Liaw, A. and Wiener, M. Classification and Regression by randomForest. *R news 2*, December (2002), 18–22.

95. Lim, B. and Dey, A. Investigating intelligibility for uncertain context-aware applications. *Proceedings of the 13th international conference on context aware applications*, (2011), 415.

96. Liu, Y. Active learning with support vector machine applied to gene expression data for cancer classification. *Journal of chemical information and computer sciences 44*, 6 (2004), 1936–1941.

97. Lowenstein, A. and Hiller, C.C. Disaggregating Residential Hot Water Use: Part I. *merican Society of Heating, Refrigerating, and Air-Conditioning Engineers Transactions: Symposia AT-96-18-1*, (1996), 1019–1027.

98. Lowenstein, A. and Hiller, C.C. Disaggregating Residential Hot Water Use: Part II. *American Society of Heating, Refrigerating, and Air-Conditioning Engineers (SF-98-31-2*, (1998), 1852–1863.

99. Mahdaviani, M. Semi-supervised and active training of conditional random fields for activity recognition. *cs.ubc.ca*, 2007. http://www.cs.ubc.ca/~maryam/MaryamMahdaviani_Thesis.pdf.

100. Maimon, O. and Rokach, L. *The Data Mining and Knowledge Discovery Handbook*. Springer Science+Business media, 2005.

101. Malouf, R. A comparison of algorithms for maximum entropy parameter estimation. *Proceedings of the sixth conference on natural language processing 20*, (2002), 1–7.

102. Mann, G.S. and McCallum, A. Generalized expectation criteria for semi-supervised learning with weakly labeled data. *The Journal of Machine Learning Research 11*, (2010), 955–984.

103. Mayer, P., DeOreo, W., Kiefer, J., Opitz, E., Dziegieliewski, B., and Nelson, J. Residential End Uses of Water. *American Water Works Association*, (1999).

104. Mayer, P.W. Residential Water Use and Conservation Effectiveness: A Process Approach. 1995.

105. McCallum, A., Nigam, K., and others. Employing EM in pool-based active learning for text classification. *Proceedings of ICML-98, 15th International Conference on Machine Learning*, (1998), 350–358.

106. McKeown, K., Thadani, K., and Wang, W. Identifying Event Descriptions using Co-training with Online News Summaries. (2011).

107. McLoughlin, I.V. Line spectral pairs. *Signal Processing 88*, 3 (2008), 448–467.

108. Merialdo, B. Tagging English text with a probabilistic model. *Comput. Linguist. 20*, 2 (1994), 155–171.

109. Milgram, J., Cheriet, M., Sabourin, R., and others. One Against One or One Against All: Which One is Better for Handwriting Recognition with SVMs? *Tenth International Workshop on Frontiers in Handwriting Recognition*, (2006).

110. Miller, S., Guinness, J., and Zamanian, A. Name tagging with word clusters and discriminative training. 2004. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.105.9395.

111. Minka, T. Discriminative Models, Not Training. 2005. http://research.microsoft.com/apps/pubs/default.aspx?id=70229.

112. Minka, T.P. A comparison of numerical optimizers for logistic regression. *Unpublished draft 2003*, 4 (2003), 1–18.

113. Muslea, I., Minton, S., and Knoblock, C.A. Selective sampling with redundant views. *Proceedings of the national conference on artificial intelligence*, (2000), 621–626.

114. Ng, A.Y. and Jordan, M.I. On Discriminative vs. Generative classifiers: A comparison of logistic regression and naive Bayes. 2001.

115. Nigam, K. and Ghani, R. Analyzing the effectiveness and applicability of co-training. *Proceedings of the ninth international conference on Information and knowledge management - CIKM '00*, (2000), 86–93.

116. Nigam, K., Mccallum, A., and Mitchell, T. Learning to Classify Text from Labeled and Unlabeled Documents. AAAI Press (1998), 792–799.

117. Olsson, F. A literature survey of active machine learning in the context of natural language processing. (2009).

118. Oppenheim, a. V.V. and Schafer, R.W.W. Dsp history - From frequency to quefrency: a history of the cepstrum. *IEEE Signal Processing Magazine 21*, 5 (2004), 95–106.

119. O'Toole, J.E., Sinclair, M.I., and Leder, K. Collecting household water usage data: telephone questionnaire or diary? *BMC medical research methodology 9*, (2009), 72.

120. Patel, K., Fogarty, J., Landay, J., and Harrison, B. Examining Difficulties Software Developers Encounter in the Adoption of Statistical Machine Learning. *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI 2008)*, (2008), 1998–2001.

121. Patel, K., Fogarty, J., Landay, J., and Harrison, B. Investigating statistical machine learning as a tool for software development. *Proceeding of the twenty-sixth SIGCHI on human factor in computing*, (2008), 667.

122. Patel, S., Reynolds, M., and Abowd, G. Detecting Human Movement by Differential Air Pressure Sensing in HVAC System Ductwork: An Exploration in Infrastructure Mediated Sensing. In J. Indulska, D. Patterson, T. Rodden and M. Ott, eds., *Pervasive Computing*. Springer Berlin / Heidelberg, 2008, 1–18.

123. Patel, S., Truong, K., and Abowd, G. PowerLine Positioning: A Practical Sub-Room-Level Indoor Location System for Domestic Use. In P. Dourish and A. Friday, eds., *UbiComp 2006: Ubiquitous Computing*. Springer Berlin / Heidelberg, 2006, 441–458.

124. Patel, S.N., Robertson, T., Kientz, J.A., Reynolds, M.S., and Abowd, G.D. At the flick of a switch: detecting and classifying unique electrical events on the residential power line. *Proceedings of the 9th international conference on Ubiquitous computing*, Springer-Verlag (2007), 271–288.

125. Pearl, J. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., 1988.

126. Rabiner, L. A tutorial on hidden Markov models and selected applications in speech recognition. *Proceedings of the IEEE*, (1989).

127. Reynolds, O. An experimental investigation of the circumstances which determine whether the motion of water shall be direct or sinuous, and of the law of resistance in parallel channels. *Philosophical Transactions of the Royal Society 174*, Cambridge University Press (1883), 51.

128. Riloff, E. and Shepherd, J. A corpus-based bootstrapping algorithm for Semi-Automated semantic lexicon construction. *Nat. Lang. Eng. 5*, 2 (1999), 147–156.

129. Rowan, J. Digital family portrait field trial: Support for aging in place. *Proceedings of the SIGCHI conference on Human*, (2005), 521–530.

130. Satterfield, Z. and Bharwaj, V. Water Meters Tech Brief. *National Environmental Services Center*, (2004), 2–5.

131. Schafer, R. Echo removal by discrete generalized linear filtering. *Mathematics of Computation 29*, 132 (1969), 1094.

132. Scheffer, T., Decomain, C., and Wrobel, S. Active hidden markov models for information extraction. In *Advances in Intelligent Data Analysis*. Springer, 2001, 309–318.

133. Schein, A.I. and Ungar, L.H. Active learning for logistic regression: an evaluation. *Machine Learning 68*, 3 (2007), 235–265.

134. Selover, C. *Hot water demand reduction credit for hers*. Berkeley, CA, 2012.

135. Settles, B. and Craven, M. An analysis of active learning strategies for sequence labeling tasks. *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, (2008), 1070–1079.

136. Settles, B. *Curious machines: active learning with structured instances*. ProQuest, 2008.

137. Settles, B. Active learning literature survey. *University of Wisconsin, Madison*, (2010).

138. Sha, F. and Pereira, F. Shallow parsing with conditional random fields. *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - NAACL '03*, Association for Computational Linguistics (2003), 134–141.

139. Sindhwani, V. and Keerthi, S. Newton methods for fast solution of semi-supervised linear SVMs. *Large scale kernel machines*, (2007).

140. Sutton, C. and McCallum, A. *An Introduction to Conditional Random Fields for Relational Learning*. 2007.

141. Swanson, E.R. Residential Water Flow Rate Analysis Using Unobtrusive Sensing. 2011.

142. Tapia, E.M., Intille, S.S., Lopez, L., and Larson, K. The Design of a Portable Kit of Wireless Sensors for Naturalistic Data Collection. *Proceedings of the International Conference on Pervasive Computing*, (2006), 117–134.

143. Tapia, E.M. and Intille, S.S. Activity Recognition in the Home Using Simple and Ubiquitous Sensors. *Proceedings of the International Conference on Pervasive Computing*, (2004).

144. Terabe, M. and Hashimoto, K. Evaluation criteria of feature splits for co-training. *Proceedings of the International Multi-conference for engineers and computer scientists I*, (2008), 19–21.

145. Tomanek, K. and Hahn, U. Semi-supervised active learning for sequence labeling. *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, (2009), 1039–1047.

146. Tong, S. and Chang, E. Support vector machine active learning for image retrieval. *Proceedings of the ninth ACM international conference on Multimedia*, (2001), 107–118.

147. Tong, S. Active learning: theory and applications. 2001.

148. Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y. Support vector machine learning for interdependent and structured output spaces. *Proceedings of the twenty-first international conference on Machine learning*, ACM (2004), 104–.

149. Tur, G., Hakkani-Tür, D., and Schapire, R.E. Combining active and semi-supervised learning for spoken language understanding. *Speech Communication 45*, 2 (2005), 171–186.

150. Ulusoy, I. and Bishop, C.M. Generative versus discriminative methods for object recognition. *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, (2005), 258–265.

151. Vail, D., Veloso, M., and Lafferty, J. Conditional random fields for activity recognition. *AAMAS*, (2007).

152. Valiant, L.G. A theory of the learnable. *Communications of the ACM 27*, 11 (1984), 1134–1142.

153. Vickers, A. Handbook of Water Use and Conservation: Homes, Landscapes, Industries, Businesses, Farms. *WaterPlow* , (2001).

154. Vishwanathan, S., Schraudolph, N., Schmidt, M.W., and Murphy, K.P. Accelerated training of conditional random fields with stochastic gradient methods. *Proceedings of the 23rd International Conference on Machine Learning*, (2006).

155. Wainwright, M. and Jordan, M.I. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, (2008).

156. Wallach, H. Efficient training of conditional random fields. (2002).

157. Weihl, J.S. and Kempton, W. Residential Hot Water Energy Analysis: Instruments and Algorithms. *Energy and Buildings 8*, 3 (1985), 197–204.

158. Wilkes, C., Mason, A., Niang, L., Jensen, K., and Hern, S. *Evaluation of the Meter-Master Data Logger and the Trace Wizard Analysis Software. Special Appendix to Quantification of Exposure-Related Water Uses for Various U.S. Subpopulations.* 2005.

159. Willis, R.M., Stewart, R. a, Panuwatwanich, K., Williams, P.R., and Hollingsworth, A.L. Quantifying the influence of environmental and water conservation attitudes on household end use water consumption. *Journal of environmental management 92*, 8 (2011), 1996–2009.

160. Willis, R.M., Stewart, R. a., Panuwatwanich, K., Jones, S., and Kyriakides, A. Alarming visual display monitors affecting shower end use water and energy conservation in Australian residential households. *Resources, Conservation and Recycling 54*, 12 (2010), 1117–1127.

161. Wilson, D. and Atkeson, C. Simultaneous Tracking & Activity Recognition (STAR) Using Many Anonymous , Binary Sensors. *Proceedings of the International Conference on Pervasive Computing*, (2005), 62–79.

162. Witten, I.H. and Frank, E. *Data Mining: Practical Machine Learning Tools and and Techniques*. 2005.

163. Wutich, a. Estimating Household Water Use: A Comparison of Diary, Prompted Recall, and Free Recall Methods. *Field Methods 21*, 1 (2008), 49–68.

164. Yarowsky, D. Unsupervised word sense disambiguation rivaling supervised methods. *Proceedings of the 33rd annual meeting on association for computational linguistics*, (1995), 189–196.

165. Yu, K., Bi, J., and Tresp, V. Active learning via transductive experimental design. *Proceedings of the 23rd international conference on Machine learning*, (2006), 1081–1088.

166. Zhou, Y. and Goldman, S. Democratic co-learning. *ICTA*, Ictai (2004).

167. Zhou, Z.-H., Chen, K.-J., and Jiang, Y. Exploiting unlabeled data in content-based image retrieval. In *Machine Learning: ECML 2004*. Springer, 2004, 525–536.

168. Zhu, X. and Ghahramani, Z. *Learning from Labeled and Unlabeled Data with Label Propagation*. 2002.

169. Zhu, X. and Ghahramani, Z. Semi-supervised learning using gaussian fields and harmonic functions. *Machine Learning*, (2003).

170. joeso. FlashRouters. 2010. https://www.flashrouters.com/blog/2012/05/21/more-ram-more-memory-on-a-router/.