UNIVERSITY OF CALIFORNIA
RIVERSIDE

Mining Time Series Data: Flying Insect Classification and Detection

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Yanping Chen

June 2015

Dissertation Committee:

Dr. Eamonn Keogh, Chairperson
Dr. Walid Najjar
Dr. Tao Jiang
Dr. Christian Shelton

The Dissertation of Yanping Chen is approved:

_____

_____

_____

_____

Committee Chairperson

University of California, Riverside

ACKNOWLEDGEMENTS

I would like to take this opportunity to express my sincerest gratitude to my advisor Dr. Eamonn Keogh for the invaluable guidance, supervision, and generous support during my doctoral study. I deeply appreciate that in spring 2012, he took me as one of his Ph.D students when I was in great desperate with my research. He guided me with great patience to do good research, and with his help, I regained my confidence. During my three and a half year's Ph.D study, Dr. Keogh taught me how to solve a problem, encouraged me to find interesting problems, and helped me with all valuable suggestions whenever I got stuck. His brilliant research philosophy and precious insightful advice have given me strength throughout my PhD stage and will continue to benefit my entire research life. It is of my *great* fortune to have Dr. Keogh as my advisor.

I would also like to thank Dr. Walid Najjar, Dr. Tao Jiang and Dr. Christian Shelton who are my committee members, for their generous support and valuable comments; Dr. Stefano Lonardi, Dr. Vegalis Christidis and Dr. Ertem Tuncel for being committee member in my oral-qualification exam.

My gratitude also goes to my colleagues in the data mining lab at UCR (names in random order), who offered me valuable help and friendship: Bing Hu, Nurjahan Begum, Mohammad Shokoohi-Yekta, Liudmila Ulanova, Jesin Zakaria, Thanawin (Art) Rakthanmanon, Gustavo Batista, Bilson Campana, Abdullah Mueen, Reza Rawassizadeh, Yan Zhu and Yuan Hao. Also I would like to thank Agenor Mafra-Neto and Adena Why for their valuable advice in entomology domain and their contributions to the flying insect classification project; Francois Petitjean and Germain Forestier for co-authoring

the Dynamic Time Warping Averaging paper. I would also like to thank Amy Ricks who is always helpful whenever I approached her.

Moreover, I would like to thank Bill & Melinda Gates Foundation for funding my research on flying insect classification; as well as all the financial support I received here: UCR Fellowship, Solid Fellowship, ICDM travel award and SDM travel award. Without these financial support, I cannot finish my doctoral program.

Finally, I would like to express my heartful gratitude to my family, especially to my parents for their constant love, patience and support. Without them, I would not reach where I am today; and thanks to my husband, Sheng Zhao, for his constant support and inspirations; as well as to all my family members for always being there to support me. I would also like to thank all my friends at UCR for being part of this wonderful journey.

ABSTRACT OF THE DISSERTATION

Mining Time Series Data: Flying Insect Classification and Detection

by

Yanping Chen

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, June 2015
Dr. Eamonn Keogh, Chairperson

The ability to use inexpensive, noninvasive sensors to accurately classify flying insects would have significant implications for entomological research, and allow for the development of many useful applications in vector control for both medical and agricultural entomology. Given this, the last sixty years have seen many research efforts on this task. To date, however, none of this research has had a lasting impact. We feel that the lack of progress in this pursuit can be attributed to two related factors: the lack of effective sensors makes data collection difficult, resulting in poor-quality and limited data; complicated classifiers built based on the limited data tend to be over-fitting and not well generalized to unknown insects classification. In this dissertation, we strive to solve these problems. The contribution of this dissertation is as follows:

First, we use optical sensors to record the "sound" of insect flight from up-to meters away, with invariance to interference from ambient sounds. With these sensors, we have recorded on the order of millions of labeled training instances, far more data than all previous efforts combined. The enormous amounts of data we have collected allow us to

avoid the over-fitting that has plagued previous research efforts. Based on the data, we proposed a simple, accurate and robust classification framework. We also explored several novel features to better distinguish different species of insects to improve the classification accuracy. Experimental results on large scale insect dataset shows the proposed classification framework is both accurate and robust.

In addition, we consider the scenario where the number of labeled training data is limited. Obviously, a good solution to the problem is semi-supervised learning, however, although there are a plethora of semi-supervised learning algorithms in the literature, we found that off-the-shelf semi-supervised learning algorithms do not typically work well for time series. In the second part of this dissertation, we explain *why* semi-supervised learning algorithms typically fail for time series problems, and we introduce a simple but very effective fix.

Finally, we consider the scenario of insect monitoring where the task is to discover new/invasive species of insects from the data stream produced by sensors while doing the monitoring. We generalize this scenario and propose a never-ending learning framework that continuously attempts to extract new concepts from time series data stream. We demonstrated the utility of the framework by applying it to detect the invasive species of insects, and the results show that it is very both accurate and robust.

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

The history of humankind is intimately connected to insects. Beneficial insects pollinate the majority of crop species that we eat. At the same time, insect borne diseases kill millions of people and destroy tens of billions of dollars' worth of crops annually. To reduce the negative effects of insects, some insect interventions have been proposed and used for a long time, such as cleaning the environments, spraying pesticides, and using the bed-nets. However, all interventions cost money, even if it is just the human labor. Nevertheless, if we know the best location and time to spend the resources, the interventions can be made much more efficient. Knowing the best locations and time requires knowledge of the insect population, which requires recognizing the species of the insects.

In this dissertation, we propose a system which automatically and accurately counts and classifies flying insects based on their wing-beat sounds. We first show how our classifier works to identify flying insects in Chapter 2. We then consider the scenario where the number of labeled data is limited and propose a time series semi-supervised learning algorithm in Chapter 3. In Chapter 4, we propose a never-ending learning framework to continuously extract new species of insects from the data stream when we initially have only vague understanding of what species we will encounter. Finally, we give conclusions in Chapter 5.

In the following text, we show the detail of the motivations of each project.

1.1 Flying Insect Classification

Understanding target insect population density is important in planning effective and efficient insect interventions. Many possible predictors of insect density have been suggested. For example, the damage of the crops is used to estimate the pests' population and the early detection of affected cases or hospital admissions is used to predict the insect vectors density[3]. However, most of the predictors are simply surrogates for the prevalence of the target insects, and surrogate variables are always doomed to be inaccurate and/or tardy. Therefore, it would be useful if we could measure insect density *directly*.

Where insect density is current measured, it is typically measured with sticky traps. The sticky traps are relatively inexpensive but they require human experts. A trained operator has to set up the traps, leave them in the field for some time, then come back to count and classify the content. The greatest problem with current insect traps is not the need of human labor, but is the time lag between the time of the insect's arrival and the time they are counted. Under realistic conditions in the developing world this lag may be a week or more, however, the adult stage of some insects (such as *Anopheles*) is only about two weeks, thus by the time the insects are detected, the damage may already be done.

What is really needed is the digital classification to allow real-time monitoring of the insect population. The ability to automatically and accurately classify insects allows for

the development of many useful applications in vector control for both medical and agricultural entomology. Some application examples are shown below:

- It enables efficient insect interventions by providing precise information that allows target spraying of pesticides with pinpoint accuracy. Nowadays, the common way for pest control is to use a blanket spray of pesticides. Blanket spraying of pesticides is very expensive. It costs money, causes pollution to the environment and has the risk of the insects developing resistance [5]. On the contrary, with the sensors that provide precise information of what insects they have in the field and where they are, the farmers will know what pesticides to use and where to spray them, and thus, to save money and reduce the environmental cost.
- Electrical Discharge Insect Control Systems EDICS ("bug zappers") are insect traps that attract and then electrocute insects. They are very popular with consumers who are presumably gratified by the characteristic "buzz" produced when an insect is electrocuted. While most commercial devices are sold as mosquito deterrents, studies have shown that as little as 0.22% of the insects killed are mosquitoes [24]. This is not surprising, since the attractant is typically just an ultraviolet light. Augmenting the traps with CO2 or other chemical attractants helps, but still allows the needless electrocution of beneficial insects. ISCA technologies (owned by author A. M-N) is experimenting with building a "smart trap" that classifies insects as they approach the trap, selectively killing the target insects but blowing the non-target insects away with compressed air.

- The Sterile Insect Technique has been used to reduce the populations of certain target insects, most notably with Screwworm flies (*Cochliomyia hominovorax*) and the Mediterranean fruit fly (*Ceratitis capitata*). The basic idea is to release sterile males into the wild to mate with wild females. Because the males are sterile, the females will lay eggs that are either unfertilized, or produce a smaller proportion of fertilized eggs, leading to population declines and eventual eradication in certain areas [12]. Note that it is important not to release females, and sexing mosquitoes is notoriously difficult. Researchers at the University of Kentucky are experimenting with our sensors to create insectaries from which only male hatchlings can escape. The idea is to use a modified EDICS or a high powered laser that selectively turns on and off to allow males to pass through, but kills the females.
- Much of the research on insect behavior with regard to color, odor, etc., is done by having human observers count insects as they move in dual choice olfactometer or on landing strips etc. For example, [19] notes, "Virgin female wasps were individually released downwind and the color on which they landed was recorded (by a human observer)." There are several problems with this: human time becomes a bottleneck in research; human error is a possibility; and for some host seeking insects, the presence of a human nearby may affect the outcome of the experiment (unless costly isolation techniques/equipment is used). We envision our sensor can be used to accelerate such research by making it significantly cheaper to conduct these types of experiments. Moreover, the unique abilities of our system will allow researchers to conduct experiments that are currently impossible. For example, a recent paper [51]

attempted to see if there are sex-specific differences in the daily flight activity patterns of Anopheles gambiae mosquitoes. To do this, the authors placed individual sexed mosquitoes in small glass tubes to record their behavior. However, it is possible that both the small size of the glass tubes and the fact that the insects were in isolation affected the result. Moreover, even the act of physically sexing the mosquitoes may affect them due to metabolic stress etc. In contrast, by using our sensors, we can allow unsexed pupae to hatch out and the adults fly in cages with order of magnitude larger volumes. In this way, we can automatically and noninvasively sex them to produce sex-specific daily flight activity plots.

Given the importance of insect classification, we introduce a general classification framework for flying insect classification in Chapter 2. We show that with pseudo-acoustic optical sensors, we can easily collect large amounts insect flight data; that a Bayesian classification approach allows to efficiently learn classification models that are very robust to over-fitting and easily incorporate arbitrary number of features. Our classifier exploits features that are both intrinsic and extrinsic to the insect's flight behavior to improve classification accuracy. Experimental results on large scale datasets shows that the proposed classification is both accurate and robust.

1.2 Time Series Semi-Supervised Learning

Classification of time series data is an important problem with applications in virtually every scientific endeavor, and much progress has been made over the past two decades. However, almost all research efforts assume that there are large amounts of

labeled training data [1][7]. In reality, the high cost of labeling the data may render such an assumption invalid. For example, a single polysomnography (sleep study) test may produce up to 40,000 such heartbeats, but it requires the time and expertise of a cardiologist to annotate individual heartbeats in the trace[1], a process that is both expensive and time-consuming.

Given that the acquisition of unlabeled data is relatively trivial but obtaining labeled data is difficult, the obvious solution to this problem may appear to be the application of semi-supervised learning. However, although the literature has offered a plethora of semi-supervised learning methods, as we shall show later, direct applications of off-the-shelf semi-supervised learning algorithms do not typically work well for time series. The research community working on time series classification has typically only used the copious UCR Archive [2] to test their algorithms, and has thus largely avoided this pressing problem.

In Chapter 3, we first explain why semi-supervised learning algorithms typically fail for time series problems, and we then introduce a simple but very effective fix. Extensive experiments on diverse real world problems show that the simple fix dramatically improves the accuracy of semi-supervised learning in time series.

1.3 Never-Ending Learning from Data Stream

Virtually all work on time series classification assumes a one-time training session in which multiple labeled examples of all the concepts to be learned are provided. This assumption is sometimes valid, for example, when learning a set of gestures to control a

game or novel HCI interface[8]. However, in many medical and scientific applications, we initially may have only the vaguest understanding of what concepts need to be learned. For example, suppose we would like to monitor the population density of flying insects in a field using a sensor. While the sensor will produce data forever, at each moment, we can only cache a fixed amount of data. Before we see the data stream, we may have the vague idea of what species of insects we will encounter, such as those that are local and frequently encountered in the area. However, we may not have a complete list of all the species of insects as we are almost guaranteed to encounter new/invasive species. Therefore, different from traditional classification problems where we initially have all the classes defined and all unknown objects belongs to one of the pre-defined classes, in this case, we need to discover new classes from the data stream as we are doing the monitoring.

For this purpose, in Chapter 4, we propose a never-ending learning framework in which an agent *observes data streams forever*, and continuously attempts to *extract new concepts* from the stream while doing the classification.

As a note, this project is a jointed work with other researchers (joint first authors). The detailed description of this work can be found in [4]. We iterate some parts of it in this dissertation, which are closely related to the theme of this dissertation.

Chapter 2: Flying Insect Detection and Classification with Inexpensive Sensors

In this chapter, we will demonstrate a system for flying insect classification. We break the chapter into six sections. Section 2.1 illustrates the motivation of the work. In Section 2.2, we review the background and the related work of this topic. We then illustrate how we do the data collection and data pre-processing in Section 2.3, and introduce the classification framework in Section 2.4. Experimental evaluation of the proposed system is shown in Section 2.5. Finally, we offer the conclusion of this part in Section 2.6.

2.1 Introduction

The idea of automatically classifying insects using the incidental sound of their flight (as opposed to deliberate insect sounds produced by stridulation [28]) dates back to the very dawn of computers and commercially available audio recording equipment. In 19451, three researchers at the Cornell University Medical College, Kahn, Celestin and Offenhauser, used equipment donated by Oliver E. Buckley (then President of the Bell Telephone Laboratories) to record and analyze mosquito sounds [29].

The authors later wrote, *"It is the authors' considered opinion that the intensive application of such apparatus will make possible the precise, rapid, and simple*

[1] An even earlier paper [46]makes a similar suggestion. However, these authors determined the wingbeat frequencies *manually*, aided by a stroboscope.

observation of natural phenomena related to the sounds of disease-carrying mosquitoes and should lead to the more effective control of such mosquitoes and of the diseases that they transmit." [30]. In retrospect, given the importance of insects in human affairs, it seems astonishing that more progress on this problem has not been made in the intervening decades.

There have been sporadic efforts at flying insect classification from audio features [52][53][59][40], especially in the last decade [39][48]; however, little real progress seems to have been made. By "lack of progress" we do not mean to suggest that these pioneering research efforts have not been fruitful. However, we would like to have automatic classification to become as simple, inexpensive, and ubiquitous as current mechanical traps such as sticky traps or interception traps [15], but with all the advantages offered by a digital device: higher accuracy, very low cost, real-time monitoring ability, and the ability to collect additional information (time of capture[2], etc.).

We feel that the lack of progress in this pursuit can be attributed to three related factors:

1. Most efforts to collect data have used acoustic microphones [46][11][35][45]. Sound attenuates according to an inverse squared law. For example, if an insect flies just three times further away from the microphone, the sound intensity (informally, the loudness) drops to one ninth. Any attempt to mitigate this by using a more

[2] A commercially available rotator bottle trap made by BioQuip® (2850) does allow researchers to measure the time of arrival at a granularity of *hours*. However, as we shall show in Section *Additional Feature: Circadian Rhythm of Flight Activity*, we can measure the time of arrival at a *sub-second* granularity and exploit this to improve classification accuracy.

sensitive microphone invariably results in extreme sensitivity to wind noise and to ambient noise in the environment. Moreover, the difficulty of collecting data with such devices seems to have led some researchers to obtain data in unnatural conditions. For example, nocturnal insects have been forced to fly by tapping and prodding them under bright halogen lights; insects have been recorded in confined spaces or under extreme temperatures [11][39]. In some cases, insects were tethered with string to confine them within the range of the microphone [46]. It is hard to imagine that such insect handling could result in data which would generalize to insects in natural conditions.

2. Unsurprisingly, the difficultly of obtaining data noted above has meant that many researchers have attempted to build classification models with very limited data, as few as 300 instances [37] or less. However, it is known that for building classification models, more data is better [27][1][54].
3. Compounding the poor quality data issue and the sparse data issue above is the fact that many researchers have attempted to learn very complicated classification models3, especially neural networks [40][39][33]. However, neural networks have many parameters/settings, including the interconnection pattern between different layers of neurons, the learning process for updating the weights of the interconnections, the activation function that converts a neuron's weighted input to its output activation, etc. Learning these on say a spam/email classification problem

[3] While there is a formal framework to define the complexity of a classification model (i.e. the VC dimension (Vapnik and Chervonenkis 1971)), informally we can think of a *complicated* or *complex* model as one that requires many parameters to be set or learned.

with millions of training data is not very difficult [62], but attempting to learn them on an insect classification problem with a mere twenty examples is a recipe for overfitting (*cf*. Figure 3). It is difficult to overstate how optimistic the results of neural network experiments can be unless rigorous protocols are followed [44].

In this work, we will demonstrate that we have largely solved all these problems. We show that we can use optical sensors to record the "sound" of insect flight from meters away, with complete invariance to wind noise and ambient sounds. We demonstrate that these sensors have allowed us to record on the order of millions of labeled training instances, far more data than all previous efforts combined, and thus allow us to avoid the overfitting that has plagued previous research efforts. We introduce a principled method to incorporate additional information into the classification model. This additional information can be as quotidian and as easy-to-obtain as the time-of-day, yet still produce significant gains in accuracy. Finally, we demonstrate that the enormous amounts of data we collected allow us to take advantage of "The unreasonable effectiveness of data" [27] to produce simple, accurate and robust classifiers.

In summary, we believe that flying insect classification has moved beyond the dubious claims created in the research lab and is now ready for real-world deployment. The sensors and software we present in this work will provide researchers worldwide robust tools to accelerate their research.

2.2 Background and Related Work

The vast majority of attempts to classify insects by their flight sounds have explicitly or implicitly used *just* the wingbeat frequency [46][55][52][53][59][40][37]. However, such an approach is limited to applications in which the insects to be discriminated have very different frequencies. Consider Figure 1.I which shows a histogram created from measuring the wingbeat frequencies of three (sexed) species of insects, *Culex stigmatosoma* (female), *Aedes aegypti* (female), and *Culex tarsalis* (male) (We defer details of how the data was collected until later in the paper).

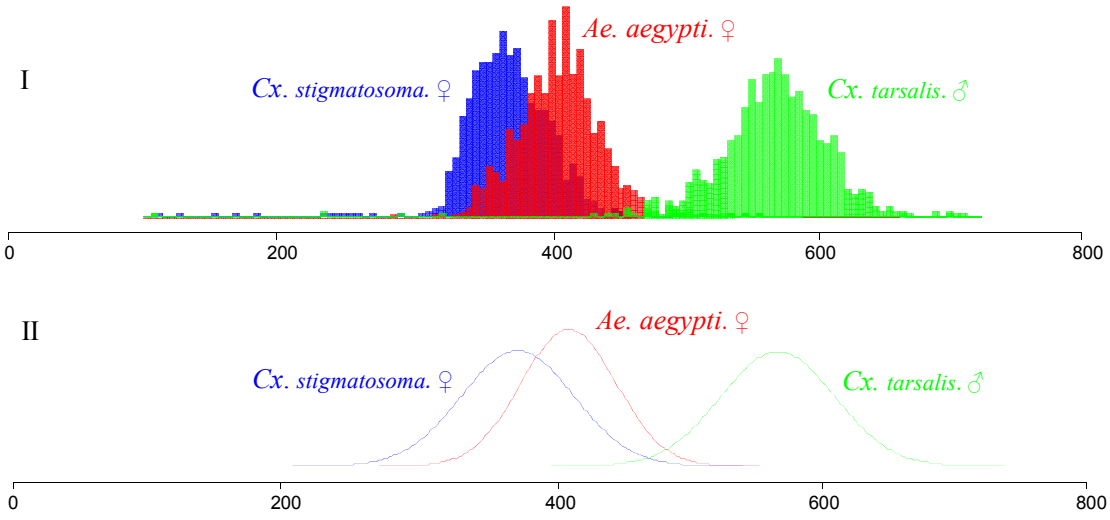


Figure 1: I) Histograms of wingbeat frequencies of three species of insects, *Cx. stigmatosoma* ♀, *Ae. aegypti.* ♀, and *Cx. tarsalis.* ♂. Each histogram is derived based on 1,000 wingbeat sound snippets. II) Gaussian curves that fit the wingbeat frequency histograms

It is visually obvious that if asked to separate *Cx. stigmatosoma* ♀ from *Cx. tarsalis* ♂, the wingbeat frequency could produce an accurate classification, as the two species have very different frequencies with minimal overlap. To see this, we can compute the

optimal Bayes error rate [25], which is a strict lower bound to the actual error rate obtained by *any* classifier that considers *only* this feature. Here, the Bayes error rate is half the *overlapping* area under both curves divided by the *total* area under the two curves.

Because there is only a tiny overlap between the wingbeat frequency distributions of the two species, the Bayes error rate is correspondingly small, 0.57% if we use the raw histograms and 1.08% if we use the derived Gaussians.

However, if the task is to separate *Cx. stigmatosoma.* ♀ from *Ae. aegypti.* ♀, the wingbeat frequency will not do as well, as the frequencies of these two species overlap greatly. In this case, the Bayes error rate is *much* larger, 24.90% if we use the raw histograms and 30.95% if we use the derived Gaussians.

This problem can only get worse if we consider more species, as there will be increasing overlap among the wingbeat frequencies. This phenomenon can be understood as a real-value version of the Pigeonhole principle [26]. Given this, it is unsurprising that some doubt the utility of wingbeat sounds to classify the insects. However, we will show that the analysis above is pessimistic. Insect flight sounds *can* allow much higher classification rates than the above suggests because:

- There is more information in the flight sound signal than just the wingbeat frequency. By analogy, humans have no problem distinguishing between Middle C on a piano and Middle C on a saxophone, even though both are the same 261.62 Hz fundamental frequency. The Bayes error rate to classify the three species in Figure 1.I using *just* the wingbeat frequency is 19.13%; however, as we shall see below, that by

using the additional features from the wingbeat signal, we can obtain an error rate of 12.43%.

- We can augment the wingbeat sounds with additional cheap-to-obtain features that can help to improve the classification performance. For example, many species may have different flight activity circadian rhythms. As we shall see below in Section 2.4.3, simply incorporating the *time-of-intercept* information can significantly improve the performance of the classification.

The ability to allow the incorporation of auxiliary features is one of the reasons we argue that the Bayesian classifier is ideal for this task (cf. Section 2.4.1), as it can gracefully incorporate evidence from multiple sources and in multiple formats.

2.3 Insect Data Collection

2.3.1. Insect Colony and Rearing

Six species of insects were studied in this work: *Cx. tarsalis*, *Cx. stigmatosoma*, *Ae. aegypti*, *Culex quinquefasciatus*, *Musca domestica* and *Drosophila simulans*.

All adult insects were reared from laboratory colonies derived from wild individuals collected at various locations. *Cx. tarsalis* colony was derived from wild individuals collected at the Eastern Municipal Water District's demonstration constructed treatment wetland (San Jacinto, CA) in 2001. *Cx. quinquefasciatus* colony was derived from wild individuals collected in southern California in 1990 (Georghiou and Wirth 1997). *Cx. stigmatosoma* colony was derived from wild individuals collected at the University of

California, Riverside, Aquatic Research Facility in Riverside, CA in 2012. *Ae*. *aegypti* colony was started in 2000 with eggs from Thailand (Van Dam and Walton 2008). *Musca domestica* colony was derived from wild individuals collected in San Jacinto, CA in 2009, and *Drosophila simulans* colony were derived from wild individuals caught in Riverside, CA in 2011.

The larvae of *Cx*. *tarsalis*, *Cx*. *quinquefasciatus*, *Cx*. *stigmatosoma* and *Ae*. *aegypti* were reared in enamel pans under standard laboratory conditions (27°C, 16:8 h light:dark [LD] cycle with 1 hour dusk/dawn periods) and fed ad libitum on a mixture of ground rodent chow and Brewer's yeast (3:1, v:v). *Musca domestica* larvae were kept under standard laboratory conditions (12:12 h light:dark [LD] cycle, 26°C, 40% RH) and reared in a mixture of water, bran meal, alfalfa, yeast, and powdered milk. *Drosophila simulans* larvae were fed ad libitum on a mixture of rotting fruit.

Mosquito pupae were collected into 300-mL cups (Solo Cup Co., Chicago IL) and placed into experimental chambers. Alternatively, adults were aspirated into experimental chambers within 1 week of emergence. The adult mosquitoes were allowed to feed ad libitum on a 10% sucrose and water mixture; food was replaced weekly. Cotton towels were moistened, twice a week, and placed on top of the experimental chambers and a 300-ml cup of tap water (Solo Cup Co., Chicago IL) was kept in the chamber at all times to maintain a higher level of humidity within the cage. *Musca domestica* adults were fed ad libitum on a mixture of sugar and low-fat dried milk, with free access to water. *Drosophila simulans* adults were fed ad libitum on a mixture of rotting fruit.

Experimental chambers consisted of Kritter Keepers (Lee's Aquarium and Pet Products, San Marcos, CA) that were modified to include the sensor apparatus as well as a sleeve (Bug Dorm sleeve, Bioquip, Rancho Dominguez, CA) attached to a piece of PVC piping to allow access to the insects. Two different sizes of experimental chambers were used, the larger 67 cm L x 22 cm W x 24.75 cm H, and the smaller 30 cm L x 20 cm W x 20 cm H. The lids of the experimental chambers were modified with a piece of mesh cloth affixed to the inside in order to prevent escape of the insects, as shown in Figure 2.I. Experimental chambers were maintained on a 16:8 h light:dark [LD] cycle, 20.5-22°C and 30-50% RH for the duration of the experiment. Each experimental chamber contained 20 to 40 individuals of a same species, in order to capture as many flying sounds as possible while limiting the possibility of capturing more than one insect-generated sound at a same time.

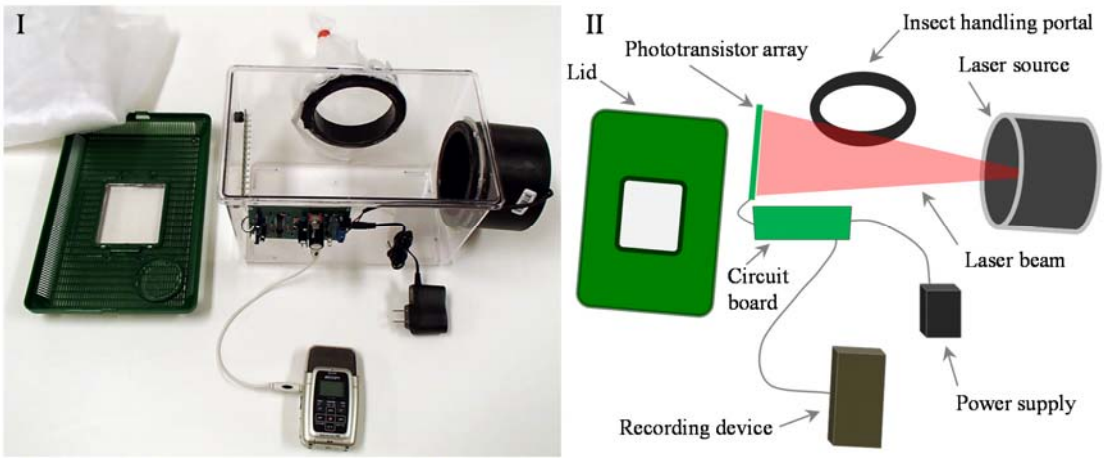


Figure 2: I) One of the cages used to gather data for this project. II) A logical version of the setup with the components annotated

Some tests were conducted with newly emerged adults, which would be virgins, but other trials were not. Anecdotally this appears to make no difference to the task-at-hand,

however a formal study is currently underway by an independent group of researchers using our sensors and software.

2.3.2. Instruments to Record Flying Sounds

We used the sensor described in [10] to capture the insect flying sounds. The logic design of the sensor consists of a phototransistor array which is connected to an electronic board, and a laser line pointing at the phototransistor array. When an insect flies across the laser beam, its wings partially occlude the light, causing small light fluctuations. The light fluctuations are captured by the phototransistor array as changes in current, and the signal is filtered and amplified by the custom designed electronic board. The physical version of the sensor is shown in Figure 2.I.

The output of the electronic board feeds into a digital sound recorder (Zoom H2 Handy Recorder) and is recorded as audio data in the MP3 format. Each MP3 file is 6 hours long, and a new file starts recording immediately after a file has recorded for 6 hours, so the data is continuous. The length of the MP3 file is limited by the device firmware rather than the disk space. The MP3 standard is a lossy format and optimized for human perception of speech and music. However, most flying insects produce sounds that are well within the range of human hearing and careful comparisons to lossless recordings suggest that we lose no exploitable (or indeed, *detectable*) information.

2.3.3. Sensor Data Processing

We downloaded the MP3 sound files to a PC twice a week and used a detection algorithm to automatically extract the brief insect flight sounds from the raw recording

data. The detection algorithm used a sliding window to "slide" through the raw data. At each data point, a classifier/detector is used to decide whether the audio segment contains an insect flying sound. It is important to note that the classifier used at this stage is solving the relatively simple two-class task, differentiating between *insect*|*non-insect*. We will discuss the more sophisticated classifier, which attempts to differentiate species and sex, in the next section.

The classifier/detector used for the *insect*|*non-insect* problem is a nearest neighbor classifier based on the frequency spectrum. For ground truth data, we used ten flying sounds extracted from early experiments as the training data for the *insect* sounds, and ten segments of raw recording background noise as the training data for the *non-insect* sounds. The number of training data was limited to ten, because more training data would slow down the algorithm while fewer data would not represent variability observed. Note that the training data for background sounds can be different from minute to minute. This is because while the frequency spectrum of the background sound has little variance within a short time interval, it can change greatly and unpredictably in the long run. This variability (called *concept drift* in the machine learning community [58][61]) may be due to the effects of temperature change on the electronics and the slow decline of battery output power etc. Fortunately, given the high signal-to-noise ratio in the audio, the high variation of the *non-insect* sounds does not cause a significant problem. Figure 4.I shows an example of a one-second audio clip containing a flying insect generated by our sensor. As we can see, the signal of insects flying across the laser is well distinguished from the

background signal, as the amplitude is much higher and the range of frequency is quite different from that of background sound.

The length of the sliding window in the detection algorithm was set to be 100 ms, which is about the average length of a flying sound. Each detected insect sound is saved into a one-second long WAV format audio file by centering the insect flying signal and padding with zeros elsewhere. This makes all flying sounds the same length and simplifies the future archiving and processing of the data. Note that we converted the audio format from MP3 to WAV at this stage. This is simply because we publicly release all our data so that the community can confirm and extend our results. Because the vast majority of the signal processing community uses Matlab, and Matlab provides native functions for working with WAV files, this is the obvious choice for an archiving format. Figure 4.II shows the saved audio of the insect sound shown in Figure 4.I.

Flying sounds detected in the raw recordings may be contaminated by the background noise, such as the 60 Hz noise from the American domestic electricity, which "bleeds" into the recording due to the inadequate filtering in power transformers. To obtain a cleaner signal, we applied the spectral subtraction technique [13][23] to each detected flying sound to reduce noise.

2.4 Flying Insect Classification

In the section above, we showed how a simple nearest neighbor classifier can *detect* the sound of insects, and pass the sound snippet on for further inspection. Here, we

discuss algorithms to actually classify the snippets down to species (and in some cases, sex) level.

2.4.1. Using Bayesian Network for Classification

While there are a host of classification algorithms in the literature (decision trees, neural networks, nearest neighbor, etc.), the Bayes classifier is optimal in minimizing the probability of misclassification [21], under the assumption of independence of features. The Bayes classifier is a simple probabilistic classifier that predicts class membership probabilities based on Bayes' theorem. In addition to its excellent classification performance, the Bayesian classifier has several properties that make it extremely useful in practice and particularly suitable to the task at hand.

1. The Bayes classifier is undemanding in both CPU and memory requirements. Any devices to be deployed in the field in large quantities will typically be small devices with limited resources, such as limited memory, CPU power and battery life. The Bayesian classifier (once constructed offline in the lab) requires time and space resources that are just linear in the number of features.
2. The Bayes classifier is very easy to implement. Unlike neural networks [39][33], the Bayes classifier does not have many parameters that must be carefully tuned. In addition, the model is fast to build, and it requires only a small amount of training data to estimate the distribution parameters necessary for accurate classification, such as the means and variances of Gaussian distributions.

3. Unlike other classification methods that are essentially "black box", the Bayesian classifier allows for the graceful introduction of user knowledge. For example, if we have external (to the training data set) knowledge that given the particular location of a deployed insect sensor we should expect to be twice as likely to encounter a *Cx. tarsalis* as an *Ae. aegypti*, we can "tell" the algorithm this, and the algorithm can use this information to improve its accuracy. This means that in some cases, we can augment our classifier with information gleaned from the text of journal papers or simply the experiences of field technicians. In Section 2.4.4, we give a concrete example of this. Another example of how the Bayesian classifier allows us to gracefully add domain knowledge is a consideration of the effect of temperature/humidity on flight. While the experiments reported here reflect a single temperate for simplicity, in ongoing work by the current authors, it appears it is possible to predict the changes in wingbeat frequency due to the temperatures effect on air density. This means we can make the Bayesian classifier invariant to changes in temperature, without having to explicitly collect data recorded at different temperatures.
4. The Bayesian classifier simplifies the task flagging anomalies. Most classifiers must make a classification decision, even if the object being classified is vastly different to anything observed in the training phase. In contrast, we can slightly modify the Bayesian classifier to produce an "*Unknown*" classification. One or two such classifications per day could be ignored, but a spate of them could be investigated in case it is indicative of an infestation of a completely unexpected invasive species.

5. When there are multiple features used for classification, we need to consider the possibility of missing values, which happens when some features are not observed. For example, as we discuss below, we use *time-of-intercept* as a feature. However, a dead clock battery could deny us this feature even when the rest of the system is working perfectly. Missing values are a problem for any learner and may cause serious difficulties. However, the Bayesian classifier can trivially handle this problem, simply by dynamically ignoring the feature in question at classification time.

Because of the considerations listed above, we argue that the Bayesian classifier is the best for our problem at hand. Note that our decision to use Bayesian classifier, while informed by the above advantages, was also informed by an extensive empirical comparison of the accuracy achievable by other methods, given that in some situations *accuracy* trumps all other considerations. While we omit exhaustive results for brevity, in Figure 3 we show a comparison with the neural network classifier, as it is the most frequently used technique in the literature [39]. We considered only the frequency spectrum of wingbeat snippets for the three species discussed in Figure 1. The training data was randomly sampled from a pool of 1,500 objects, and the test data was a completely disjoint set of 1,500 objects, and we tested over 1,000 random resamplings. For the neural network, we used a single hidden layer of size ten, which seemed to be approximately the default parameters in the literature.

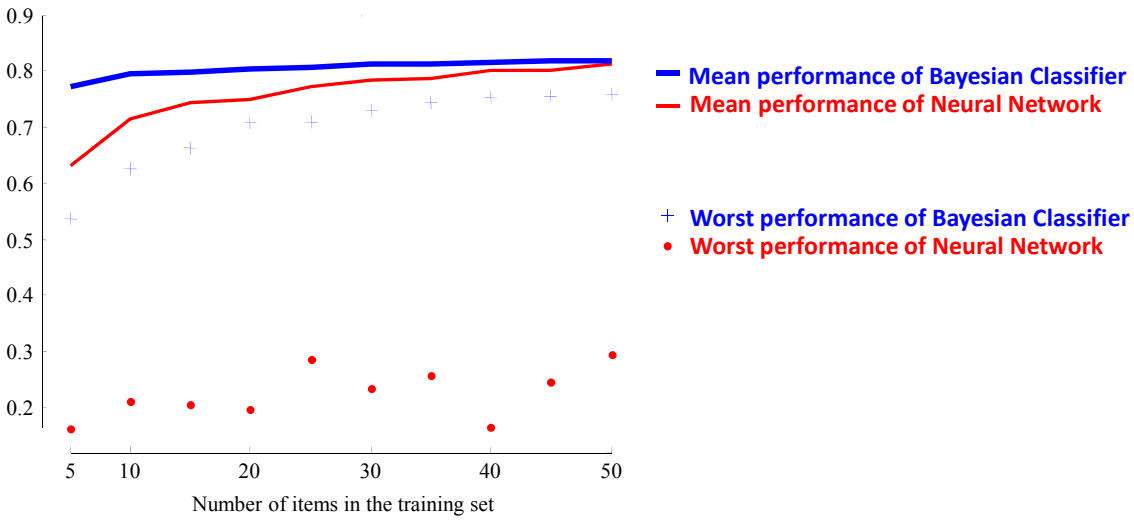


Figure 3: A comparison of the mean and worst performance of the Bayesian versus Neural Networks Classifiers for datasets ranging in size from five to fifty

The results show that while the neural network classifier eventually converges on the performance of the Bayesian classifier, it is significantly worse for smaller datasets. Moreover, for any dataset size in the range examined, it can occasionally produce pathologically poor results, doing worse than the default rate of 33.3%.

The intuition behind Bayesian classification is to find the mostly likely class given the data observed. When the classifier is based on a single feature F_1, the probability that an observed data f_1 belongs to a class C_i is calculated as:

$$P(C_i|\mathrm{F}_1 = f_1) \propto P(C_i)P(\mathrm{F}_1 = f_1|C_i) \quad (1)$$

Where $P(C_i)$ is the prior probability of class C_i, and $P(\mathrm{F}_1 = f_1|C_i)$ is the class-conditioned probability of observing feature f_1 in class C_i.

2.4.2. Primary Feature: Insect Flight Sound Spectrum

For insect classification, the primary data we observed are the flight sounds, as illustrated in Figure 4.I. The flying sound signal is the non-zero amplitude section (red/bold) in the center of the audio, and can be represented by a sequence S = $<s_1,s_2,\ldots s_N>$, where s_i is the signal sampled in the instance i and N is the total number of samples of the signal. This sequence contains a lot of acoustic information, and features can be extracted from it.

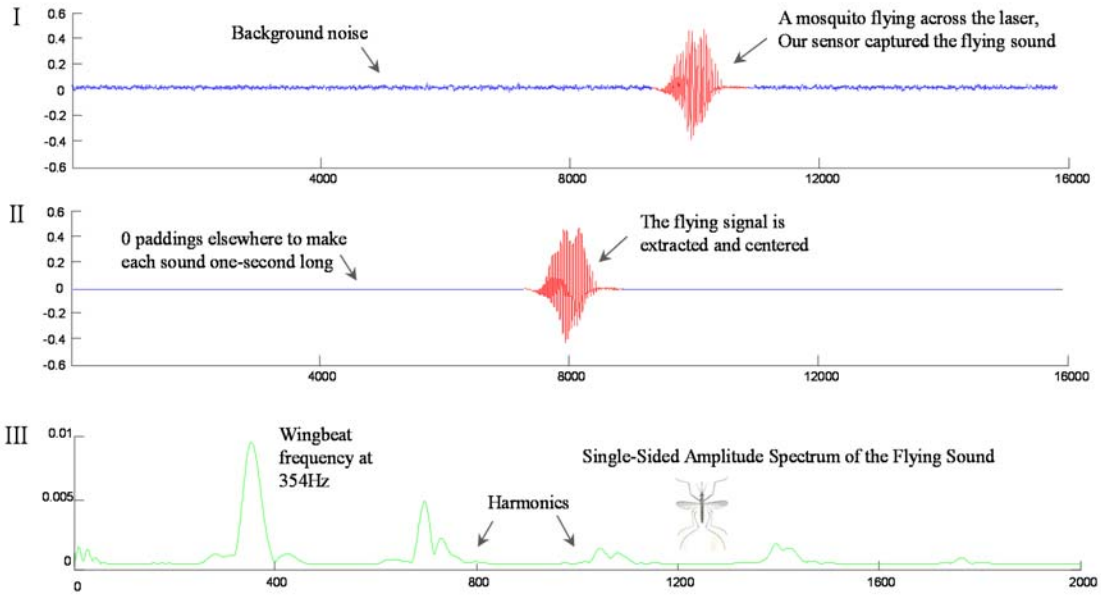


Figure 4: I) An example of a one-second audio clip containing a flying sound generated by the sensor. The sound was produced by a female *Cx. stigmatosoma*. The insect sound is highlighted in red/bold. II) The insect sound that is cleaned and saved into a one-second long audio clip by centering the insect signal and padding with 0s elsewhere. III) The frequency spectrum of the insect sound obtained using DFT

The most obvious feature to extract from the sound snippet is the wingbeat frequency. To compute the wingbeat frequency, we first transform the audio signal into a frequency spectrum using the Discrete Fourier Transform (DFT) [14]. As shown in Figure 4.III, the

frequency spectrum of the sound in Figure 4.II has a peak in the fundamental frequency at 354 Hz, and some harmonics in integer multiples of the fundamental frequency. The highest peak represents the frequency of interest, i.e., the insect wingbeat frequency.

The class-conditioned wingbeat frequency distribution is a univariate density function that can be easily estimated using a histogram. Figure 1.I shows a wingbeat frequency histogram plot for three species of insects (each for a single sex only). We can observe that the histogram for each species is well modeled by Gaussian distribution. Hence, we fit a Gaussian for each distribution and estimated the means and variances using the frequency data. The fitted Gaussian distributions are shown in Figure 1.II. Note that as hinted at in the introduction, the Bayesian classifier does not have to use the idealized Gaussian distribution; it could use the raw histograms to estimate the probabilities instead. However, using the Gaussian distributions is computationally cheaper at classification time and helps guard against overfitting.

With these class-conditioned distributions, we can calculate the class-conditioned probability of observing the flying sound from a class, given the wingbeat frequency. For example, suppose the class of the insect shown in Figure 4.I is unknown, but we have measured its wingbeat frequency to be 354 Hz. Further suppose that we have previously measured the mean and standard deviation of female *Cx. stigmatosoma* wingbeat frequency to be 365 and 41, respectively. We can then calculate the probability of observing this wingbeat frequency from a female *Cx. stigmatosoma* insect using the Gaussian distribution function:

$$P(wingbeat = 354 \text{ Hz}|\text{female } Cx.stigmatosoma\ ) = \frac{1}{41\sqrt{2\pi}} e^{-\frac{(354-365)^2}{2\times 41^2}}$$

We can calculate the probabilities for the other classes in a similar way, and predict the unknown insect as the most likely class. In this example, the *prior* probability is equal for each class, and the unknown insect is about 2.1 times more likely to be a female *Cx. stigmatosoma* than be a female *Ae. aegypti* (the second most likely class), and thus is (in this case, *correctly*) classified as a female *Cx. stigmatosoma*.

Note that the wingbeat frequency is a scalar, and learning the class-conditioned density functions for a low-dimensional feature (typically no more than 3 dimensions) is easy, because they can either be fitted using some distribution models, such as the Gaussian distribution, or be approximated using a histogram which can be constructed with a small amount of training data. However, if a feature is high-dimensional, we need other multivariate density function estimation methods, because we usually do not have any idea of what distribution model should fit the distributions of the high-dimensional features, and building a histogram of a high-dimensional feature requires a prohibitively large size of training dataset (the size of training dataset grows exponentially with the increase of dimensionality).

The literature has offered some multivariate density function estimation methods for high-dimensional variables, such as the Parzen–Rosenblatt window method [49][43] and the k-Nearest-Neighbors (kNN) approach [34]. The kNN approach is very simple; it leads to an approximation of the optimal Bayes classifier, and hence, we use it in this work to estimate the class-conditioned density functions for high-dimensional features.

The kNN approach does not require a training phase to learn the class-conditioned density function. It directly uses the training data to estimate the probability of observing an unknown data in a class. Specifically, given an observed data X, the kNN approach first searches the training data to find the top k nearest neighbors of X. It then computes the probability of observing X in class C_i as the fraction of the top k nearest neighbors which are labeled as class C_i, that is,

$$P(X|C_\mathrm{i}) = \frac{k_{C_i}}{k} \tag{2}$$

Where k is a user-provided parameter specifying the number of nearest neighbors, and k_{C_i} is the number of neighbors that are labeled as class C_i among the top k nearest neighbors.

With equation (2), we can calculate the class-conditioned probability, and "plug" this into the Bayesian classifier. As an example, imagine that we use the entire spectrum as a feature for the insect sound (cf. Figure 4.II). Given an unknown insect, we first transform its flight sound into the spectrum representation, and then search the entire training data to find the top k nearest neighbors. Suppose we set $k = 8$, and among the eight nearest neighbors, three of them belong to female *Cx. stigmatosoma*, one belongs to female *Ae. aegypti*, and four belong to male *Cx. tarsalis*. We can then calculate the conditional probability using equation (2) as

$$P(\mathrm{X}|\text{female } Cx.\,stigmatosoma) = \frac{3}{8}$$

$$P(X|\text{female } Ae.\,aegypti) = \frac{1}{8}$$

$$P(X|\text{male } Cx.tarsalis) = \frac{4}{8}$$

These conditional probabilities are then multiplied by the class *prior* probability to calculate the *posterior* probability, and the observed insect is predicted to be the class which has the highest *posterior* probability. As such, we are able to estimate the class-conditioned probability for features in any format, including the feature of distance returned from an opaque similarity function, and thus generalize the Bayesian classifier to subsume some of the advantages of the nearest neighbor classifier.

Table 1 outlines the Bayesian classification algorithm. The algorithm begins in Lines 1-3 by estimating the *prior* probability for each class. This is done by counting the number of occurrences of each class in the training data set. It then estimates the conditional probability for each unknown data using the kNN approach. Specifically, given an unknown insect sound, the algorithm first searches the entire training data to find the top *k* nearest neighbors using some distance measure (Lines 5-9); it then counts for each class the number of neighbors which belong to that class and calculates the class-conditioned probability. With the prior probability and the class-conditioned probability known for each class, the algorithm calculates the *posterior* probability for each class (Lines 13, 15-18) and predicts the unknown data to belong to the class that has the highest posterior probability (Line 19).

Table 1: The Bayesian Classification Algorithm Using a High-dimensional Feature

Notation

k: the number of nearest neighbors in kNN approach

disfunc: a distance function to calculate the distance between two data

C: a set of classes

TRAIN: the training dataset

T_{Ci}: number of training data that belong to class C_i

```
1   for i = 1 : |C|
2       P(C_i) = T_Ci / |TRAIN|;            //estimate prior probability
3   end
4   for each unknown data F_1
5       for j =1: |TRAIN|
6           d(j) = disfunc(F_1, TRAINj); //the distance of F_1to each training data
7       end
8       [d, sort_index] = sort(d, 'ascend'); //sort the distance in ascending order
9       top_k = sort_index(1 to k);          // find the top-k nearest neighbors
10      for  i = 1 : |C|
11          k_Ci = number of data in top_k that are labeled as class C_i;
12      P(F_1|C_i) = k_Ci / k;   // the conditional probability with kNN approach
13          P(C_i|F_1) = P(C_i) P(F_1|C_i);      // calculate posterior probability
14      end
15      normalize_factor = sum_{i=1}^{i=|C|} P(C_i|F_1) //normalize the posterior probability
16      for i = 1 : |C|
17          P(C_i|F_1) = P(C_i|F_1)/normalize_factor;
18      end
19      Ĉ = argmax_{C_i∈C} P(C_i|F_1)     // assign the unknown data F_1 to the class Ĉ
20  end
```

The algorithm outlined in Table 1 requires two inputs, including the parameter *k*. The goal is to choose a value of *k* that minimizes the probability estimation error. One way to do this is to use *validation* [32]. The idea is to keep part of the training data apart as

validation data, and evaluate different values of *k* based on the estimation accuracy on the validation data. The value of *k* which achieves the best estimation accuracy is chosen and used in classification. This leaves only the question of which *distance measure* to use, that is, how to decide the distance between any two insect sounds. Our empirical results showed that a simple algorithm which computes the Euclidean distance between the truncated frequency spectrums of the insect sounds works quite well. Our distance measure is further explained in Table 2. Given two flying sounds, we first transform each sound into frequency spectrums using DFT (Lines 1-2). The spectrums are then truncated to include only those corresponding to the frequency range from 100 to 2,000 (Lines 3-4); the frequency range is thus chosen, because according to entomological advice[4], all other frequencies are unlikely to be the result of insect activity, and probably reflect noise in the sensor. We then compute the Euclidean distance between the two truncated spectrums (Line 5) and return it as the distance between the two flying sounds.

Table 2: Our Distance Measure for two Insect Flight Sounds

Notation:
S_1,S_2: two sound sequences
dis: the distance between the two sounds

	function dis = disfunc(S_1,S_2)
1	spectrum1 = DFT(S1);
2	spectrum2 = DFT(S2);
3	truncateSpectrum1 = spectrum1(frequency range= [100, 2000]);
4	truncateSpectrum2 = spectrum2(frequency range= [100, 2000]);
5	$\text{dis} = \sqrt{\sum(\text{truncateSpectrum}_1 - \text{truncateSpectrum}_2)^2}$

[4] Many large insects, i.e. most members of Odonata and/or Lepidoptera, have wingbeat frequencies that are significantly slower than 100 Hz; our choice of truncation level reflects our special interest in Culicidae.

Our flying-sounds-based insect classification algorithm is obtained by 'plugging' the distance measure explained in Table 2 into the Bayesian classification framework outlined in Table 1. To demonstrate the effectiveness of the algorithm, we considered the data that was used to generate the plot in Figure 1. These data were randomly sampled from a dataset with over 100,000 sounds generated by our sensor. We sampled in total 3,000 flying sounds, 1,000 sounds for each species, so the *prior* probability for each class is one-third. Using our insect classification algorithm with k set to eight, which was selected based on the validation result, we achieved an error rate of 12.43% using leave-one-out. We then compared our algorithm to the *optimal* result possible using only the wingbeat frequency, which is the most commonly used approach in previous research efforts. The optimal Bayes error-rate to classify the insects using wingbeat frequency is 18.13%, which is the lower bound for any algorithm that uses just that feature. This means that using the truncated frequency spectrum is able to reduce the error rate by almost a third. To the best of our knowledge, this is the first explicit demonstration that there is exploitable information in the flight sounds *beyond* the wingbeat frequency.

It is important to note that we do not claim that the distance measure we used in this work is optimal. There may be better distance measures, especially if we are confining our attention to *just* Culicidae or *just* Tipulidae, etc. However, if and when a better distance measure is found, we can simply 'plug' the distance measure in the Bayesian classification framework to get a better classification performance.

2.4.3. Additional Features: Circadian Rhythm of Flight Activity

In addition to the insect flight sounds, there are other features that can be used to reduce the error rate. The features can be very cheap to obtain, as simple as noting the *time-of-intercept*, yet the improvement can be significant.

It has long been noted that different insects often have different circadian flight activity patterns [56], and thus the time when a flying sound is intercepted can be used to help classify insects. For example, house flies (*Musca domestica*) are more active in the daytime than at night, whereas *Cx. tarsalis* are more active at dawn and dusk. If an unknown insect sound is captured at noon, it is more probable to be produced by a house fly than by *Cx. tarsalis* based on this *time-of-intercept* information.

Given an additional feature, we must consider how to incorporate it into the classification algorithm. One of the advantages of using the Bayesian classifier is that it offers a principled way to gracefully combine multiple features; thus, the solution is straightforward here. For simplicity, we temporarily assume the two features, the *insect-sound* and the *time-of-intercept*, are conditionally independent, i.e., they are independent given the class (we will revisit the reasonableness of this assumption later). With such an independence of feature assumption, the Bayesian classifier is called Naïve Bayesian classifier and is illustrated in Figure 5. The two arrows in the graph encode the fact that the probability of an unknown data to be in a class depends on the features F_1 and F_2, whereas the lack of arrows between F_1 and F_2 means the two features are independent.

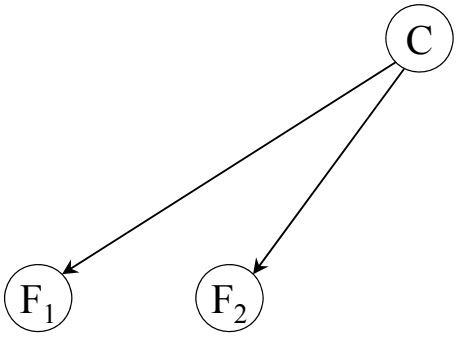


Figure 5: A Bayesian network that uses two independent features for classification

An observed object X now should include two values, f_1 and f_2, and the posterior probability of X belonging to a class C_i is calculated as $\mathrm{P}(C_i|X) = \mathrm{P}(C_i|\mathrm{F}_1 = f_1, \mathrm{F}_2 = f_2)$. With the independence assumption, this probability is proportional to:

$$\mathrm{P}(C_i|\mathrm{F}_1 = f_1, \mathrm{F}_2 = f_2) \propto \mathrm{P}(C_i)\mathrm{P}(\mathrm{F}_1 = f_1|C_i)\mathrm{P}(\mathrm{F}_2 = f_2|C_i) \quad (3)$$

Where $\mathrm{P}(\mathrm{F}_\mathrm{j} = f_j|\mathrm{C}_\mathrm{i})$ is the probability of observing the feature-value pair $\mathrm{F}_\mathrm{j} = f_j$ in class C_i. For concreteness, the feature F_1 in our algorithm is the insect sound, and F_2 is the time when the sound was produced.

In the previous section, we have shown how to calculate the class-conditioned probability of the *insect-sound*, $P(\mathrm{F}_1 = f_1|C_i)$, using the kNN estimation method and the *prior* probability $P(C_i)$. To incorporate our additional feature, we only need to calculate the class-conditioned probability $P(\mathrm{F}_2 = f_2|C_i)$. Note that the *time-of-intercept* is a scalar. As we discussed in the section above, learning the class-conditioned distributions of a one-dimensional feature can be easily done by constructing a histogram.

This histogram of the *time-of-intercept* for a species is simply the insect's flight activity circadian rhythm. In the literature, there have been many attempts to quantify these patterns for various insects. However, due to the difficulty in obtaining such data, most attempts were made by counting 1 if there is activity observed in a time period (such as a ten-minute window) and 0 otherwise [57][50]. Without distinguishing the number of observations in each period, the resulting patterns have a course granularity. In contrast, by using our sensors we have been able to collect on the order of *hundreds of thousands* of observations per species, and count the *exact* number of observations at sub second granularity, producing what we believe are the "densest" circadian rhythms ever recorded. Figure 6 shows the flight activity circadian rhythms of *Cx. stigmatosoma* (female), *Cx. tarsalis* (male), and *Ae. aegypti* (female). Those circadian rhythms were learned based on observations collected over one month. The results are consistent with the report in [35][57], but with a much finer temporal resolution (down to minutes). Note that although all three species are most active at dawn and dusk, *Ae. aegypti* females are significantly more active during daylight hours.

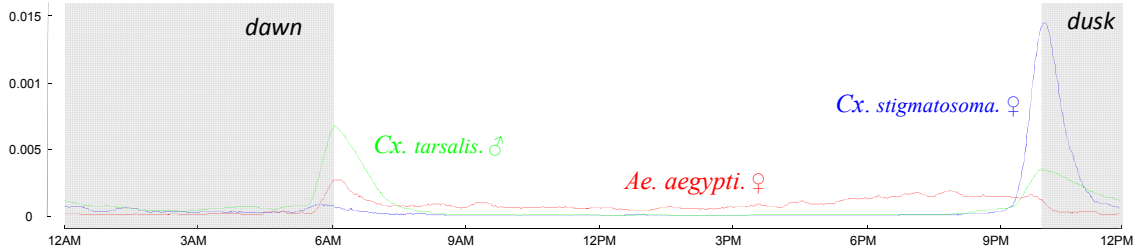


Figure 6: The flight activity circadian rhythms of *Cx. stigmatosoma* (female), *Cx. tarsalis* (male), and *Ae. aegypti* (female), learned based on observations generated by our sensor that were collected over one month

For the insects discussed in this work, we constructed circadian rhythms based on many hundreds of thousands of individual observations. However, it is obvious that as our sensors become more broadly used by the community, we cannot always expect to have such fine-grained data. For example, there are approximately 3,500 mosquito species worldwide; it is unlikely that high quality circadian rhythms for all of them will be collected in the coming years. However, the absence of "gold standard" circadian rhythms should not hinder us from using this useful additional feature. Instead, we may consider using approximate rhythms.

One such idea is to use the circadian rhythms of the most closely related insect species, taxonomically, for which we *do* have data. For example, suppose we do not have the circadian rhythm for *Cx. stigmatosoma*, we can use the rhythm of *Cx. tarsalis* as an approximation if the latter is available.

In the cases where we do not have the circadian rhythms for any taxonomically-related insects, we can construct approximate rhythms simply based on *text* descriptions in the entomological literature. Some frequently encountered descriptions of periods when insects are active include *diurnal*, *nocturnal*, *crepuscular*, etc. Offline, we can build a dictionary of templates based on these descriptions. This process of converting text to probabilities is of course subjective; however, as we will show below, it does lead to improvements over using no time-of-day information. Our simple first attempt at this work is by quantifying different levels of activities with numbers from 1 to 3,

representing, *low*, *medium*, and *high* activity[5]. For example, if an insect is described as diurnal and most active at dawn and dusk, we can use these three degrees to quantify the activities: highest degree at dawn and dusk, second in the daytime, and low activity during the night. The resulting template is shown in Figure 7.IV. Note that a circadian rhythm is a probability distribution that tells how likely we are to capture a certain insect species' flights at a certain time, and thus each template is normalized such that the area under the template sums to one. In Figure 7.II, we show an approximate circadian rhythm for *Cx. stigmatosoma* that we constructed this way. We spend two minutes searching the web for an academic paper that describes *Cx. stigmatosoma* flight activity, discovering that according to [37], *Cx. stigmatosoma* is active at dawn and dusk (*crepuscular*).

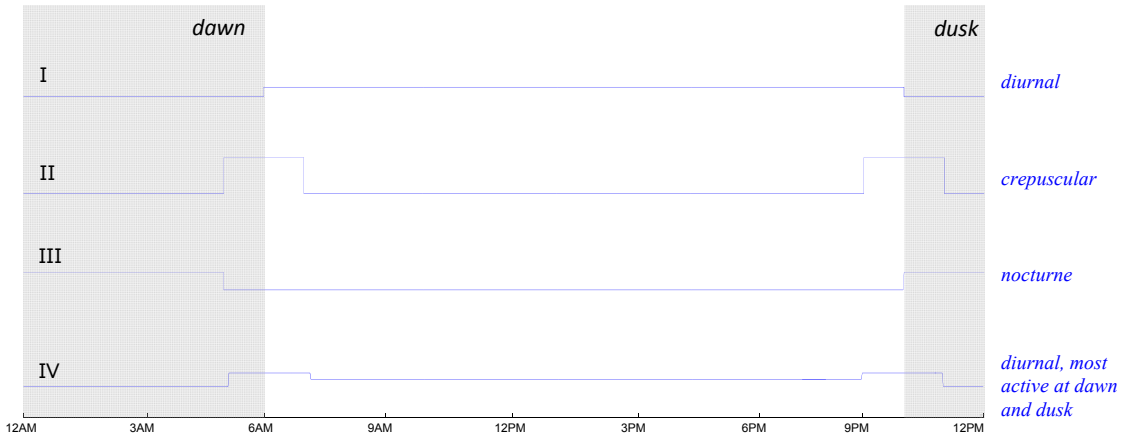


Figure 7: Examples of approximation templates for insects' flight activity circadian rhythm. No markings are shown on the Y-axis; all templates are normalized such that the area under the curve is one, and the smallest value is an epsilon greater than zero

[5] The lowest level of flight activity we represent is *low*, but not *zero*. In a Bayesian classifier, we never want to assign zero probabilities to an event, unless we are sure it is logically impossible to occur. In practice, a technique called Laplacian correction is typically used to prevent any probability estimate from being exactly zero.

In the worst case, if we cannot glean any knowledge about the insect's circadian rhythm (e.g. a newly-discovered species), we can simply use a constant line as an approximation. The constant line encodes no information about the activity hours of that insect, but it enables the incorporation of the more familiar insects' circadian rhythms into the classifier to improve performance. In the pathological case where all the circadian rhythms are approximated using a constant line, the classifier degenerates to a Bayesian classifier that does not use this additional feature.

Given the above, we can almost always incorporate some of the circadian rhythm information into our classifier. Given a *time-of-intercept* observation, we can calculate the class-conditioned probability of observing the activity from a species simply by looking up the flight activity circadian rhythms of that species. For example, suppose an insect sound was detected at 6:00am, the probability of observing this activity from a *Cx. tarsalis* male is three times the probability of observing it from an *Ae. aegypti* female, according to the circadian rhythms shown in Figure 6.

For concreteness, the insect classification algorithm that uses two features is outlined in Table 3. It is similar to the algorithm outlined in Table 1 that uses a single feature. Only five modifications are made.

Table 3: The Insect Classification Algorithm Using two Features

This algorithm is similar to the one outlined in Table 1. Only three modifications are needed, which are listed below. The modifications are highlighted in blue/bold.

1	Line 6: for each unknown data $\mathbf{[F_1, F_2]}$
2	Line 13: $P(C_i|\mathbf{F_1, F_2}) = P(C_i)\ P(F_1|C_i)\ \mathbf{P(F_2|C_i)}$; // calculate the *posterior* probability
3	Line 15: normalize_factor = $\sum_{i=1}^{i=\|C\|} \mathbf{P(C_i|F_1, F_2)}$ //normalize the posterior probability
4	Line 17: $\mathbf{P(C_i|F_1, F_2) = P(C_i|F_1, F_2)}$//normalize_factor;
5	Line 19: $\hat{C} = \underset{C_i \in C}{\text{argmax}}\ \mathbf{P(C_i|F_1, F_2)}$ // assign the unknown data X to the class $\hat{C}$

To demonstrate the benefit of incorporating the additional feature in classification, we again revisit the toy example in Figure 1. With the *time-of-intercept* feature incorporated and the accurate flight activity circadian rhythms learned using our sensor data, we achieve a classification accuracy of 95.23%. Recall that the classification accuracy using *just* the *insect-sound* is 87.57% (cf. the paragraph right below Table 2). Simply by incorporating this cheap-to-obtain feature, we reduce the classification error rate by about two-thirds, from 12.43% to only 4.77%.

To test the effect of using proxies of the learned flight activity circadian rhythm, we imagine that we do not have the flight activity circadian rhythm for *Cx. stigmatosoma* female, and that we must use one of the approximate rhythms discussed above. The results are shown in Table 4. As we can see, even with a constant line approximation, the classification accuracy is 88.73%, slightly better than not using the *time-of-intercept* feature. This is because, although the algorithm has no knowledge about the circadian rhythm for *Cx. stigmatosoma* females, it does have knowledge of the other two species' circadian rhythms. With the approximation created based on the text description, we

achieve an accuracy of 90.80%, which is better than using the constant line. This is as we hoped, as the approximate rhythm carries some useful information about the insects' activity pattern, even though it is at a very coarse granularity. An even better classification accuracy of 93.87% is achieved by using the circadian rhythm of *Cx. tarsalis* males as the approximation. As can be seen from Figure 6, the circadian rhythm of *Cx. tarsalis* males is quite close to that of *Cx. stigmatosoma* females.

Table 4: Classification Performance using Different Approximations for *Cx. stigmatosoma* (female) Flight Activity Circadian Rhythm

Flight activity circadian rhythm approximations	No rhythm used	constant line approximation	description based approximation	using the taxonomically-related insect's rhythm	learned using our sensor data
Classification Accuracy	87.57%	88.73%	90.80%	93.87%	95.23%

Note that the classification accuracy with *any* approximation is worse than that of using the accurate circadian rhythm (95.23% accuracy). This is not surprising, as the more accurate the estimated distribution is, the more accurate the classification will be. This reveals the great utility of our sensor: it allows the inexpensive collection of massive amounts of training data, which can be used to learn accurate distributions.

2.4.4. A Tentative Additional Feature: Geographic Distribution

In addition to the *time-of-intercept*, we can also use the *location-of-intercept* as an additional feature to reduce classification error rate. The *location-of-intercept* is also very cheap-to-obtain. It is simply the location where the sensor is deployed.

We must preempt a possible confusion on behalf of the reader. One application of our sensors is estimating the relative abundance of various species of insects at some

particular location. However, we are suggesting here that we can use estimates of the relative abundance of the species of insects at that location to do this more accurately. This appears to be a "chicken and egg paradox." However, there *is* no contradiction. The classifier is attempting to optimize the accuracy of its *individual* decisions about each particular insect observed, and knowing, even approximately, the expected prevalence of each species can improve accuracy.

The *location-of-intercept* carries useful information for classification because insects are rarely evenly distributed at any spatial granularity we consider. For example, *Cx. tarsalis* is relatively rare east of the Mississippi River [47], whereas *Aedes albopictus* (the Asian tiger mosquito) has now become established in most states in that area[41]. If an insect is captured in some state east of the Mississippi River, it is more probable to be an *Ae. albopictus* than a *Cx. tarsalis*. At a finer spatial granularity, we may leverage knowledge such as "since this trap is next to a dairy farm (an animal manure source), we are five times more likely to see a *Sylvicola fenestralis* (Window Gnat), than an *Anopheles punctipennis.*"

A Bayesian classifier that uses three features is illustrated in Figure 8. Here, we again assume that all the three features, *insect-sound*, *time-of-intercept*, and *location-of-intercept*, are independent.

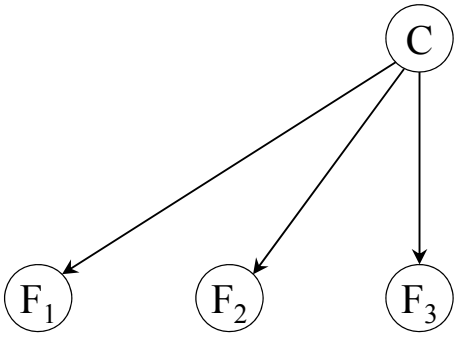


Figure 8: A Bayesian network that uses three independent features for classification

Based on Figure 8, the probability of an observed object X belonging to a class C_i is calculated as:

$$\mathrm{P}(C_i|\mathrm{F}_1 = f_1, \mathrm{F}_2 = f_2, \mathrm{F}_3 = f_3) \propto \mathrm{P}(C_i)\mathrm{P}(\mathrm{F}_1 = f_1|C_i)\mathrm{P}(\mathrm{F}_2 = f_2|C_i)\mathrm{P}(\mathrm{F}_3 = f_3|C_i) \quad (4)$$

Where $P(\mathrm{F}_3 = f_3|C_i)$ is the probability of observing an insect from class C_i at location f_3. This probability reflects the geographic distribution of the insects. For classification, we do not need the true *absolute* densities of insect prevalence; we just need the *ratio* of densities for the different species at the observation location. This is because with the ratio of $P(\mathrm{F}_3 = f_3|C_i)$, we can calculate the ratio of the posterior probability $\mathrm{P}(C_i|\mathrm{F}_1 = f_1, \mathrm{F}_2 = f_2, \mathrm{F}_3 = f_3)$ of each species, and predict an observation to belong to the species which has the highest posterior probability. In the case where we do need the actual posterior probability values, we can always calculate them from the posterior probability ratio, based on the constraint that the sum of all posterior probabilities over different classes should be one, as shown in Lines 15-18 in Table 1.

To obtain the ratio of $\mathrm{P}(\mathrm{F}_3 = f_3|C_i)$ at a given a location is simple. We can glean this information from the text of relevant journal papers or simply from the experiences of

local field technicians. For example, suppose we deploy our insect sensor at a location where we should expect to be twice as likely to encounter *Cx. tarsalis* as *Cx. stigmatosoma,* the ratio of *Cx. tarsalis* to *Cx. stigmatosoma* is 2:1. In the case where we cannot glean any such knowledge about the local insect population, we can temporarily augment our sensor with various insect traps that physically capture insects (i.e. CDC trap for mosquitos, "yellow sticky cards" traps for sharpshooters, etc.), and use these manually counted number of observations of each species to estimate the ratios.

To demonstrate the utility of incorporating this *location-of-intercept* feature, we did a simple *simulation* experiment. Note that the *insect-sound* and *time-of-intercept* features are *real* data; only the *location-of-intercept* was simulated by assuming there are two species of insects, *Cx. stigmatosoma* (female) and *Ae. aegypti* (female), which are geographically distributed as shown in Figure 9. We further assumed our sensors are deployed at three different locations, S_1, S_2 and S_3, where S_2 is about the same distance from both centers, and S_1 and S_3 are each close to one of the centers. Here, we model the location distributions as Gaussian density "bumps" for simplicity; however, this is not a necessary assumption, but we can use any density distribution.

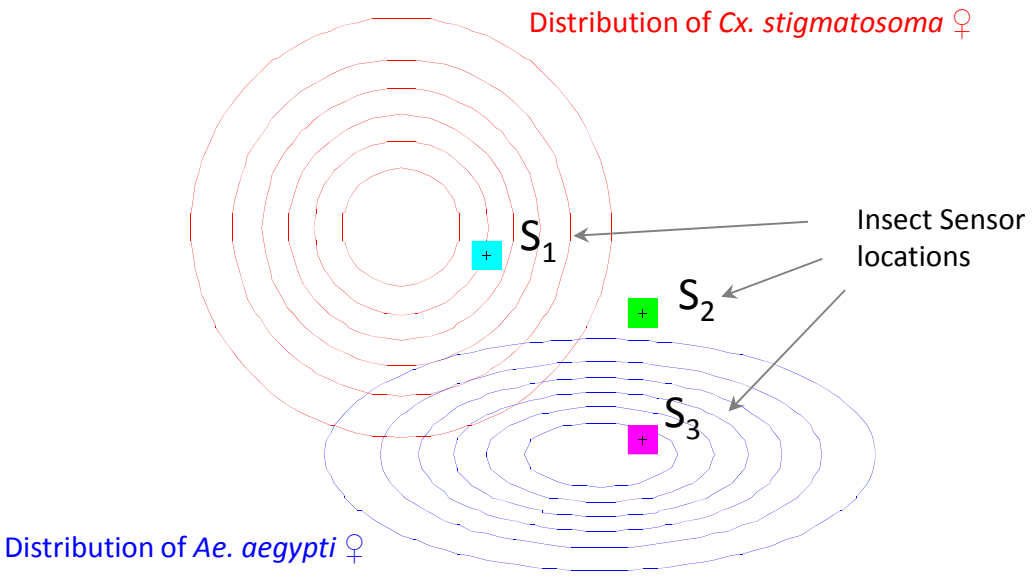


Figure 9: The assumptions of geographic distributions of each insect species and sensor locations in our simulation to demonstrate the effectiveness of using *location-of-intercept* feature in classification

To simulate the data captured by the three sensors, we project ten thousand insect exemplars of each species onto the map according to the geographic distribution assumption. We then sample these insects that are within the capture range of the sensors, which is assumed to be a square region centering at the sensor, as shown in Figure 9. Each sampled insect is assumed to fly across our sensor once and have its data captured.

In our experiment, we sampled 277 *Cx. stigmatosoma* females and 3 *Ae. aegypti* females at location S_1, 24 and 44 at location S_2, and 4 and 544 at location S_3. Using just the *frequency spectrum* and the *time-of-intercept* to classify those sampled insects, we achieved an error rate of 5.41%. However, by incorporating the *location-of-intercept*, we

reduced the error rate to 2.54%. This is impressive, as the *location-of-intercept* information is very cheap-to-obtain, yet it reduced the error rate by more than half.

2.4.5. A General Framework for Adding Features

There may be dozens of additional features that could help improve the classification performance. In this section, we generalize our classifier to a framework that is easily extendable to incorporate arbitrarily many features.

With n independent features, the posterior probability that an observation belongs to a class C_i is calculated as:

$$\mathrm{P}(C_i|\mathrm{F}_1 = f_1, \mathrm{F}_2 = f_2, \dots, \mathrm{F}_\mathrm{n} = f_n) \propto \mathrm{P}(C_i) \prod\nolimits_{j=1}^{n} \mathrm{P}(\mathrm{F}_\mathrm{j} = f_j|C_i) \quad (5)$$

Where $\mathrm{P}(\mathrm{F}_\mathrm{j} = f_j|C_i)$ is the probability of observing f_j in class C_i.

Note that the posterior probability can be calculated incrementally as the number of features increases. That is, if we have used some features to classify the objects, and later on, we have discovered more useful features and would like to add those new features to the classifier to re-classify the objects, we do not have to re-compute the entire classification from scratch. Instead, we can keep the posterior probability obtained from the previous classification (based on the old features), update each posterior probability by multiplying it with the corresponding class-conditioned probability of the new features, and re-classify the objects using the new posterior probabilities.

In our discussions thus far, we have assumed that all the features are independent given the class. In [17], it was shown that this independence assumption is reasonable for the Bayesian classifier to work well. However, it is also possible that users may wish to

use features that clearly violate the independence assumption in our general framework. For example, if the sensor was augmented to obtain *insect mass* (a generally useful feature), it is clear from basic principles of allometric scaling that the frequency spectrum feature would *not* be independent [20]. The good news is that as shown in Figure 10, the Bayesian network can be generalized to encode the dependencies among the features. In the cases where there is clear dependence between some features, we can consider adding an arrow between the dependent features to represent this dependence. For example, suppose there is dependence between features F_2 and F_3, we can add an arrow between them, as shown by the red arrow in Figure 10. The direction of the arrow represents *causality*. The only drawback to this augmented Bayesian classifier [31] is that more training data is required to learn the classification model if there are feature dependences, as more distribution parameters need to be estimated (e.g., the covariance matrix is required instead of just the standard deviation) .

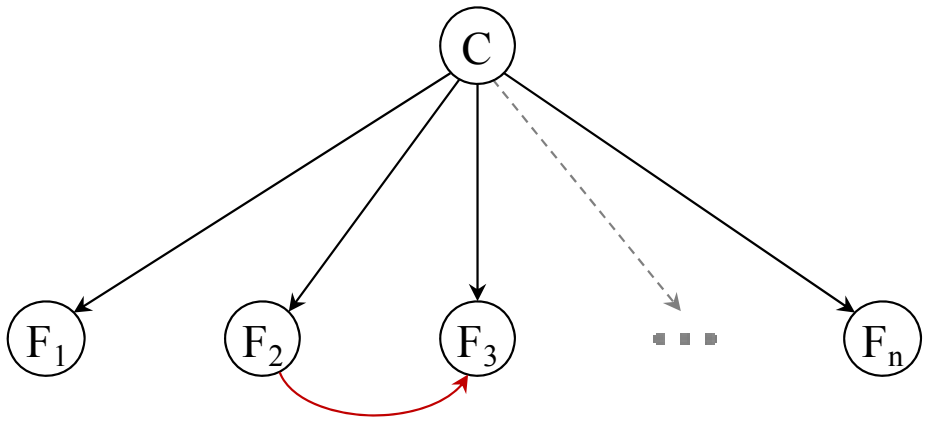


Figure 10: The Bayesian network that uses n features for classification, with feature $\mathbf{F_2}$ and $\mathbf{F_3}$ being conditionally dependent.

2.5 Experimental Evaluation

We begin by noting that *all* experiments are reproducible. All experimental code and data are archived in perpetuity at (supporting page).

2.5.1. A Case Study: Sexing Mosquitoes

Sexing mosquitoes is required in some entomological applications. For example, the Sterile Insect Technique, a method which eliminates large populations of breeding insects by releasing only sterile males into the wild, has to separate the male mosquitoes from the females before being released [42]. Here, we conducted an experiment to see how well it is possible to distinguish female and male mosquitoes from a single species using our proposed classifier.

In this experiment, we would like to distinguish male *Ae*. *aegypti* mosquitoes from females. The only feature used in this experiment is the *frequency spectrum*. We did not use the *time-of-intercept*, as there is no obvious difference between the flight activity circadian rhythms of the males and the females that belong to a same species (A recent paper offers evidence of minor, but measurable differences for the related species *Anopheles gambiae* [51]; however, we ignore this possibility here for simplicity). The data used were randomly sampled from a pool of over 20,000 exemplars. We varied the number of exemplars from each sex from 100 to 1,000 and averaged over 100 runs, each time using random sampling with replacement. The average classification performance using leave-one-out cross validation is shown in Figure 11.

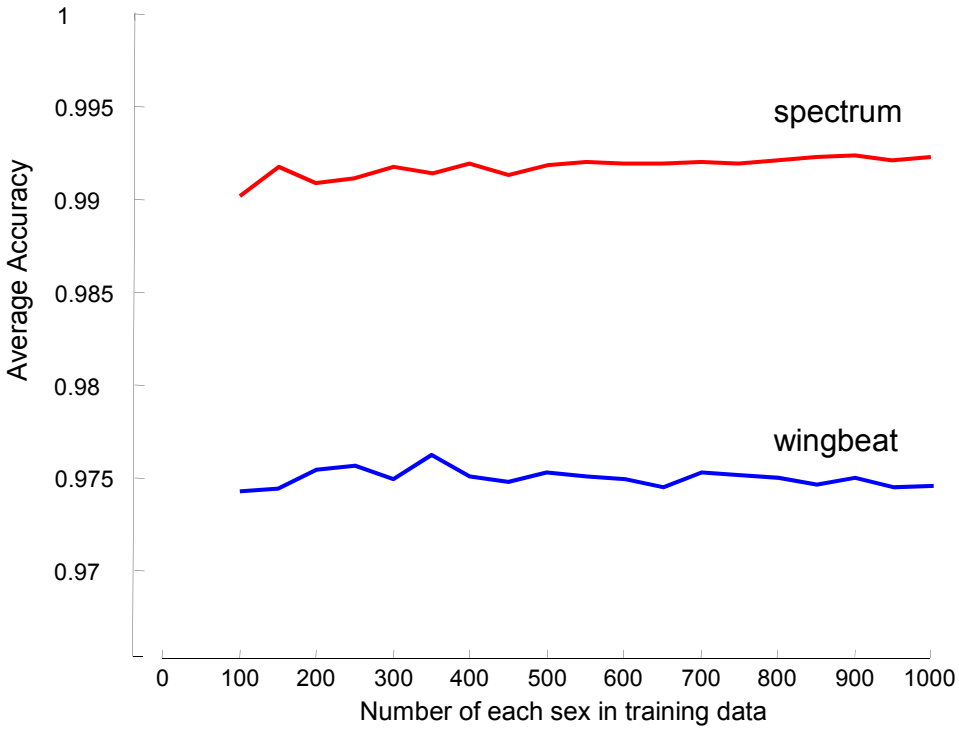


Figure 11: The classification accuracy of sex discrimination of *Ae. agypti* mosquitoes with different numbers of training data using our proposed classifier and the wingbeat-frequency-only classifier.

We can see that our classifier is quite accurate in sex separation. With 1,000 training data for each sex, we achieved a classification accuracy of 99.22% using *just* the truncated frequency spectrum. That is, if our classifier is used to separate 1,000 mosquitoes, we will make about eight misclassifications. Note that, as the amount of training data increases, the classification accuracy increases. This is an additional confirmation of the claim that more data improves classification [27].

We compared our classifier to the classifier using just the wingbeat frequency. As shown in Figure 11, our classifier consistently outperforms the wingbeat frequency classifier across the entire range of the number of training data. The classification

accuracy using the wingbeat classifier was 97.47% if there are 1,000 training data for each sex. Recall that the accuracy using our proposed classifier was 99.22%. By using the frequency spectrum instead of the wingbeat frequency, we reduced the error rate by more than two-thirds, from 2.53% to 0.78%. It is important to recall that in this comparison, the data and the basic classifier were *identical*; thus, all the improvement can be attributed to the additional information available in the frequency spectrum beyond just the wingbeat frequency. This offers additional evidence for our claim that wingbeat frequency by itself is insufficient for accurate classification.

In this experiment, we assume the cost of female misclassification (misclassifying a female as a male) is the same as the cost of male misclassification (misclassifying a male as a female). The confusion matrix of classifying 2,000 mosquitoes (equal size for each sex) with the same cost assumption from one experiment is shown in Table 5. I.

Table 5: (I) The confusion matrix for sex discrimination of *Ae. aegypti* mosquitoes with the decision threshold for female being 0.5 (i.e., same cost assumption). (II) The confusion matrix of sexing the same mosquitoes with the decision threshold for female being 0.1

I (Balanced cost)

		Predicted class	
		female	male
Actual class	female	993	7
	male	5	995

II (Asymmetric cost)

		Predicted class	
		female	male
Actual class	female	1,000	0
	male	22	978

However, there are cases in which the misclassification costs are asymmetric. For example, when the Sterile Insect Technique is applied to mosquito control, failing to release an occasional male mosquito because we mistakenly thought it was a female does

not matter too much. In contrast, releasing a female into the wild is a more serious mistake, as it is only the females that pose a threat to human health. In the cases where we have to deal with asymmetric misclassification costs, we can change the decision boundary of our classifier to lower the number of high-cost misclassifications in a principled manner. Of course, there is *no free lunch*, and a reduction in the number of high-cost misclassifications will be accompanied by an increase in the number of low-cost misclassifications.

In the previous experiment, with equal misclassification costs, an unknown insect is predicted to belong to the class that has the higher posterior probability. This is the equivalent of saying the threshold to predict an unknown insect as female is 0.5. That is, only when the posterior probability of belonging to the class of females is larger than 0.5 will an unknown insect be predicted as a female. Equivalently, we can replace Line 19 in Table 1 with the code in Table 6 by setting the threshold to 0.5.

Table 6: The decision making policy for the sex separation experiment

```
if ( P(female|X) ≥ threshold )
   X is a female
else
   X is a male
end
```

We can change the threshold to minimize the *total* cost when the costs of different misclassifications are different. In the Sterile Insect Technique, the goal is to reduce the number of female misclassifications. This can be achieved by lowering the threshold required to predict an exemplar to be female. For example, we can set the threshold to be

0.1, so that if the probability of an unknown exemplar belonging to a female is no less than this value, it is predicted as a female. While changing the threshold may result in a lower overall accuracy, as more males will be misclassified as females, it reduces the number of females that are misclassified as male. By examining the experiment summarized in Table 5. I, we can predict that by setting the threshold to be 0.1, we reduce the female misclassification rate to 0.075%, with the male misclassification rate rising to 0.69%. We chose this threshold value because it gives us an approximately one in a thousand chance of releasing a female. However, any domain specific threshold value can be used; the practitioner simply needs to state her preference in one of two intuitive and equivalent ways: "*What is the threshold that gives me a one in* (some value) *chance of misclassifying a female as a male*" or "*For my problem, misclassifying a male as a female is* (some value) *times worse than the other type of mistake, what should the threshold be?*"[22].

We applied our 0.1 threshold to the data which was used to produce the confusion matrix shown in Table 5.I and obtained the confusion matrix shown in Table 5.II. As we can see, of 2,000 insects in this experiment, twenty-two males, and *zero* females where misclassified, numbers in close agreement to theory.

2.5.2. Insect Classification with Increasing Number of Species

When discussing our sensor/algorithm, we are invariably asked, "*How accurate is it*?" The answer to this depends on the insects to be classified. For example, if the classifier is used to distinguish *Cx. stigmatosoma* (female) from *Cx. tarsalis* (male), it can

achieve near perfect accuracy as the two classes are radically different in their wingbeat sounds; whereas when it is used to separate *Cx. stigmatosoma* (female) from *Ae. aegypti* (female), the classification accuracy will be much lower, given that the two species have quite similar sounds, as hinted at in Figure 1. Therefore, a single absolute value for classification accuracy will not give the reader a good intuition about the performance of our system. Instead, in this section, rather than reporting our classifier's accuracy on a fixed set of insects, we applied our classifier to datasets with an incrementally increasing number of species and therefore increasing classification difficulty.

We began by classifying just two species of insects; then at each step, we added one more species (or a single *sex* of a sexually dimorphic species) and used our classifier to classify the increased number of species. We considered a total of ten classes of insects (different sexes from the same species counting as different classes), 5,000 exemplars in each class. Our classifier used both *insect-sound* (frequency spectrum) and *time-of-intercept* for classification. The classification accuracy measured at each step and the relevant class added is shown in Table 7. Note that the classification accuracy at each step is the accuracy of classifying all the species that come *at* and *before* that step. For example, the classification accuracy at the last step is the accuracy of classifying *all* ten classes of insects.

Table 7: Classification accuracy with increasing number of classes

Step	Species Added	Classification Accuracy		Step	Species Added	Classification Accuracy
1	*Ae. aegypti* ♂	N/A		6	*Cx. quinquefasciatus* ♂	92.69%
2	*Musca domestica*	98.99%		7	*Cx. stigmatosoma* ♀	89.66%
3	*Ae. aegypti* ♀	98.27%		8	*Cx. tarsalis* ♂	83.54%
4	*Cx. stigmatosoma* ♂	97.31%		9	*Cx. quinquefasciatus*♀	81.04%
5	*Cx. tarsalis* ♀	96.10%		10	*Drosophila simulans*	79.44%

As we can see, our classifier achieves more than 96% accuracy when classifying no more than five species of insects, significantly higher than the default rate of 20% accuracy. Even when the number of classes considered increases to ten, the classification accuracy is never lower than 79%, again significantly higher than the default rate of 10%. Note that the ten classes are not easy to separate, even by human inspection. Among the ten species, eight of them are mosquitoes; six of them are from the same genus.

2.6 Conclusions

In this work we have introduced a sensor/classification framework that allows the inexpensive and scalable classification of flying insects. We have shown experimentally that the accuracies achievable by our system are good enough to allow the development of commercial products and to be a useful tool for entomological research. To encourage the adoption and extension of our ideas, we are making all code, data, and sensor schematics freely available at the UCR Computational Entomology Page [16]. Moreover,

within the limits of our budget, we will continue our practice of giving a complete system (as shown in Figure 2) to any research entomologist who requests one.

Chapter 3: DTW-D: Semi-Supervised Learning of Time Series using a Single Example

In some cases, when dealing with new insect species, it may be necessary to bootstrap the modeling of the species by using just a handful of annotated examples to find more (unannotated) examples in the archives, a process known as semi-supervised learning. The semi-supervised learning process is not limited to the insect classification, but is generally used in many time series classification applications. For example, the increasing prevalence of wearable devices (smartphones, pacemakers, Google Glass etc.) has created a new urgency for time series classification algorithms, but such uses of classification are typically plagued by limited data. For instances, we cannot expect a user to perform a gesture one hundred times just to ensure a surfeit of training data for our favorite machine learning algorithm. And the obvious solution to these problems is semi-supervised learning.

However, as we shall show, direct applications of off-the-shelf semi-supervised learning algorithms do not typically work well for time series. The research community working on time series classification has typically only used the copious UCR Archive to test their algorithms, and has thus largely avoided this pressing problem. In this work we first explain why semi-supervised learning algorithms typically fail for time series problems, and we then introduce a simple but very effective fix. Finally, we demonstrate our ideas on diverse real word problems.

This chapter is organized as follows: in Section 3.1, we briefly introduce our observation and our idea. Then we review the related work of this topic in Section 3.2. In Section 3.3, we introduce the definitions and notations used in this chapter. We then illustrate the proposed idea in Section 3.4 and formalize the proposed algorithm in Section 3.5. We present a detailed empirical evaluation of our ideas in Section 3.6. Finally, in Section 3.7, we offer conclusions and directions for future work.

3.1 Introduction

Time series classification has been an active area of research in the last two decades [64][70][75]. Two related conclusions have begun to emerge as a consensus in the community. First, while there is a plethora of classification algorithms in the literature, the nearest neighbor algorithm seems particularly suited to the unique structure of time series, and virtually all competitive attempts at time series classification use it [102]. Second, while there is also a surfeit of possible distance measures for time series, Dynamic Time Warping (DTW), a technique from the dawn of computing, is exceptionally difficult to beat [70]. In particular, a recent paper tested the most cited distance measures on 47 different datasets, and no method consistently outperforms DTW. Thus recent papers that claim improvements over DTW must resort to very powerful statistical tests to demonstrate *tiny* improvements in accuracy.

In the last decade, virtually all of the community has used the UCR Archive to test their algorithms [77]. We believe that the availability of this (admittedly very useful) resource has isolated much of the research community from the following reality, *labeled*

time series data is often *very* difficult to obtain. For example, in many situations, from medicine [87] to astronomy [89], obtaining labeled data requires the time and attention of a busy domain expert.

The obvious solution to this problem may appear to be the application of semi-supervised learning; however, direct applications of off-the-shelf semi-supervised learning algorithms do not typically work well for time series. In this work we make several related contributions. We explain *why* semi-supervised learning algorithms typically fail for time series problems, and we introduce a simple but very effective fix.

While we defer a detailed explanation of our ideas until Section 3.4, we offer a quick and intuitive preview of our ideas here:

Under certain assumptions, unlabeled members of a circumscribed positive class may be closer to some unlabeled members of a diverse negative class than to the labeled positive data. This is true *even* under DTW. Nevertheless, unlabeled positive data tend to benefit more from using DTW than unlabeled negative examples. The amount of *benefit* from using DTW over Euclidean Distance (ED) is a meta-feature that can be exploited.

We illustrate this in Figure 12 where we show the hierarchical clustering of five objects under various measures. Two of the five objects are randomly chosen examples (red/bold) from class 3 of the Trace dataset [77]. The other three objects are simply random-walk time series (blue/light).

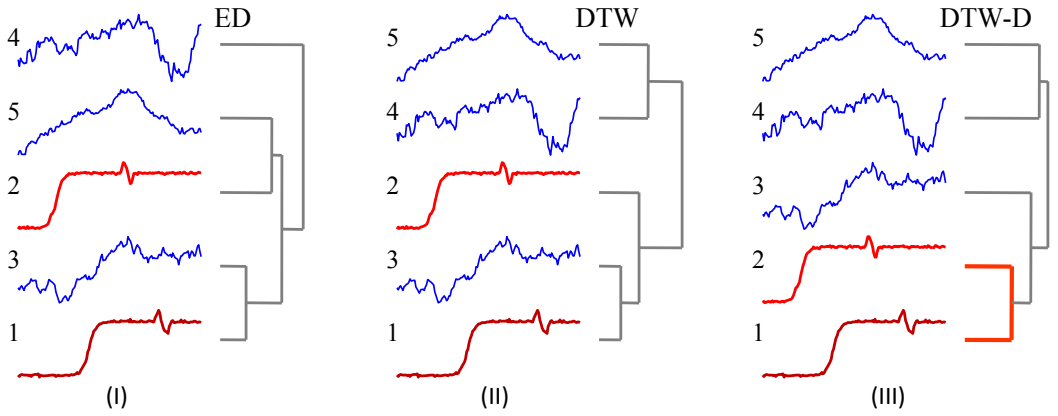


Figure 12: A complete linkage hierarchical clustering of two items from the Trace dataset with three random walks. From left to right, Euclidean distance (ED), Dynamic Time Warping (DTW) and Dynamic Time Warping-Delta (DTW-D), our proposed technique

As we can see, Euclidean Distance does poorly here. This is not surprising, since the Trace dataset is known to have classes that contain exemplars that are time-warped versions of a prototypical shape. Indeed, we see that DTW *does* manage to do better, reducing the distance between Trace-1 and Trace-2. However, this reduction is not enough; random-walk-3 is still closer to Trace-1 than Trace-2 is.

Our key observation is that moving from ED to DTW seems to help the true class data more than the random unstructured data. We can encode this difference/delta that DTW makes with DTW-D, the ratio of DTW over ED. And as we see in Figure 12.III, this *does* produce the correct clustering, at least in this example.

Imagine that we had been doing semi-supervised learning in this dataset using just Trace-1 as our sole positive example. For both ED and DTW, the very first item we added to the positive set would have been a false positive, and it would be very difficult for any algorithm to recover from this. In contrast, DTW-D would have correctly

suggested Trace-2 as the next item to add, and assuming only that we had good stopping criterion, we would have done very well.

The issue of a "good" stopping criterion we address uses a Minimum Description Length (MDL) inspired approach [72], which in essence stops the learning process when the next exemplar to be added does not compress well using the known positive data.

3.2 Related Work

There has been an enormous interest in time series classification over the past two decades. However, almost all research efforts assume that there are large amounts of labeled training data [91][96]. In reality, the high cost of labeling the data may render such an assumption invalid. For example, it requires the time and expertise of a cardiologist to annotate individual heartbeats in an ECG data trace [67], but a *single* polysomnography (sleep study) test may produce up to 40,000 such heartbeats. Given that the *acquisition* of unlabeled data is trivial, there is abundance of unlabeled data readily available. For instance, PhysioBank contains over 36,000 recordings of digitized polysomnographic and other physiologic signals, only a tiny fraction of which are labeled [71]; likewise there are tens of millions of books, images, maps and historical manuscripts available on the internet [74], many of which could be fruitfully mined in the time series space [93][100], if only we had more *labeled* data (cf. Section 3.6.4).

Semi-supervised learning (SSL) is a learning paradigm useful in application domains in which labeled data are limited, but unlabeled data are plentiful [90][73][68]. The literature offers a plethora of SSL methods, among which, *self-training* is perhaps the

most commonly-used [83][94][69][103]. *Self-training* is a general framework with very few underlying assumptions. In self-training, a classifier is first trained with a small number of labeled data. It is then used to classify the unlabeled data, and adds the most confidently classified object into the labeled set. The classifier re-trains itself using the new labeled set and the procedure repeated until adding new objects to the labeled set does not increase the accuracy of the classifier or some other stopping criteria is met. This general review of SSL is necessarily brief; we refer the readers to [103] and the references therein for more details.

Recently, some SSL techniques explicitly designed for time series have been proposed. To our best knowledge, thus far there are only three approaches [82][92][85]. The first paper [82] proposed to iteratively expand the labeled set by adding the closest object that is classified as positive to the labeled set. The classifier considers all unlabeled data as negative, and uses the Euclidean Distance. As we shall show, and as has been noted elsewhere [70][102], the inferiority of Euclidean Distance to DTW is mitigated by large training set sizes. Conversely Euclidean Distance is much more brittle a measure than DTW for tiny datasets, which is of course exactly the situation we face here. Thus [92] proposed to build a SSL classifier using DTW distance. Although moving from ED to DTW helps to improve the accuracy of the classifier, the algorithm is still not accurate enough in most real applications (cf. Section 3.6.1).

More recently, the authors of [85] introduced a SSL technique that interleaves exemplar selection with feature selection, using the work of [82] as a starting point. The method of [85] improves the SSL algorithm, but still uses standard distance measures

(Euclidean Distance). As such it is orthogonal to our contribution, which demonstrates that a subtle change in the *distance measure* dwarfs all possible changes in the *algorithms*.

In retrospect, only *three* research efforts on SSL for time series is a surprisingly small number, given that both SSL and time series classification are very popular research topics. In this work we venture to claim that we understand *why* progress in this area has been so slow. In brief, our claim is that there is little utility in tweaking the architecture of the SSL algorithms for time series, as they are all condemned to perform poorly if they use DTW or Euclidean Distance. The contribution of this work is to show a simple but effective fix that *will* allow the existing SSL methods to work very well for time series. It is important to recognize that we are not claiming a contribution to SSL algorithms per se. Rather we will show that changing the distance function used in SSL algorithms can produce a remarkable improvement for time series.

Given the importance of SSL, it is somewhat surprising that the critical subroutine of finding a stopping point has been underexplored [65][66][85][86][92][101]. Most of the literature studying this problem suffer from some shortcomings which have limited their generalized adoption. For example, [101] suffers from the problem of *early stopping*, which means that the method is *overly* conservative, and misses many true positives. The approach in [92] has difficulty in identifying the positive and negative class boundaries, therefore is typically doomed to produce too many false negatives. A more recent work [85] proposed a cluster-based approach with the assumption that all the instances of the same local cluster have the same class label. Therefore, this method is prone to inherit the errors made by the clustering subroutine. In addition to this, because [85] uses K-means

algorithm, it inherits K-means lack of determinism. Because of the shortcomings of [85], [86] proposed an ensemble-based approach which performs the clustering process several times with different settings. In this ensemble approach, each instance is assigned a probabilistic confidence score, and based on these scores, an adaptive nearest neighbor classifier is constructed. However, this method is very complicated and requires extensive parameter tuning. Most recently there has been some work [65][66] based on the Minimum Description Length (MDL) for finding a parameter-free stopping criterion for SSL. Although this work leverages the intrinsic structure of the data, it is not suitable for datasets containing time-warped versions of a prototypical shape. This is because in the MDL framework the algorithm in [65][66] encodes the time series in the *Euclidean* space, which performs poorly for instances with warping. In addition to this, this algorithm assumes the number of the instances is known in *advance*, which is an unrealistic assumption if this number is too large. Finally, the performance of this algorithm depends directly on the order of the instances given by SSL, which in a sense makes the stopping criterion *post hoc*.

As we claimed above, our proposed DTW-D distance measure works well if the target concept contains time-warped versions of some prototype, and if the negative class occasionally produces objects close to positive class, even under DTW. Given these two core assumptions, we suggest that "*assuming only that we had good stopping criterion, we would have done very well*". Having expressed this motivation for finding a good stopping criterion for datasets with time-warped objects, in this paper we propose a Minimum Description Length (MDL) inspired stopping criterion based on the Run

Length Encoding of the warping path vector (we define *warping path vector* in the next section) between two time series instances.

3.3 Definitions and Notations

We begin by introducing all necessary notation and definitions. Although the algorithm presented in this work is applicable to all SSL methods, for ease of exposition, we present just the notations for Positive Unlabeled learning (PU learning), which is a collection of SSL methods that trains a classifier based on the positive (labeled) dataset and the unlabeled dataset only.

Definition 1: *P* is the set of training data, including all positively labeled objects.

P initially contains only a small number of labeled objects from the positive class, perhaps as few as one. As learning proceeds, the size of *P* increases as some of the previously unlabeled objects in *U* are labeled as positive and moved to *P*. Thus, *P* eventually contains both the original labeled objects, as well as the objects chosen by the classifier from the unlabeled dataset.

Definition 2: *U* is the set of unlabeled data.

Objects in dataset *U* can be from the positive class or the negative class. It is generally expected that the vast majority of *U* is from the negative class [103]. The goal of SSL is to map all the objects in *U* to the correct class so that the classifier is accurately trained with the classified objects. We denote individual time series objects from these two sets with subscripts, thus the i^{th} time series object in *P* is denoted P_i.

Rather than making a onetime explicit decision as to which objects from *U* should be added to *P*, most algorithms simply iteratively add the next most likely candidate [94][84][103]. This means that we must also specify a *stopping criterion* for the algorithm to predict that it has added all the unlabeled positive objects [103]. The problem of finding a good stopping criteria has attached significant research, with tentative solutions based on MDL, Bayesian information criterion, bootstrapping [95][103], etc. However, note that as we shall show in the empirical section, the difference our algorithm makes completely dwarfs any considerations of the optimal stopping criteria. That is to say, even if we did a post-hoc discovery of the optimal stopping criteria for the state-for-art rival, our method would have much higher accuracy for a huge range of "sub-optimal" stopping values.

Nevertheless, even given this, we feel it is still important to address the question of finding a good stopping criterion. As we discussed above, the state of the art algorithms to solve this problem have limited applicability, especially to dataset with any amount of warping, which is typical in most real-world datasets [70]. Our contribution to solving this problem is an MDL inspired algorithm based on Run Length Encoding of the minimum cost of the warping path of two time series.

The MDL principle is based on the intuition that the more regularities there are in a dataset, the more compressible it is. As data compression is equivalent to probabilistic prediction, MDL searches for a model with good predictive performance on unseen data. Thus MDL can be considered as a formalization of Occam's Razor, which states that the best model for a given data is the one that gives the best compression [72].

As we stated above, our approach is MDL inspired, but our domain of interest is real-valued time series, and classic MDL is defined in *discrete* space only, therefore we need to cast the real-valued time series to discrete domain.

In order to adopt MDL in our stopping criterion framework, we discretize the time series from real-valued domain using the function below:

Definition 3: *Discrete Normalization Function*

A Discrete Normalization function normalizes a time series subsequence T_S into b-bit discrete values in the range $[1,2^b]$. We define it as below:

$$\boldsymbol{DiscreteNorm(T_S) = round\left(\frac{T_S - min}{\max - min}\right) * \left(2^b - 1\right) + 1}$$

where *min* and *max* are the minimum and maximum values of T_S respectively.

We exploit the inherent regularity in the data in order to minimize the number of symbols needed to express it. Given two discretized time series DT_1 and DT_2, we encode their warping path vector using Run Length Encoding. We define the warping path vector as below:

Definition 4: *Warping Path Vector*

In order to align two discretized time series DT_1 and DT_2 of length n, Dynamic Time Warping constructs an n-by-n matrix where the $(i,j)^{th}$ element contains the distance $d(DT_{1_i}, DT_{2_j}) = (DT_{1_i} - DT_{1_i})^2$ between the two points DT_{1_i} and DT_{2_j}. A warping path vector V is a contiguous set of matrix elements which defines a mapping between DT_1 and DT_2. The k^{th} element of V is defined as $v_k = (i, j)_k$. Therefore we have:

$$V = v_1, v_2, v_{3,\ldots,} v_m \text{ , where } n \leq m \leq 2n\text{-}1$$

The warping path vector is subject to constraints of boundary, continuity, and monotonicity [70][79]. These conditions can be satisfied by exponentially many warping paths, however we are only interested in the path that minimizes the following warping cost:

$$DTW(DT_1, DT_2) = \min(\sqrt{\sum_{k=1}^{m} v_k})$$

This path can be found using dynamic programming [70][79]. From now on when we mention warping path vector, we mean the *minimum cost* warping path vector.

Definition 5: *Euclidean Warping Path Vector*

The Euclidean warping path vector is a special case where the k^{th} element of V is constrained such that $v_k = (i,j)_k$, where $i = j = k$.

Because our algorithm is MDL-inspired, we use a single time series as the model to encode the warping path vector between it and the other time series in question. We call this model a *hypothesis*.

Definition 5: *Hypothesis*

A hypothesis H is a time series subsequence used to encode other subsequences of the same length in the dataset.

We can measure the cost to represent data objects with the description length.

Definition 6: *Description Length*

A description length DL of a warping path vector V is the total number of bits required to represent it.

$$DL(V) = m * \log_2 c$$

where c is the cardinality of V.

We use Run Length Encoding to encode the warping path vector between two discretized time series. Consider the following toy example. Suppose we have two 3-bit time series as below:

$$DT_1 = [2\ 3\ 2\ 1\ 3\ 4] \text{ and } DT_2 = [1\ 2\ 5\ 4\ 3\ 7]$$

The cumulative distance matrix for these two time series comes from the recurrence below:

$$\chi(i,j) = d\left(DT_{1_i}, DT_{2_j}\right) + \min\{\chi(i-1,j-1), \chi(i-1,j), \chi(i,j-1)\}$$

where $\chi(i,j)$ is the cumulative distance which is the sum of the distance in the $(i,j)^{th}$ cell and the minimum of the cumulative distances in the adjacent cells. Having this recurrence, we have the distance matrices as below:

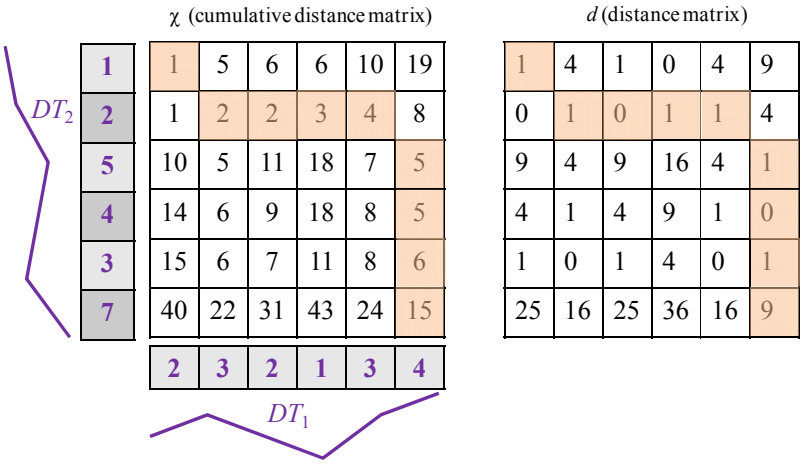


Figure 13: The cumulative distance matrix χ (*left*) and the distance matrix *d* (*right*) of DT_1 and DT_2. The highlighted cells form the minimum cost warping path

From Figure 13 we can see that the warping path vector V of DT_1 and DT_2 is, V = 1 1 0 1 1 1 0 1 9. The Euclidean warping path vector $V_{Euclidean}$ = 1 1 9 9 0 9. The raw description length of V is, $DL(V) = 9 * \log_2(8-1)^2$ = 51 bits. After encoding V by Run

Length Encoding, the encoded warping path vector becomes, $V_{encoded}$ = **1** 1 **0** *1* **1** *3* **0** *1* **1** *1* **9** *1*, where the red/**bold** numbers are symbols, and the green/*italicized* numbers are the counts. The description length of $V_{enccoded}$ is, $DL(V_{encoded}) = \left(\frac{m}{2}\right) * \{\log_2(1 + max_{symbol}) + \log_2 m\}$

$$= \left(\frac{9}{2}\right) * \{\log_2 10 + \log_2 9\} = 30 \text{ bits}$$

where max_{symbol} is the maximum of the symbols in V. Therefore run length encoding of V saved 21 bits. From $V_{encoded}$ we can see that there are five one-length runs and one three-length run.

For brevity, we do not explicitly define *time series*, *Euclidean distance* or *Dynamic Time Warping*, which in any case are rather well known. Instead we use the notation from [70], a heavily cited survey paper on these topics. We do note however the following useful fact that we later exploit, that the ED is an upper bound to the DTW. That is to say, for all x, y, we have DTW(x,y) ≤ ED(x,y).

3.4 DTW-D

To explain our observations and our key insight, we consider a concrete example. Let us imagine that we have target class of objects that are defined by having three periods of a sine wave. The instances may be corrupted by warping, different dampening rates, noise, minor changes to the starting phases etc., but as shown in Figure 14 they are unambiguously recognizable to the human eye.

In this example the negative class consists of just a constant line with the same mean as the positive class[6], corrupted by some noise.

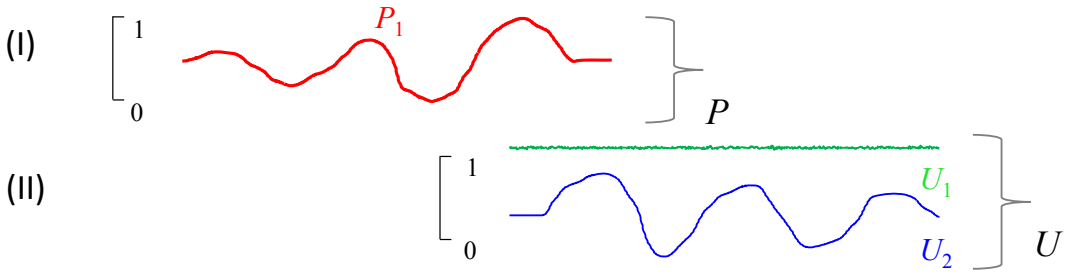


Figure 14: I) A labeled dataset P that consists of a single object P_1. II) The unlabeled dataset consist of a single true negative U_1 and a single true positive U_2

Suppose we ask any SSL algorithm to choose one object from U to add to P using the Euclidean distance. As we can see in Figure 15, U_1 is much closer to P_1 than U_2 is, thus our SSL algorithm would do poorly here.

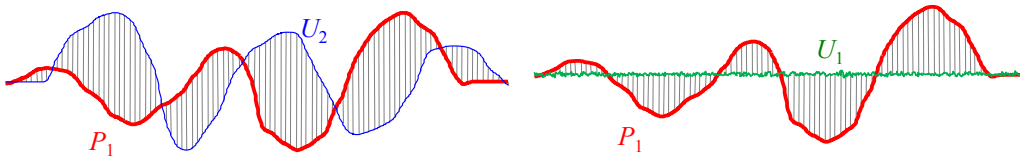


Figure 15: The Euclidean distance between two time series is proportional to the length of the gray hatch lines. It is easy to see that $ED(P_1,U_2) > ED(P_1,U_1)$

In retrospect this is not surprising. The brittleness of Euclidean distance to even small amounts of warping is well known, and explains the ubiquity of DTW in most research efforts [79][70][93]. By finding the optimal "peak-to-peak/valley-to-valley" alignment between two time series *before* calculating the distances, DTW is largely invariant to warping, as shown in Figure 16.

[6] In **Figure 14** the objects are shown to the same scale. They are *not* normalized for visual clarity. However, our analysis can be demonstrated for z-normalization, min/max normalization etc.

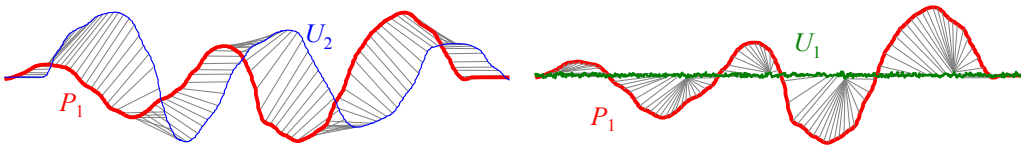


Figure 16: The DTW distance between two time series is proportional to the y-axis length of the gray lines

Unfortunately, while DTW helps significantly, as shown in Table 8, it is not enough: U_1 is *still* closer to P_1 than U_2 is, and the SSL algorithm would *still* pick the wrong object to move from U to P. Why did DTW not solve our problem? While DTW *is* invariant to warping, there are other differences between P_1 and U_2, including the fact that the first and last peaks have different heights. DTW cannot mitigate this.

Table 8: The Distance Matrices for the Three Objects in [P,U], under both the ED and DTW Distances

	ED			DTW		
	P_1	U_1	U_2	P_1	U_1	U_2
P_1	0	6.2	11	0	5.8	6.1
U_1		0	6.8		0	6.5
U_2			0			0

Moreover, the problem is compounded by the fact that U_1 is a "simple" shape. As pointed out in a recent paper, simple shapes tend to be "close to everything" [64]. Figure 15 gives a hint as to why this is true. The flat shape of U_1 means that no part of it is more than 0.5 away from any part of P_1. In contrast where P_1 and U_2 are out of phase, and the Euclidean distance is forced to match a peak to a valley, the distance can be as much as 0.8. This is why smooth, flat or least very slowly changing time series tend to be

(subjectively) surprisingly close to other objects [64]. This is a grave disappointment – this seems to be a dataset for which DTW is ideally suited, yet DTW fails here.

However, an examination of distance matrices shown in Table 8 *does* reveal an interesting fact. Moving from ED to DTW barely changed the distance between P_1 and U_1, but it did greatly affect the distance between P_1 and U_2. We can codify this with the following observation:

Observation 1: If a class is characterized by the existence of intra-class warping (possibly among other distortions [64]), then we should expect that moving from ED to DTW reduces distances more for intra-class comparisons than interclass comparisons.

To see this more clearly, we can consider the ratio of distance under DTW and ED as shown in Table 9.

Table 9: The Ratio of the ED and DTW Distances Shown in Table 8. The ratio is called DTW-D

	DTW/ED			DTW-D = DTW/ED		
	P_1	U_1	U_2	P_1	U_1	U_2
P_1	0	5.8/6.2	6.1/11	0	0.93	0.55
U_1		0	6.5/6.8		0	0.95
U_2			0			0

Note that if we consider the DTW-D *ratios*, we finally have P_1 and U_2 appear closer than P_1 and U_1. Figure 17 visualizes all three distance matrices with a complete linkage clustering.

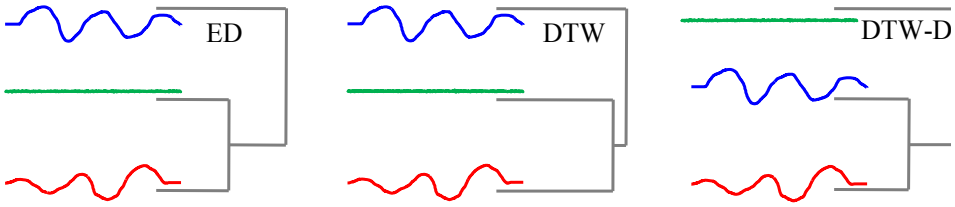


Figure 17: A visualization of the three distance matrices shown in Table 8 and Table 9 under complete linkage hierarchical clustering

Note that there is one minor special case we need to consider. If the ED is zero, then DTW-D would give a *divide-by-zero* error. We simply define this special case as having a value of zero. If the ED distance (and therefore also the DTW distance) between two objects is zero, it would be perverse to call them anything but the same class. This is a moot point, as we never expect observe perfect duplicates for real-values objects.

We have now concretely seen the problem with using ED/DTW for SSL, and our suggested fix, on a toy problem. However, it is natural to ask when this phenomenon actually occurs in the real-world, and would be amenable to our DTW-D solution. In the next section, we explicitly discuss our assumptions about when our ideas can help.

3.4.1. Two Key Assumptions

We do not claim our ideas will help for all time series problems. In particular, we are making two explicit assumptions which we will enumerate and discuss below. We will later show that these assumptions are very often true in real world domains.

Assumption 1: The positive class (the target concept) contains time warped versions of some platonic ideal (some prototypical shape), possibly with other types of noise/distortions.

Note that this assumption was true of our toy example in Figure 12. While all members of the Trace dataset have some noise, as shown in Figure 18, the most obvious variability between instances is in the timing of the onset of the "ramp-up" and the "oscillation" patterns. Dynamic time warping is able to compensate for and remove this variability.

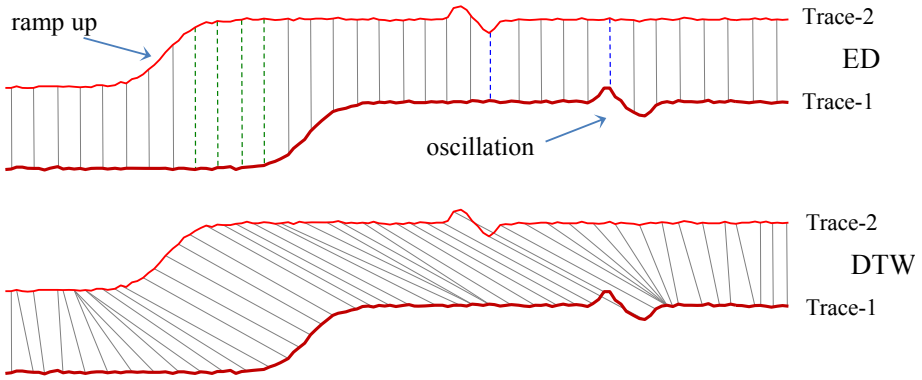


Figure 18**:** The two examples from "Trace" shown in Figure 12 compared under ED and DTW. In this plot, the distances are proportional to the variance of the y-axis lengths of the hatch lines. Thus the longer and shorter hatch lines in ED contribute to its large distance, whereas the y-axis lengths for DTW are almost the same, producing a small distance

This ability of DTW to compensate for the inherent warping in this class can produce a dramatic difference in classification accuracy. In the UCR Archive dataset the four-class Trace data is provided with a 100/100 train test split. The ED error rate on this dataset is 0.24, whereas DTW has an error-rate of 0.0. Since the exact same splits and classification algorithm (1NN) were used, and zero parameters are tuned for either approach, *all* of this difference can be attributed to the superiority of DTW over ED.

Assumption 2: The negative class may be very diverse, and occasionally by chance produces objects close to a member of the positive class, even under DTW.

Empirically, negative classes do tend to be diverse [81][97]. For example, there are only a limited number of ways an audio snippet can sound like a mosquito, but there are infinite ways a sound can be a non-mosquito (c.f. Section 6.1). Once again, this assumption was illustrated by our toy example in Figure 12. The random walk class is naturally very diverse, and it can (and did) produce an instance that is closer to Trace-1 than the other member of the positive class (Trace-2).

It is our central claim that if the two assumptions are true for a given problem, our novel scoring function DTW-D will be better than either ED or DTW. As these are the central assumptions, we will next consider *when* we might expect them to be true.

3.4.2. Observations on our Key Assumptions

In the following sections we consider the implications of our assumptions for the task at hand, and empirically investigate whether these assumptions are warranted.

A. Assumption 1 is Mitigated by Large Amounts of Labeled Data

If we have a large enough set of *labeled* examples, we expect that simple DTW or even ED will work very well. Our noted weakness of semi-supervised learning happens when the nearest instance to a labeled positive exemplar is a negative instance. With more labeled positive instances this becomes less and less likely to happen. To see this, we performed an experiment that generalizes the toy example in Figure 12. We created an unlabeled dataset *U* that contains just one exemplar from Class 3 of Trace, and 200

random walks. We then consider the question of what is the probability that the first object added to the labeled dataset *P* is that sole true positive in *U*, for various sizes of *P* from 1 to 10 (i.e. *P* has 1 to 10 true members from Trace). To smooth out our estimate, we averaged over 1,000 runs. As we can see in Figure 19, this probability does indeed increase as |*P*| gets larger.

This plot suggests that if we had a large enough *P*, then DTW-D would offer only a small advantage over ED, and a barely perceptible improvement over DTW. However when *P* is small, DTW-D is dramatically better than both ED/DTW, supporting our assumption.

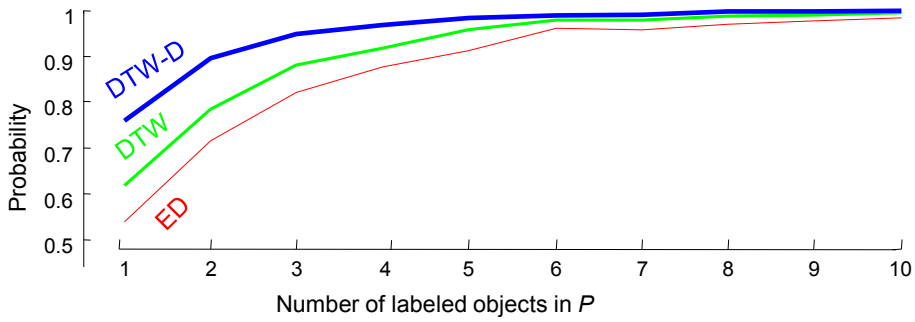


Figure 19: The probability that the first object added to *P* is a true positive as the number of labeled objects increases, for three distance measures

B. Assumption 2 is Compounded by a Large Negative Dataset

In a sense this observation is a direct corollary of the above. If the negative class is random and/or diverse, then the larger the negative class is, the more likely it is that it will produce an instance that just happens to be close to a labeled positive item.

To see this, we again perform an experiment that generalizes our toy example. We created a dataset *P* that contains just one exemplar from Class 3 of Trace. Once again *U* contains a single true positive, but this time we vary the number of random walks from 100 to 1,000. As before we measure the probability that the first object added to *P* is the true positive, averaged over 1,000 runs. Figure 20 shows the results.

In most semi-supervised settings, we expect |*U*| to be many orders of magnitude larger than the |*P*|, thus this assumption is almost always true in real settings.

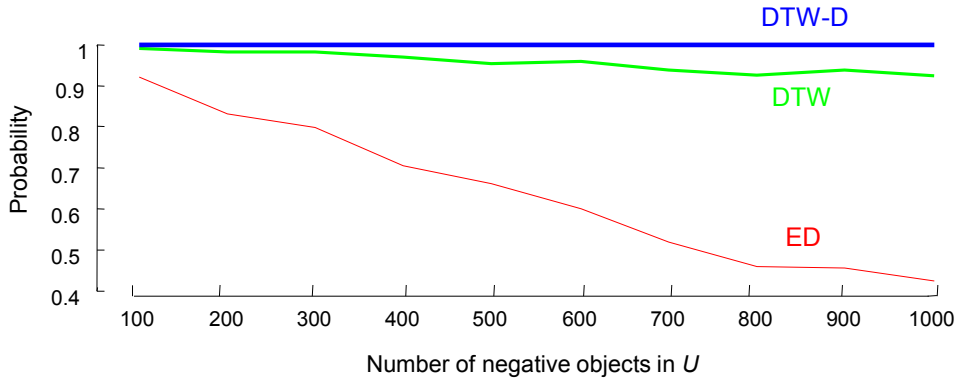


Figure 20: The probability the first object added is a true positive as the number of true negatives in *U* increases

Again this figure strongly supports our assumption. When we have relatively few true negatives, all methods work well. However, as the number of true negatives in *U* increases, ED rapidly deteriorates. In contrast, DTW deteriorates more slowly. Remarkably, however, DTW-D is completely unaffected by a surfeit of true negatives, maintaining perfect accuracy.

C. Assumption 2 is Compounded by Low Complexity Negative Data

Our final observation requires us to define what is meant by the "complexity" of a time series. Our remarks here are inspired by [64], which makes a similar observation, but in a very different context. As noted in [64], while a "complex" time series is hard to define, it is something we can intuitively understand. For our purposes, let us say that a complex time series is one that is not well approximated by few DFT coefficients or by a low degree polynomial.

The problem caused by low complexity data is that it is "close to everything", and as such, the chances that at least one instance from the negative class is closer to an exemplar from *P* than a true positive is much greater if some or all the negative data has low complexity.

To see this we can repeat the experiment shown in Figure 12 after replacing the random walk series by randomly generated third-degree polynomials.

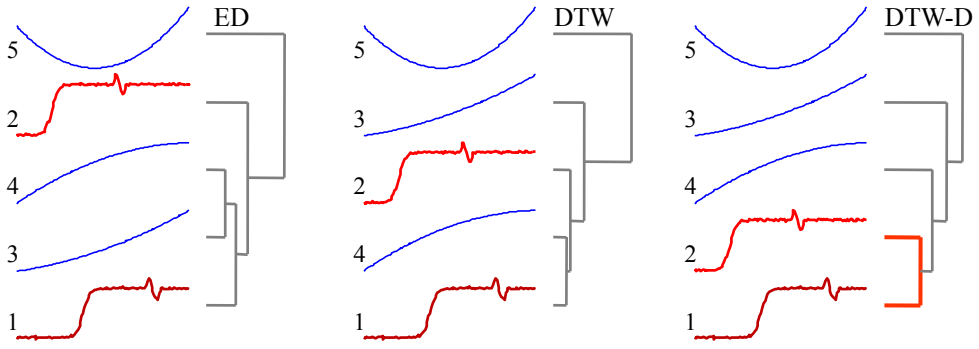


Figure 21: The experiment shown in Figure 12 after replacing the three random walks with three random third-degree polynomials

The results here are visually jarring. It is important to emphasize that this is not the result of an error, contriving of the data, or crippling the ED/DTW in any way. It is

simply the case that low complexity time series have a tendency to have a small distance to all other objects.

Apart from [64], other works have indirectly noted this phenomenon. For example, [78] notes that if we average all subsequences in a long time series, we will get a *constant line*, which is surely the *least* complex time series under any definition. Implicitly, this means that a constant line is the time series with minimal *expected distance* to any randomly chosen time series.

Thus, if the negative class is complex, we should expect the DTW or even ED will work well for semi-supervised learning. To see this, we can repeat the experiment shown in Figure 12/Figure 21 after replacing negative class with pure random vectors. Note that while we may consider random vectors as "noisy", it is incidental to the point that they are *complex*.

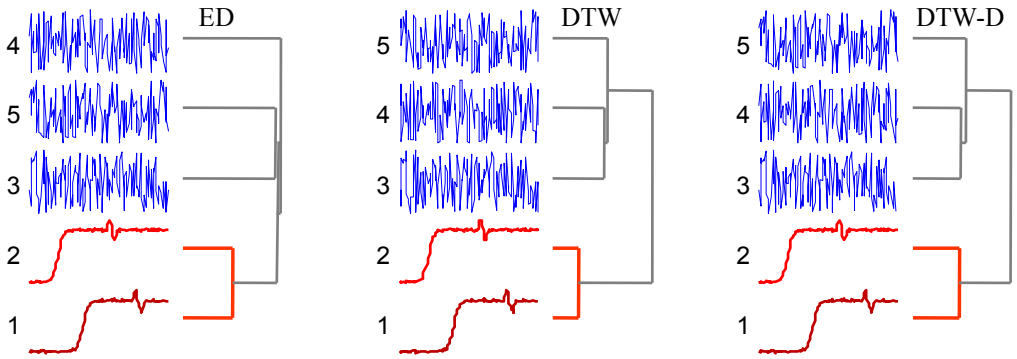


Figure 22: The experiment shown in Figure 12/Figure 21 after replacing the negative class with random vectors

Figure 22 demonstrates that when the unlabeled data are complex, even ED has little trouble in grouping the positive class.

Beyond the visual evidence shown in Figure 21 and Figure 22, we can test our observation with another simple experiment. Once more we perform an experiment that generalizes our toy example. We created a labeled dataset *P* that contains just one exemplar from Class 3 of Trace. This time *U* contains one true positive and 200 random time series that are approximated by *k* non-zero DFT coefficients, with *k* ranging from 5 to 20. Figure 23 shows some examples of time series that are approximated by 5 and 20 non-zero DFT coefficients respectively.

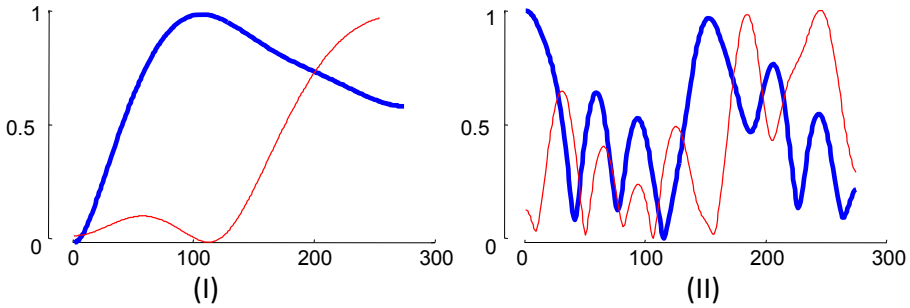


Figure 23: I) Two time series examples that are created using 5 non-zero DFT coefficients. II) Two time series examples that are created using 20 non-zero DFT coefficients. Clearly the latter are more complex

As before we measure the probability that the first object added to *P* is a true positive, averaged over 1,000 runs. Figure 24 shows the results.

Once again this experiment strongly supports our hypothesis. Low complexity items in the negative class make SSL more difficult for all distance measures, but using DTW-D does greatly mitigate the problem.

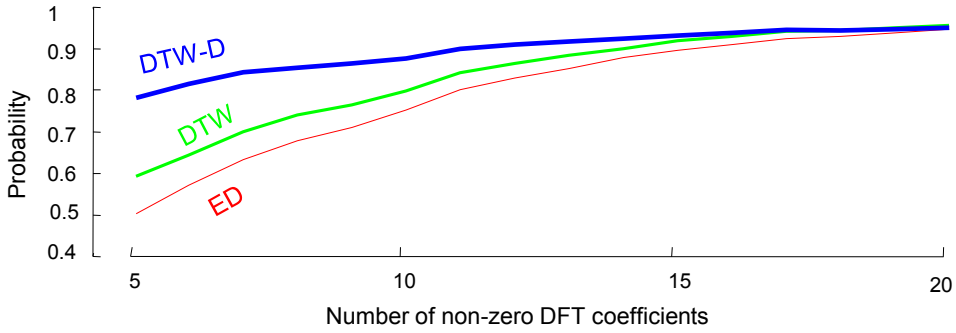


Figure 24: The probability the first object added is true negative as the negative objects increase in complexity

3.4.3. Implications of Observations/Assumptions

The observations and experiments in the previous sections tell us when we should expect the DTW-D method to help. We should *not* expect DTW-D to help with the classic time series classification problems as exemplified by the UCR archive [77]. These datasets typically *do* have a relatively large set of label positive data (at least dozens), and typically do *not* have one class with much lower complexity and/or much higher diversity than the other classes.

In contrast, it is easy to see that all our assumptions mesh perfectly with most SSL assumptions [95][94][90]:

- We *do* have very small (as few as one) positive examples (cf. Section 3.4.2.A.).
- We *do* have relatively large amounts of negative data. For example, when trying to learn the `vacuum-cleaning` concept. (cf. Section 3.6.6), we must expect that most people spend less than 0.01% of their time vacuuming. Likewise, if we monitor audio streams, most sounds we hear are *not* insects (cf. Section 3.6.1) etc.

- We *do* have at least some low complexity negative instances (cf. Section 3.4.2.C.). This is true because the negative class is usually highly variable. For human activity monitoring, there are likely moments when a person is sitting still, producing very low complexity data (cf. Section 3.6.6)

As we shall show in Section 3.6, SSL problems in very diverse domains fit into our assumptions.

3.5 Algorithms

While we believe that our ideas can be applied to essentially any time series SSL learning framework, simply by replacing the ED or DTW distance calculations with DTW-D. However, for concreteness, in this section we will explicitly define the exact SSL algorithm used in our experiments. Note that we took pains to choose a simple SSL algorithm here, because as we noted before, we are not claiming a contribution to SSL per se. Rather, our contribution is a simple but effective fix to a problem that will otherwise plague any attempt at SSL for time series. Our focus in this work is to demonstrate the effectiveness of our ideas, even with a simple SSL algorithm.

Note that we are assuming that whatever "flavor" of SSL is, the underlying classification algorithm will use Nearest Neighbor (NN) [70]. This is because, in spite of two decades of experimentation with neural networks, decision trees, Bayesian methods [96] etc., there is strong empirical evidence that nearest neighbor algorithms are the best approach for time series (see [102], and the references therein and thereof).

3.5.1. DTW-D Algorithm

As hinted in Section 3.4, our proposed distance measure DTW-D is accomplished with a simple equation:

$$\text{DTW-D}(x, y) = \frac{\text{DTW}(x,y)}{\text{ED}(x,y) + \epsilon} \qquad (1)$$

Where ϵ is an extremely small positive quantity used to avoid divide-by-zero error. We reiterate that ϵ is *not* parameter of our system, it is device to enable a terser definition.

As shown in Table 10, the computation of DTW-D can be achieved on two series x and y using one line of matlab code:

Table 10: Our Proposed Distance Measure

```
function      distance = DTW-D(x, y)
    distance = DTW(x,y) / (ED(x,y) + eps);
```

Having explained our technique for choosing *which* objects to add to the training set, in the next section we explain our technique for deciding *when* to stop.

3.5.2. MDL Based Stopping Criterion

Our MDL inspired stopping criterion is based on two key empirical observations:

Observation 1: The warping path vector between the hypothesis and a positive instance is likely to have long runs of zeros.

Observation 2: The description length of the warping path vector between the hypothesis and a positive instance is likely to be smaller than the description length of the Euclidean warping path vector of the same instances, and vice versa.

Consider the following example. We randomly choose one exemplar time series from the Insect Wingbeat sound dataset (cf. Section 3.6.1), to act as our sole positive example, our hypothesis. We then randomly took 50 positive instances and 50 negative instances from that dataset, and for each of the instance, plot the description lengths of their warping path vector with the hypothesis both in DTW and Euclidean space.

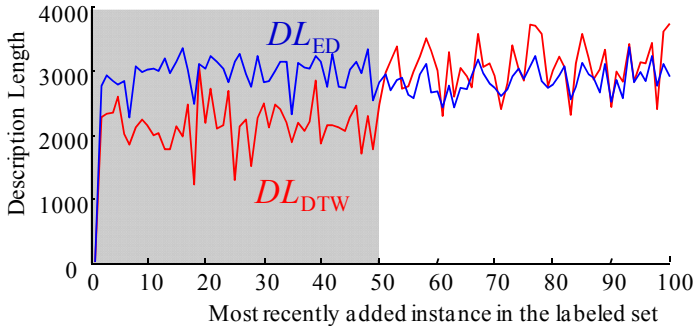


Figure 25: The description length of the warping path vector, DL_{DTW} of positive instances to the hypothesis tends to remain smaller than the description length of the Euclidean warping path vector, DL_{ED} of the same instances (gray background). For negative instances DL_{DTW} tends to be greater than DL_{ED}. While the data is noisy, there is an unmistakable "phase change" at 50 for DL_{DTW}

As shown in Figure 25, DL_{DTW} tends to remain smaller than DL_{ED} for positive instances, and for negative instances the situation reverses. This is an example of our second observation which is confirmed by our first observation. As positive instances are supposed to be more similar to the hypothesis, therefore their warping path vectors *typically* have long runs of zeros, which means DTW has *mostly* found point-to-point exact matches. Consequently the run length encoding of the warping path vectors gives higher compression than the corresponding Euclidean warping path vector. In contrast, because negative instances are less likely to be similar to the hypothesis, therefore, their warping path *typically* do not have as long runs of zeros as the positive instances. This

results in lots of short runs, for which run length encoding cannot give as good compression as the positive instances. Thus DL_{DTW} tends to be greater than DL_{ED} in the negative space.

Note that these results are for a single dataset with a single setting. However, many additional experiments (cf. Section 3.6.8) confirm this general behavior.

Having the two observations above, we are now in a position to describe our MDL inspired stopping criterion. If we increase the number of instances in the labeled set, then the difference of their description length in the DTW space to the description length in the ED space will decrease as long as the set contains instances similar to the hypothesis. However this quantity will start increasing as dissimilar instances to the hypothesis start getting added to the set. Therefore at the minimum value of the difference of DL_{DTW} and DL_{ED} is the point where the self-training should stop.

Consider the Insect dataset we discussed above. At each timestamp, we increase the number of instances in the labeled set and compute the differences of DL_{DTW} and DL_{ED}. From Figure 26 we can see that our algorithm suggests stopping when we are exactly at the boundary of positive and negative instances.

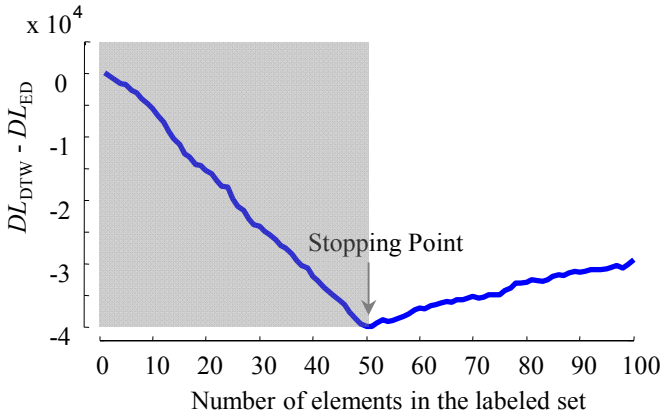


Figure 26: The DTW and ED space description length difference vs. the number of instances in the labeled set plot. Our algorithm suggests stopping when we are at the boundary of the positive and negative instances in the dataset

In Table 11, we formally describe our stopping criterion algorithm.

Table 11: Stopping Criterion Algorithm

	Function *StopPoint* = findStoppingPoint (*data,H*)
	Input: *data,* the dataset with the instances discretized at given cardinality *H,* the hypothesis discretized at given cardinality Output: *StopPoint*, index of the instance with the smallest *DL* cost
1	**function** *StopPoint* = findStoppingPoint (*data*)
2	*DLCost* = [];
3	cost = 0;
4	**for** i = 1:size(*data*)
5	[V_{DTW}, V_{ED}] = findWarpingPathVector(*H*,*data*(i));
6	cost = cost + (DL_{DTW} (V_{DTW}) – DL_{ED} (V_{ED});
7	*DLCost* = [*DLCost* cost];
8	**end**
9	*StopPoint* = findMinimumIndex(*DLCost*);
10	**return**

3.5.3. Training the Classifier

The classifier used is a one-class classifier [97]. The training dataset contains *only* the objects from the positive class. The goal of the classifier is to accurately extract all the positive class objects from the unlabeled dataset [81].

The data used to train the classifier includes a labeled dataset P and an unlabeled dataset U. In the beginning, there is as few as one labeled object in P. As shown in Table 12, the classifier trains itself through the following steps:

Step 1: The classifier is trained on the initial labeled dataset, which contains as few as only one object from the positive class. Note that the labeled dataset is augmented gradually during the training process.

Step 2: For each object in the unlabeled dataset U, we compute its distance to the labeled dataset using DTW-D (Line 10 to Line 13). An object's distance to the labeled dataset is the distance of the object to its nearest neighbor in the labeled dataset.

Step 3: Among all the unlabeled objects, the one we can most confidently classify as positive is the object that is closest to the labeled dataset (Line 3). The object is added into the labeled dataset (Line 4) and removed from the unlabeled dataset (Line 5). With the labeled dataset being adjusted, we return to Step 1 to re-train the classifier. The procedure repeated until some stopping criterion is met.

The intuition behind the algorithm is straightforward. The labeled dataset defines the concept space for the positive objects. The object closest to the labeled dataset is deemed to have the highest probability to belong to the positive class.

Table 12: Time Series Semi-supervised Learning Algorithm

	Function P = Train_Classifier (P, U, distance, N)
	Input: P, the initial training dataset with a single training object U, the unlabeled dataset distance, the distance function N, the number of objects to be moved from U to P Output: a trained classifier (for a NN classifier, it is the learned P)
1	**function** P = Train_Classifier (P, U, distance, N)
2	**for** iterations = 1 : N
3	NNObject = findUnlabeledNN(P, U, distance);
4	P = [P, NNObject]; // update P
5	U = U - NNObject ; // update U
6	**end**
7	**return** P;
8	**end**
9	**function** NNObject = findUnlabeledNN (P, U, distance)
10	Dist = zeros(1, $\|U\|$);
11	**for** i = 1 : $\|U\|$
12	Dist (i) = $\min_{j = 1,\ldots\|P\|}$ distance(U_i,P_j);
13	**end**
14	[~, NN_index] = min(Dist); // closest to P
15	NNObject = U_{NN_index} ;
16	**return** NNObject;
17	**end**

At the first blush, our algorithm to train the classifier seems quite similar to the algorithm used in [82]. However, a more careful introspection would reveal that they are fundamentally different. The classifier used in [82] is a binary classifier, with all the unlabeled objects regarded as training examples from the negative class, whereas our classifier is a one-class classifier with no training examples from the negative class. The

advantage of our classifier is that it makes much more realistic assumptions about how SSL work in practice. As noted elsewhere, the negative class is typically not a single well-defined concept as [82] and others assume, rather it tends to be an *extremely* heterogeneous class. To give an example in a domain we consider, there are very limited ways humans can perform `vacuum cleaning`, but there are an infinite number of possible human activities that are not `vacuum cleaning`.

3.5.4. Evaluating the Classifier

To evaluate the accuracy of all classifiers, we test the classifier using data that is "hidden" during the training stage. The test (holdout) dataset contains some positive class objects and many other objects. The goal of the classifier is to accurately extract all the positive class objects from the test dataset. We use the classic notion of *precision* and *recall* [98] to measure the performance of the classifier. If an instance in the test dataset is top K closest to the labeled dataset, the instance is classified as positive, otherwise it is negative. K is the number of positive objects in the test dataset. Thus we can count the number of true positives out of K classifications.

Note that with this evaluation method, the value of *recall* equals to the value of *precision* (because the number of false negatives is the same as the number of false positives). For brevity, we report only the *precision* here. The computation of *precision* is shown in Equation (2), where N_{positive} denotes the number of true positives among the top K closest instances.

$$precision = \frac{N_{positive}}{K} \quad (2)$$

3.6 Experimental Evaluation

We begin by noting that *all* experiments (including *all* the figures above) are completely reproducible. All experimental code and data (and additional experiments omitted for brevity) are archived in perpetuity at [105].

For all experiments, we divide the data into two mutually exclusive datasets: the learning dataset and the holdout dataset.

- Learning dataset: The *learning dataset* is used in the SSL process to train the classifier. It is divided into the labeled dataset *P* and the unlabeled dataset *U*. The labeled dataset includes a *single* positive example, which is a randomly selected true positive object from the learning dataset. The rest of objects in the learning dataset are regarded as unlabeled objects and are included in *U*.
- Holdout dataset: The *holdout dataset* is used to test the accuracy of the learned classifier. Objects in the holdout dataset are hidden from the SSL process.

The performance of the trained classifier can be sensitive to the initial training (labeled) example. To mitigate this sensitivity, for each experiment, we repeat the training process by each time starting from a different training example. In particular, we allow each positive object in the learning dataset to be used as the initial training example once, and average the accuracy of the classifier over all runs.

To show the changes in the performance of the classifier as the labeled dataset *P* is gradually augmented, we show the average accuracy for each size of *P*. That is, we evaluate the classifier using the holdout dataset each time an unlabeled object is added to *P*. Thus all figures shown below show the *holdout* accuracy.

For each experiment, we compare the performance of three different classifiers, the classifier using ED, the classifier using DTW, and the classifier using DTW-D. All three classifiers are trained using the same SSL algorithm as shown in Table 12. The only difference among them is the distance function used. As we shall show, by simply changing the distance function from ED or DTW to DTW-D, we can improve the performance of SSL algorithms for time series by a significant amount.

In all our experiments, DTW-D learns from a *single* positive example. There is nothing about our technique that requires this. We simply wish to show we can learn under the most hostile assumptions.

We also compare our idea with rival time series SSL approaches [82][92]. In order to be scrupulously fair to the rival approaches, we allow them to "cheat" by starting with more training examples. As we shall show, even given this severe disadvantage, our algorithm still significantly outperforms the rival approaches.

Finally, for clarity we test our two major contributions independently. First showing that we add the *right objects* to the training set (Sections 3.6.1 to 3.6.7), then showing that we stop at the *right time*, when the true positives are exhausted (Section 3.6.8).

3.6.1. Insect Wingbeat Sound Detection

In this experiment, we would like to detect insect wing-beat sounds from unstructured audio streams. The insect used in this experiment is *Culex quinquefasciatus* female. The wingbeat sounds are acquired using the sensors described in [63], and the data stream also contains diverse negative data, including speech/music etc.

For this experiment, we randomly select 1,000 insect sounds and 4,200 non-insect segments. The length of each sound snippet is 0.1 second. All the sound data are first converted into "time series" using DFT [63]. Based on entomological advice, we preserve only the coefficients corresponding to the frequency range between 200 and 2,000, because all other coefficients are unlikely to be the result of insect activity.

We divide the time series dataset into two parts: a learning dataset with 500 insect sounds and 2000 non-insect sounds, and a holdout dataset with 500 insect sounds and 2,200 radio sounds.

The SSL process is repeated 500 times, each time starting with a different training example. For each run, we trained three classifiers, the NN classifier using ED distance, the NN classifier using DTW and the NN classifier using DTW-D. The average performance of the three classifiers over 500 runs for each size of P is shown in Figure 27.

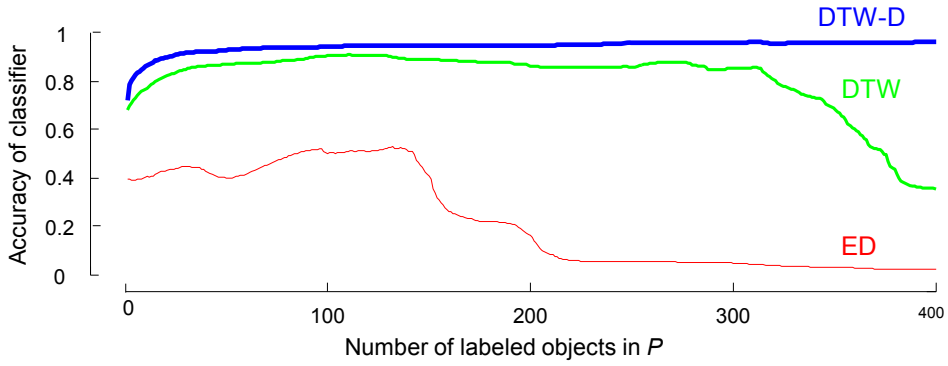


Figure 27: The average accuracy of the three classifiers for different size of *P*, evaluated using the holdout dataset

The results are impressive, given that the default accuracy is just 20%. With DTW-D, we can converge on greater than 95% accuracy. The DTW-D classifier consistently outperforms the other two classifiers across the entire range of values for *P*. Note that, after the size of *P* reaches 150, the accuracy of the ED classifier begins to decrease. As we might expect, the DTW classifier's invariance to warping allows it *both* to start from a higher baseline, *and* keep improving for a longer time. However it too is doomed to eventually add true negatives into *P* and begin a rapid decrease in performance.

3.6.2. Why is DTW-D Better?

While DTW-D is clearly better than its rivals, Figure 27 does not tell us *why*. In particular we may ask if it is because: DTW-D generally selects better labeled objects during the SSL process, *or* because DTW-D selects better top *K* nearest neighbors from the holdout dataset in the classifier's evaluation process (recall that *K* is the number of true positives in the holdout dataset).

To answer this question, we conducted a combinatorial experiment in which we crippled DTW-D independently in each phase (training/evaluating).

For example, to see if DTW-D selects better labeled objects than DTW, we train two NN classifiers, one using DTW-D and one using DTW. We then evaluate both classifiers using the same holdout dataset. In the evaluation process, we use the same distance function (DTW) to find the top *K* nearest neighbors for both classifiers. In this way, we ensure that the *only* difference between the two classifiers is the use of two different distance functions in the training process.

The experiment to see if DTW-D is better at selecting the top *K* nearest neighbors during evaluation process is similar. This time, we train only one classifier, the NN classifier using DTW. In the evaluating process, we use two different distance functions, DTW and DTW-D, to find the top *K* nearest neighbors for this classifier. In this experiment, the labeled dataset learned from the training process is the same. The *only* difference is the use of different distance functions in the evaluation process.

Figure 28 shows the results of the combinatorial experiment for this dataset.

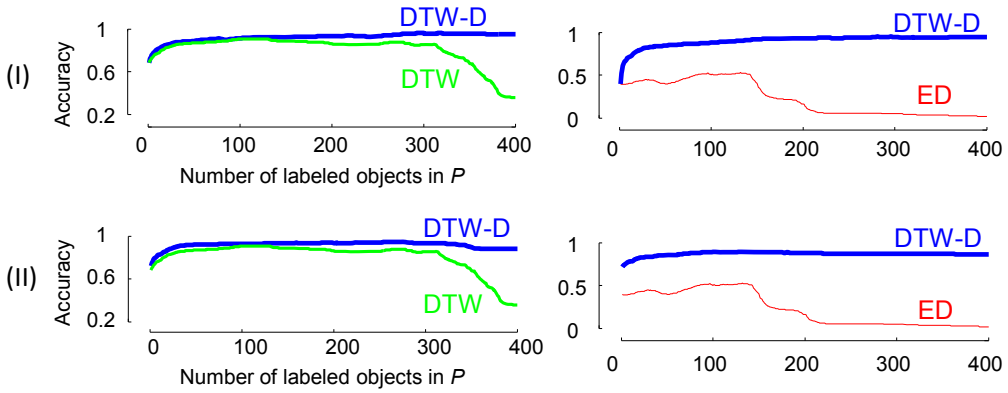


Figure 28: I) Comparison of DTW-D with DTW / DTW-D with ED, to see if DTW-D helps the training process by selecting better exemplars. II) Comparison of DTW-D with DTW / DTW-D with ED, to see if DTW-D helps the evaluating process by selecting better top K nearest neighbors

The results show that DTW-D is superior to ED and DTW in both the training (exemplar *choosing*) and evaluation (the later *classification*) processes. The results shown in Figure 28 are generally true for all the experiments done in this work. For brevity, we omit these results for the following applications, but archive them in [105] for interested readers.

3.6.3. Comparison to Rival Methods

We compare our algorithm with the widely-used rival approaches that are specially designed for time series [82][92]. In the comparison, we favor our rival approaches by offering them with fifty more initial labeled examples. That is, through the entire range of comparison, our rivals always have fifty more labeled objects in their labeled dataset than our method. Recall that our algorithm starts with a *single* labeled example, thus the rival methods have a fiftyfold advantage here. Figure 29 shows the results for this dataset.

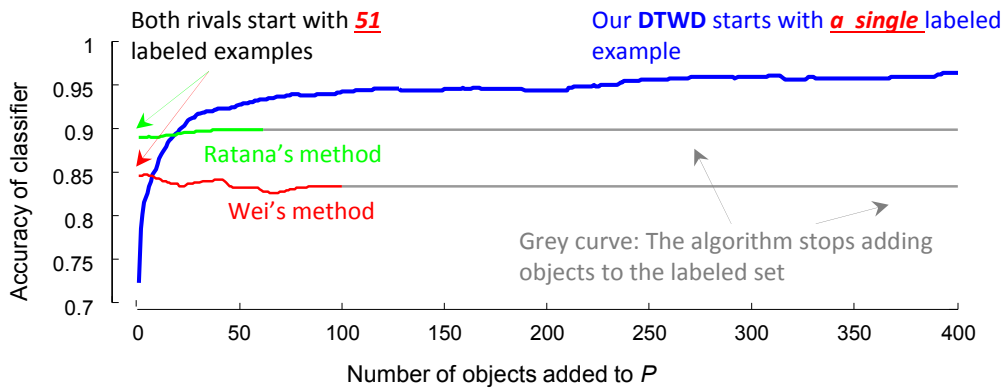


Figure 29: Comparison of our SSL method using DTW-D with two rival methods in the literature. The curves show the average performance of classifiers over 100 runs

As we can see, our method was not as good as the rivals at the beginning. This is hardly surprise given that the rival methods had fifty times as many labeled objects. However, using DTW-D, our method intelligently selects objects from the unlabeled dataset *U* to expand the labeled set *P*. After adding just nine objects, we beat Wei's method. This is very impressive because this happened when we have only ten labeled objects while our rivals have sixty. In other words, we evaluate our classifier with the holdout dataset with only ten labeled objects while our rivals have sixty labeled objects. We are doing better here because while both methods have added nine objects, DTW-D made better choices of what to add. Ratana's method is beaten after adding 21 objects.

In addition, as can be seen from Figure 29, our method continues to do better after we beat the rivals, which implies that whatever stopping criterion is used, we will always do better. In this example, Wei's method stops after adding 103 objects. Ratana's method stops after adding 62 objects. No matter if we stop at 62 or 103 or any other position greater than 21, we are always better than the rivals.

It might be imagined that the two rival methods [82][92] do (eventually) make better decisions about which objects to add, but are crippled by too conservative a stopping criteria. However, to be clear, this is not the case. When the two algorithms terminate, they have labeled *everything* in the learning dataset. Thus, there are no actions (wise or unwise) left for them to perform.

In addition to [82][92], there is only one other semi-supervised approach for time series that we are aware of. In [85], the authors introduce a technique that interleaves exemplar selection with feature selection. We do not compare to this work for the

following reasons: The work assumes that the positive class objects are similar to each other, *and* the negative class objects are similar to each other. For this reason they test on a subset of the UCR archive datasets, with one class acting as *P* and another class acting as *U*. This is in sharp contrast to our more relaxed assumption that *only* positive class objects are similar to each other. We make no such assumptions about the negative class. Our initial attempts to test their algorithm with our assumptions (the authors generously donated their code), yielded very poor results, but in fairness, the authors made no claims about the utility of their ideas for our problem statement/assumptions.

The comparison results shown in Figure 29 are generally true for all experiments done here. For brevity, we omit these results for the following applications, but archive them in [105] for interested readers.

3.6.4. Historical Manuscript Mining

We can apply "time series" SSL to detect particular image patches in historical manuscripts [100]. These manuscripts often are hand colored over years, thus some warping is needed to detect the similarity of the ("drifting") color distributions, as shown in Figure 30.

The task in this application is to detect examples of the Fugger of the Deer heraldic shield from a huge manuscript [104]. Data used in this experiment includes 67 positive Fugger shields and 828 negative patches selected from the same manuscript [104]. Each image patch is converted to a RGB color histogram, which is normalized using sum-to-

one normalization to eliminate the variability of image size. Figure 30.II shows two examples of the converted color histograms.

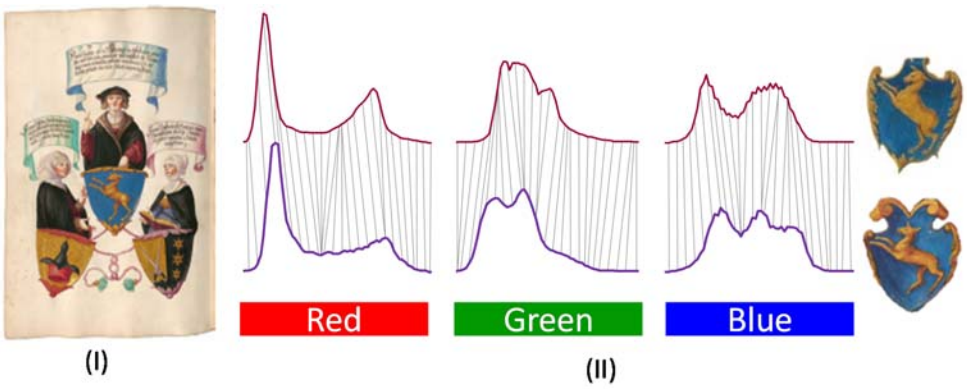


Figure 30: I) A page from a 16th century text [104] shows three heraldic shields including that of Fugger vom Reh (Fugger of the Deer) granted to Andreas Fugger in 1464. II) Two additional examples of the shield from the same text have been converted to RGB color histograms and compared using DTW

Again, we first divide the data into two datasets: a learning dataset with 16 positive objects and 207 negative ones, and a holdout dataset with 51 positive objects and 621 negative ones. We repeat the SSL process 16 times, each time starting from a different training seed. Figure 31 shows the averaged holdout accuracy of the three classifiers for different sizes of *P*.

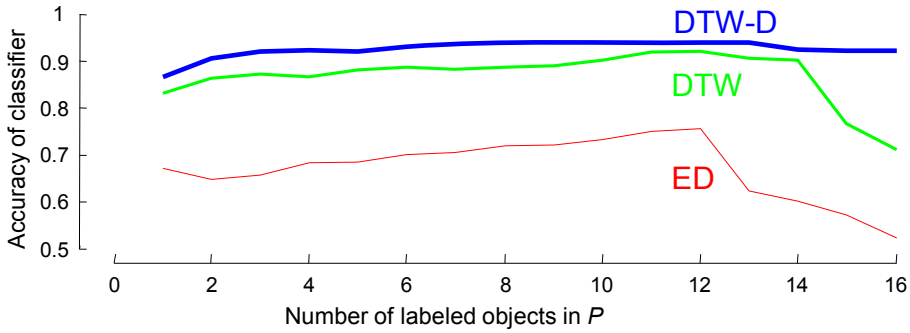


Figure 31: The average accuracy of the three classifiers for different size of *P*, evaluated using the holdout dataset

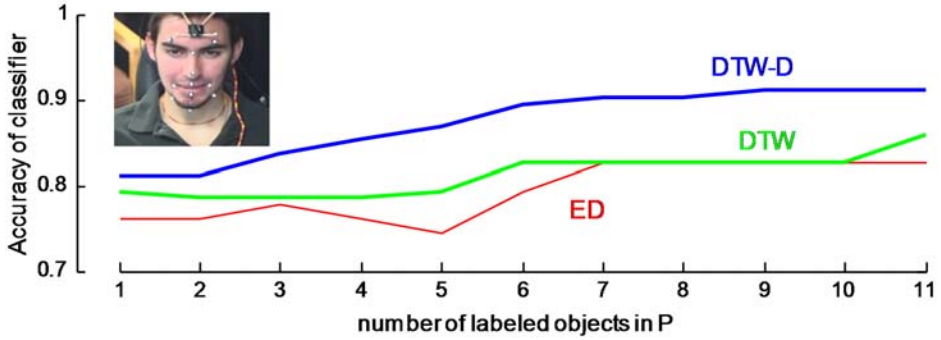
The default holdout accuracy is 51/672, which is about 7.6%. With DTW-D, we can achieve more than 90% accuracy. The performance of the DTW classifier is much better than the ED classifier, which is not surprising since, as we noted before, there is clearly some warping in the color distributions of objects belonging to a same class.

Note that in Figure 31, there is a rapid decrease in the accuracy of the ED classifier when the size of *P* reaches 12. With inspection we find this decrease is caused by the first false positive that is added into *P* at the point (on average). This false positive object in *P* corrupts the concept of the positive class, resulting in more negative objects added to *P,* and thus, decreasing the classifier's accuracy. Although the DTW classifier's invariance to warping enables it to have a higher accuracy as well as to keep improving for a longer time, it too experiences a rapid decrease when the size of *P* reaches 14, where it (on average) mistakenly accepts its first true negative.

3.6.5. Word Recognition from Articulatory Movement

An ElectroMagnetic Articulograph (EMA) is an apparatus used to measure the movement of the tongue and lip during speech. The device is used to study the mechanics of speech, with applications in neurology, phonetics and even dentistry. We consider the data set in [99] which contains data collected from multiple English native speakers producing 25 words. For simplicity, we used data collected from a *single* speaker. For each experiment, we randomly picked a word as the positive class and treated instances of the other 24 words as negative examples.

We divide the data into two datasets, half for training and half for holdout evaluation. We repeat the SSL process 11 times, each time starting from a different training seed, and



report the average classification accuracy. Figure 32 shows the results for an experiment where the word *corpse* is considered as the positive class. As we can see, the DTW-D classifier is better than both DTW and ED across the entire range of different sizes of *P*.

Figure 32: The average accuracy of the three classifiers for different size of *P* for the word recognition dataset, evaluated using the holdout dataset. Inset) A picture of the EMA apparatus used to gather data for this experiment.

3.6.6. Trace Dataset

In this experiment, we consider the Trace dataset from UCR Archive [77]. This dataset may be considered the "Iris dataset" of the time series world, as it has been considered in several hundred papers [70][77].

We randomly chose a class as positive, and treat the remaining three classes as negative. Again, we divide the dataset into two parts, half for training and half for holdout testing. In the training dataset, there are 12 positive objects, and thus, we repeat the SSL process 12 times, each time starting from a different training seed. Figure 33 shows the results for the experiment where the class one is considered as the positive

class. Once again, we can see that the DTW-D classifier is better than both DTW and ED across the entire range of different sizes of P.

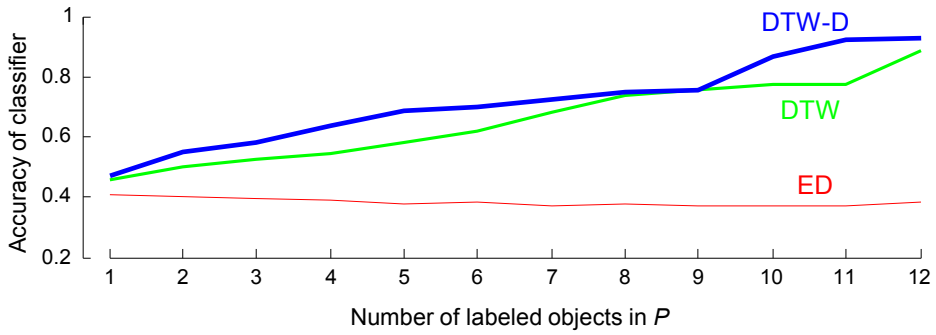


Figure 33: The average accuracy of the three classifiers for different size of P for the word recognition dataset, evaluated using the holdout dataset

3.6.7. Activity Recognition

We finally consider a widely studied benchmark dataset that contains data of 18 different activities, such as *running*, *rope-jumping*, *ironing*, *vacuum-cleaning*, performed by 9 subjects wearing 3 inertial measurement units (IMUs) [88]. We randomly pick one such activity as the positive class and treat the rest as negative.

Figure 34 shows the results for an example experiment where *vacuum cleaning* is considered as the positive activity. We randomly select 400 positive segments and 1,600 negative ones from the dataset, and divide the selected data into two datasets, the learning dataset with 100 positive objects and 400 negative objects, and the holdout dataset with 300 positive objects and 1,200 negatives. The SSL process is repeated 100 times, each time starting from a different training seed. The results show that the DTW-D classifier both starts from a higher baseline and continues to improve over the entire

range of values. In contrast, both ED and DTW start from a lower baseline and eventually get worse.

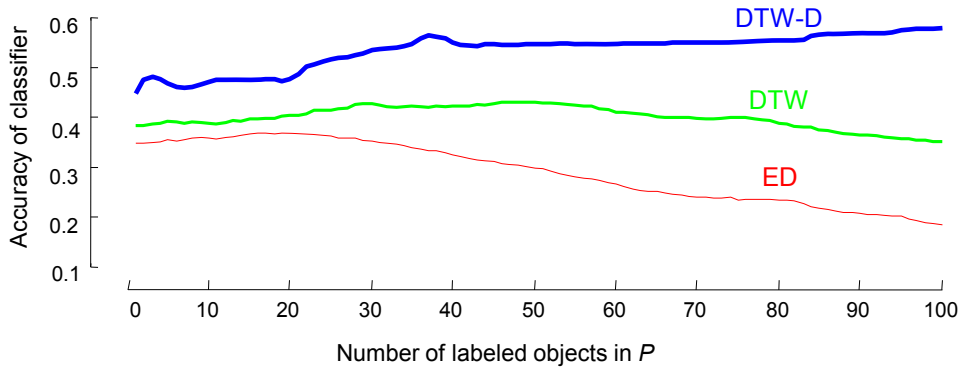


Figure 34: The average accuracy of the three classifiers for different size of *P*, evaluated using the holdout dataset

We remind the reader that as with all experiments in this work, the three lines in this figure are based on identical data splits, identical conditions and identical algorithms. The *only* difference is the distance measure used, thus we can safely attribute *all* improvement observed to DTW-D.

3.6.8. Results of the Stopping Criterion

We performed experiments on the stopping criterion using six datasets. In Table 13 we show the configuration of the datasets:

Table 13: Datasets for the stopping criterion experiment

Dataset	Positive instances	Negative instances
Insect Wingbeat	500	2000
Historical Manuscript	16	207
Articulograph	11	55
Trace	25	75
Activity	100	400
Projectile	38	98

In all our experiments we used a cardinality of 64. For the first four datasets in Table 13, we present the results in Figure 35.

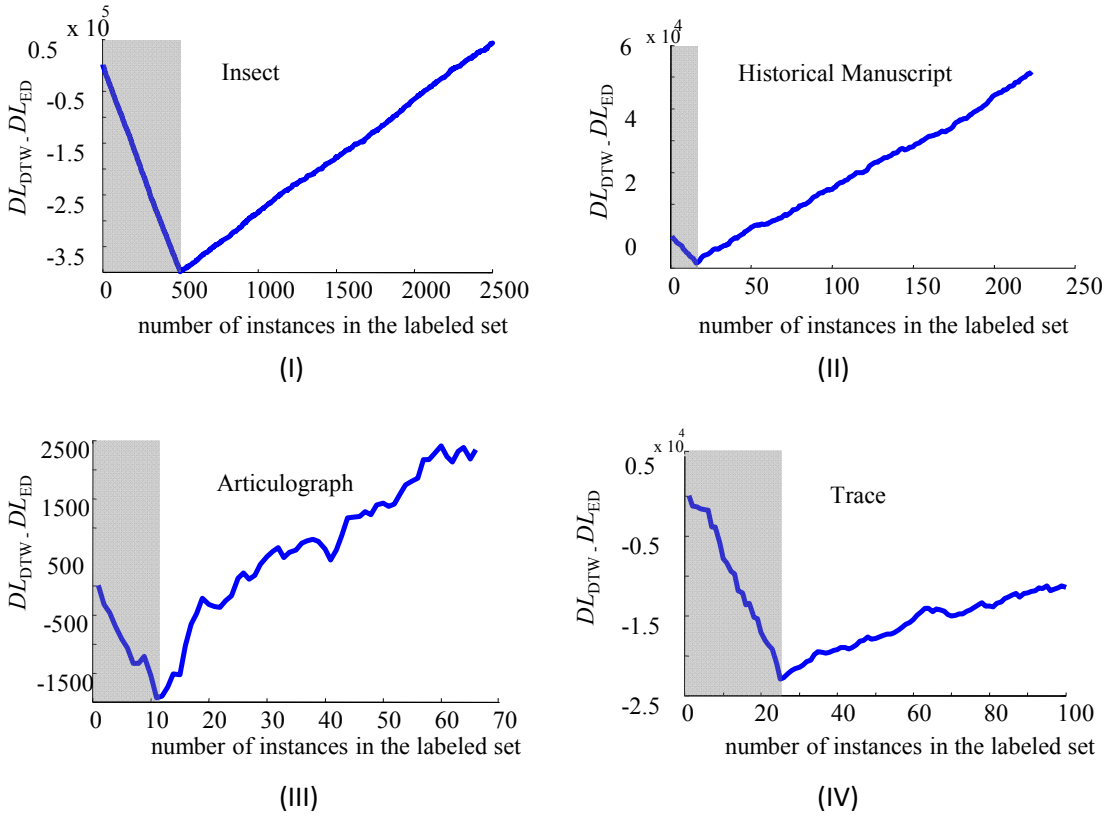


Figure 35: The DTW and ED space description length difference vs. the number of instances in the labeled set plot for the first four datasets in Table 13. For all these datasets, our algorithm suggests the stopping point almost at the boundary of the positive and negative instances (gray background)

As we can see from Figure 35, for the first four datasets in Table 13, our algorithm suggests stopping almost *exactly* at the boundary of the positive and negative instances.

Note however that our algorithm does not work perfectly for the last two datasets in Table 13 given the same experimental setup. In Figure 36.I,III we present the results for these datasets.

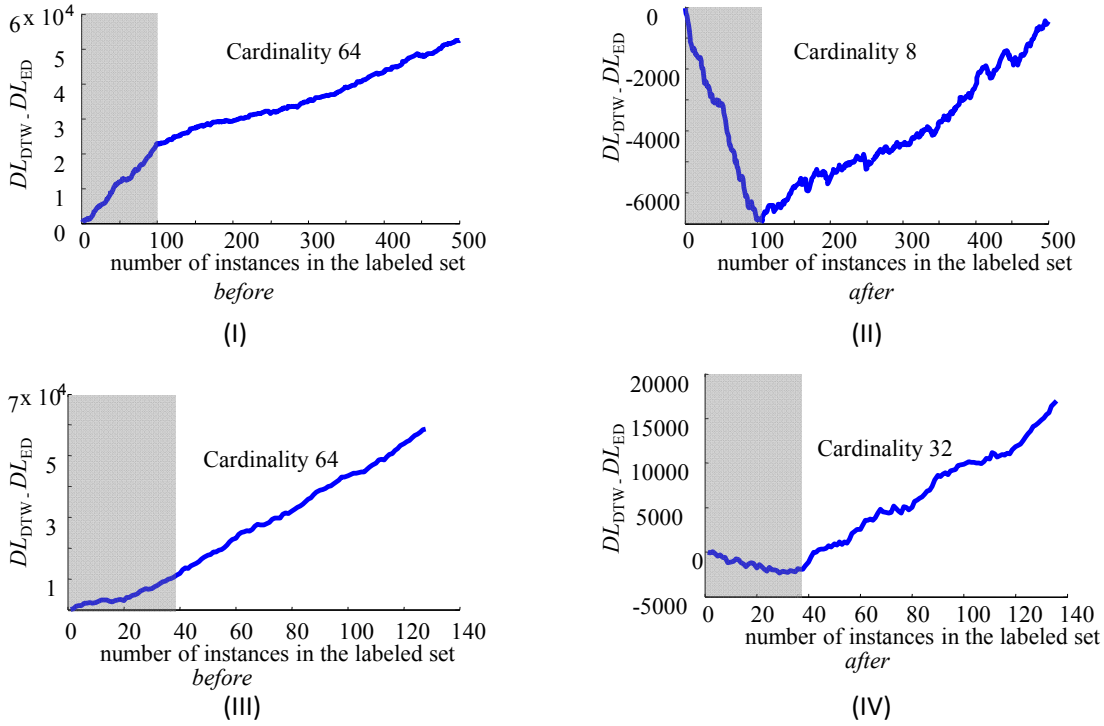


Figure 36: (I and III) The DTW and ED space description length difference vs. the number of instances in the labeled set plot for the Activity and Projectile datasets respectively. (II and IV) The same experiment with reduced a cardinality (8 and 32 respectively) produces better results

From Figure 36 we can see that given the same experimental setup for the last two datasets in Table 13, we do not have desired, *at first* decreasing *and then* the increasing

trend in the description length difference of the warping path vectors. This suggests that our two core observations in Section 3.1 do not hold for these datasets. Because the description length difference curve has an always increasing nature, it means that for the given experimental setup, the run length encoding of the warping path vector costs always more bits than the corresponding Euclidean warping path vector. This means that for the positive instances we do not receive any data compression at all.

To understand why this is the case, we carefully inspected these two datasets, and discovered that both these datasets have many very noisy instances. Therefore when DTW tries aligning these noisy exemplars with the hypothesis while calculating the warping path vector, DTW cannot help much, but rather introduces many symbols in the run length vector with *very* short runs. As a result even if for positive instances we *barely* achieve data compression, and therefore do not find clear separation between positive and negative instances (Figure 36.I and Figure 36.III).

In order to fix this problem we simply reduced the cardinality of the instances from the original 64. This reduction essentially "smoothed" the noisy instances, and as shown in Figure 36.II and Figure 36.IV, at cardinality 8 and 32 we obtained significantly improved results for Activity and Projectile datasets. These results suggest that we need to revise the issue of setting cardinality; one size does not fit all. In [76] the authors consider the issue of setting cardinality in a more general context, again using an MDL-like approach. We believe that this work could solve the problem at hand, but relegate further discussion to future work for brevity.

For the Projectile dataset, we also noticed that the positive class is *polymorphic*. That is to say, there are multiple ways to be in a single class; for example, handwritten number sevens are polymorphic as they may been written as "7" or "7" (the latter being the official style in most of Continental Europe).

Therefore we removed these instances from the positive class resulting in 24 positive instances, and ran our algorithm with cardinality 32. From Figure 37 we can see that the result further improved with only five true positives missed.

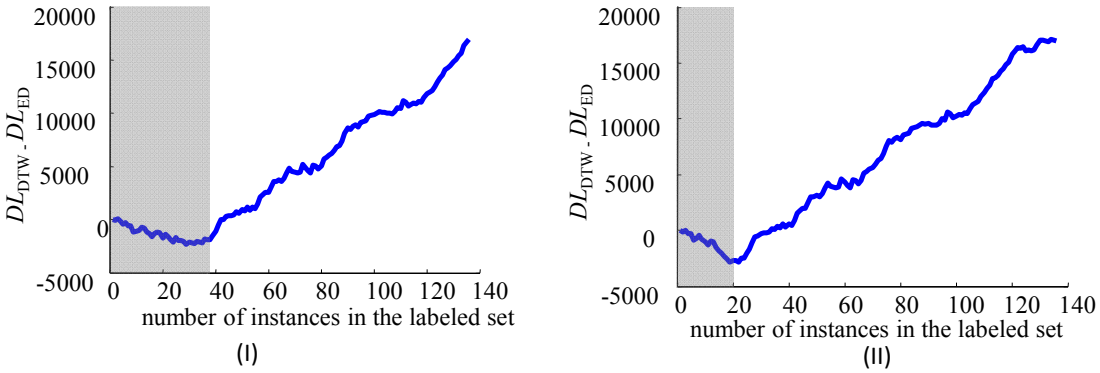


Figure 37: I) The DTW and ED space description length difference vs. the number of instances in the labeled set plot for the Projectile dataset with cardinality 32. II) After removing the polymorphic instances from the positive class, there are 24 positive instances left. The result quality is improved, with a miss of only 5 true positives

Unlike the *wrong cardinality* problem, the *polymorphism* problem is surely insurmountable given our single positive exemplar assumption. For example, in the visual domain, if our single example is one particular cat, from the positive class *Felinae*, and the negative class is non-cat mammals, we might hope to be able to generalize from our single example. However, if our single positive example is one particular cat, and

from the positive class *Mammal*, and the negative class is non mammals, then generalizing from a cat to a whale or a bat would require too great an inductive leap.

3.7 Conclusion and Future Work

In this chapter, we have introduced a simple idea that dramatically improves the quality of SSL in time series domains. We have conducted our experiments such that all improvements observed can be *only* attributed to the use of DTW-D.

Our work has the following advantages: It is completely parameter-free, and thus requires no tuning/tweaking. It allows the use of existing state-of-the-art indexing methods and fast similarity search methods [70]. The time and space overhead are inconsequential, as is the coding effort; requiring only a single line of code to be changed. While we choose the simplest SSL method to demonstrate our ideas, they can trivially be used with almost any SSL algorithm.

We have made all code and data freely available to allow others to confirm and extend our work.

Future work includes considering other avenues where DTW-D may be useful, such as clustering and segmentation.

Chapter 4: A General Framework for Never-Ending Learning from Time Series Streams

As shown in Chapter 2, it is possible to accurately classify the species of flying insects by transforming the faint audio produced by their flight into a periodogram and doing nearest neighbor time series classification on this representation. This allows us to classify known species, such as species we have raised in our lab, to obtain training data. However, in many insect monitoring settings we are almost guaranteed to encounter some unexpected or invasive species [113]. The question is, can we detect these invasive species and classify them?

Note that different from traditional classification problems where we initially have all the classes defined and all unknown objects belong to one of the pre-defined classes, in this case, we need to discover new classes from the data stream as we are doing the monitoring.

For this purpose, we propose a never-ending learning framework in which an agent *observes data streams forever*, and continuously attempts to *extract new concepts* from the stream while doing the classification. Since this project is a jointed work with other researchers (joint first authors), the detailed description of this work can be found in [111]. In this dissertation, for brevity, we omit the detailed algorithms, but only introduce the overview of the proposed framework and focus on showing how the framework can be applied to detect invasive species of insects in the scenario as described above.

4.1 Introduction

Virtually all work on time series classification assumes a one-time training session in which multiple labeled examples of all the concepts to be learned are provided. This assumption is sometimes valid, for example, when learning a set of gestures to control a game or novel HCI interface [115]. However, in many medical and scientific applications, we initially may have only the vaguest understanding of what concepts need to be learned. Given this observation, and inspired by the Never-Ending Language Learning (NELL) research project at CMU [109], we propose a time series learning framework in which we observe streams forever, and we continuously attempt to learn new (or drifting) concepts.

Our ideas are best illustrated with a simple visual example. In Figure 38, we show a time series produced by a light sensor at Soda Hall in Berkeley. While the sensor will produce data forever, we can only keep a fixed amount of data in a buffer. Here, the daily periodicity is obvious, and a more careful inspection reveals two *very* similar patterns, annotated A and B.

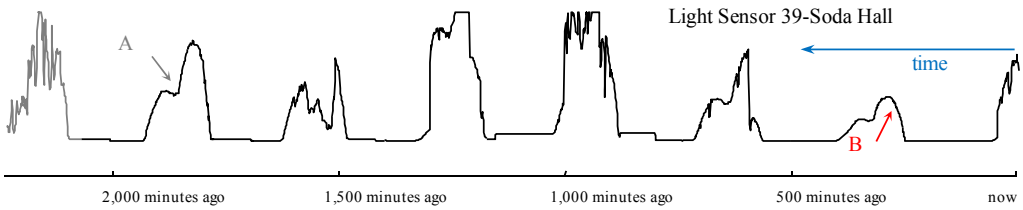


Figure 38: The light sensors at Soda Hall produce a never-ending time series, of which we can cache only a small subset in the main memory

As we can see in Figure 39. I and Figure 39. II, these patterns are even more similar after we z-normalize them [110]. Suppose that the appearance of these two similar patterns (or "motif") causes an agent to query a teacher as to their *meaning*.

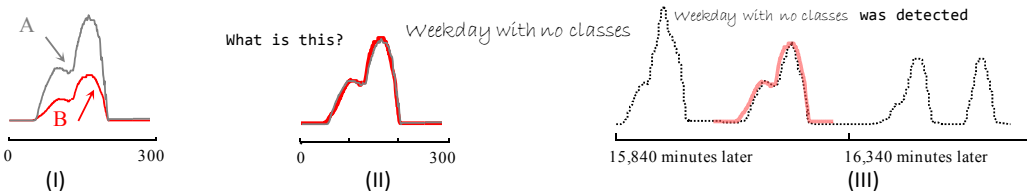


Figure 39: I) A "motif" of two patterns annotated in Figure 38 aligned to highlight their similarity. II) We imagine asking a teacher for a label for the pattern. III) This allows us to detect and classify a new occurrence eleven days later

This query could be implemented in a number of ways; moreover, the teacher need not necessarily be human. Let us assume in this example that an email is sent to the building supervisor with a picture of the patterns and any other useful metadata. If the teacher is willing to provide a label, in this case *Weekday with no classes*, we have learned a concept for this time series, and we can monitor for future occurrences of it.

An important generalization of the above is that the time series may only be a proxy for another much higher-dimensional streaming data source, such as video or audio. For example, suppose the classrooms are equipped with surveillance cameras, and we had conducted our monitoring at a finer temporal resolution, say seconds. We could imagine that our algorithm might notice a novel pattern of short-lived but dramatic spikes in light intensity. In this case we could send the teacher not the time series data, but some short video clips that bracket the events. The teacher might label the pattern *Camera use with flash*. This idea, that the time series is only a (more tractable) proxy for the real stream of

interest, greatly expands the generality of our ideas, as time series has been shown to be a useful proxy of audio, video, text, networks, and a host of other types of data [108].

This example elucidates our aims, but suggests a wealth of questions. How can we detect repeated patterns, especially when the data arrives at a *much* faster rate, and the probability of two patterns from a rare concept appearing close together is very small? Assuming the teacher is a finite or expensive resource, how can we optimize the set of questions we might ask of it/him/her, and how do we act on this feedback?

In the next section, we will introduce the overview of the framework, then we show how the framework can be used to detect invasive insects in section 4.3. Finally, we will offer a conclusion in section 4.4.

4.2 Overview of System Architecture

We begin by stating our assumptions:

- We assume we have a never-ending[7] data stream **S.**

S could be an audio stream, a video stream, a text document stream, multi-dimensional time series telemetry, etc. Moreover, **S** could be a combination of any of the above. For example, all broadcast TV in the USA has simultaneous video, audio, and text.

- Given S, we assume we can record or create a real-time *proxy* stream *P* that is "parallel" to **S**.

[7]For our purposes, a "never-ending" stream may only last for days or hours. The salient point is the contrast with the *batch* learning algorithms that the vast majority of time series papers consider [110].

P is simply a single time series that is a low-dimensional (and therefore easy to analyze in real time) proxy for the higher dimensional/higher arrival rate stream **S** that we are interested in. In some situations, *P* may be a *companion* to **S**. For example, in [107], which manually attempts some of the goals of this work, **S** is a night-vision camera recording sleeping postures and *P* is a time series stream from a sensor worn on the wrist of the sleeper. In other cases, *P* could be a *transform* or low-dimensional projection of **S**. In one example we consider in our experimental section, **S** is a stereo audio stream recorded at 44,100Hz, and *P* is a single-channel 100Hz Mel-frequency cepstral coefficient (MFCC) transformation of it. Note that our framework includes the possibility of the special case where **S** = *P*, as in Figure 38.

- We assume we have access to a teacher (or *Oracle* [114]), and the cost of querying the teacher is both fixed and known in advance.

The space of possible teachers is large. The teacher may be *strong*, giving only correct labels to examples, or *weak*, giving a set of probabilities for the labels. The teacher may be *synchronous*, providing labels on demand, or *asynchronous*, providing labels after a significant delay, or at fixed intervals.

Given these definitions our problem can be defined as:

Problem Definition: Given a stream *P*, which may be a proxy for a higher dimensional stream **S** that is recorded in parallel, and given access to a teacher, which may be a human or an algorithm, which can provide class labels for subsections of *P* (possibly by exploiting information available in the corresponding subsections of **S**),

extract concepts from *P* and label them. The success at this task is measured by *coverage*, the number of concepts learned, and the average *accuracy* of each learned concept.

Given the sparseness of our assumptions and especially the generality of our teaching model, we wish to produce a very general framework in order to address a wealth of domains. However, many of these domains come with unique domain-specific requirements. Thus, we have created the framework outlined in Figure 40, which attempts to divorce the domain-dependent and domain-independent elements.

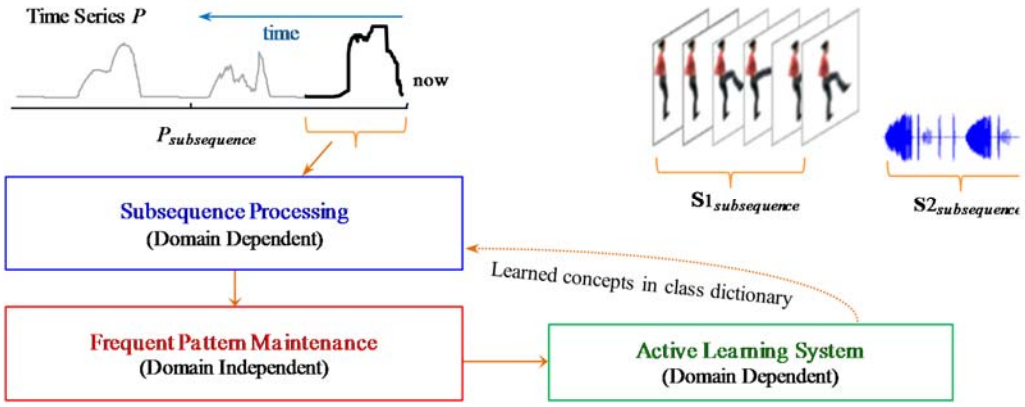


Figure 40 : An overview of our system architecture. The time series *P* which is being processed may actually be a proxy for a more complex data source such as audio or video (*top right*)

Recall that *P* itself may be the signal of interest, or it may just be a proxy for a higher-dimensional stream S, such as a video or audio stream, as shown in Figure 40 top-right.

Our framework is further explained at a high level in Table 14. We begin in Line 1 by initializing the class dictionary, in most cases just to empty. We then initialize a dendrogram of size *w*. This dendrogram is initialized with random data, but as we shall see, these random data are quickly replaced with subsequences from *P* as the algorithm runs.

After these initialization steps, we enter an infinite loop in which we repeatedly extract the next available subsequence from the time series stream *P* (Line 4), and pass it to a module for *subsequence processing*. In this unit, domain-dependent normalization may take place (Line 5), and we will attempt to classify the subsequence using the class dictionary. If the subsequence is not classified and is regarded as valid, then it is passed to the frequent pattern maintenance algorithm in Line 6, which attempts to maintain an approximate history of all data seen thus far. If the new subsequence is similar to previously seen data, this module may signal this by returning a new 'top' motif. In Line 7, the active learning module decides if the current top motif warrants seeking a label. If the motif is labeled by a teacher, the current dictionary is updated to include this now *known* pattern.

Table 14: The Never-Ending Learning Algorithm

Algorithm: Never_Ending_Learning(**S**,*P*,*w*)

```
1   dict ← initialize_class_dictionary
2   global dendro = create_random_dendrogram_of_size(w)
3   For ever
4         sub  ← get_subsequence_from_P(S,P)
5         sub  ← subsequence_processing(sub, dict)
6         top  ← frequent_pattern_maintenance(sub)
7         dict ← active_learning_system(top, dict)
8   End
```

4.3 Experiment: Invasive Species of Flying Insects

The classifier shown in Chapter 2 allows us to classify *known* species, such as species we have raised in our lab, to obtain training data. However, in many insect monitoring settings we are almost guaranteed to encounter some unexpected or invasive species [113]; can we use our framework to detect and classify them?

At first blush, this does not seem possible. The **S** data source is a high quality audio source, and while entomologists could act as our teachers, at best they could recognize the sound at the family level, i.e., some kind of *Apoidea* (bee). We could hardly expect them to recognize which of the 21,000 or so species of bee they heard.

We had considered augmenting **S** with HD video, and sending the teacher short video clips of the novel insects. However, many medically and agriculturally important insects are tiny; for example, some species of *Trichogramma* (parasitic wasps) are just 0.2 mm.

Our solution is to exploit the fact that some insect traps can *physically* capture the flying insects themselves and record their time of capture [112]. Thus, the **S** data source consists of audio snippets of the insects as they flew into the trap *and* the physical bodies of insects. Naturally, this causes a delay in the teaching phase, as we cannot digitally transmit **S** to the teacher but must wait until she comes to physically inspect the trap once a day.

Using insects raised from larvae in our lab, we learned two concepts: *Culexstigmatosoma* male (C*stig* ♂) and female (C*stig* ♀). These concepts are just the

periodograms shown in Figure 41 with the thresholds that maximized cross-validated accuracy.

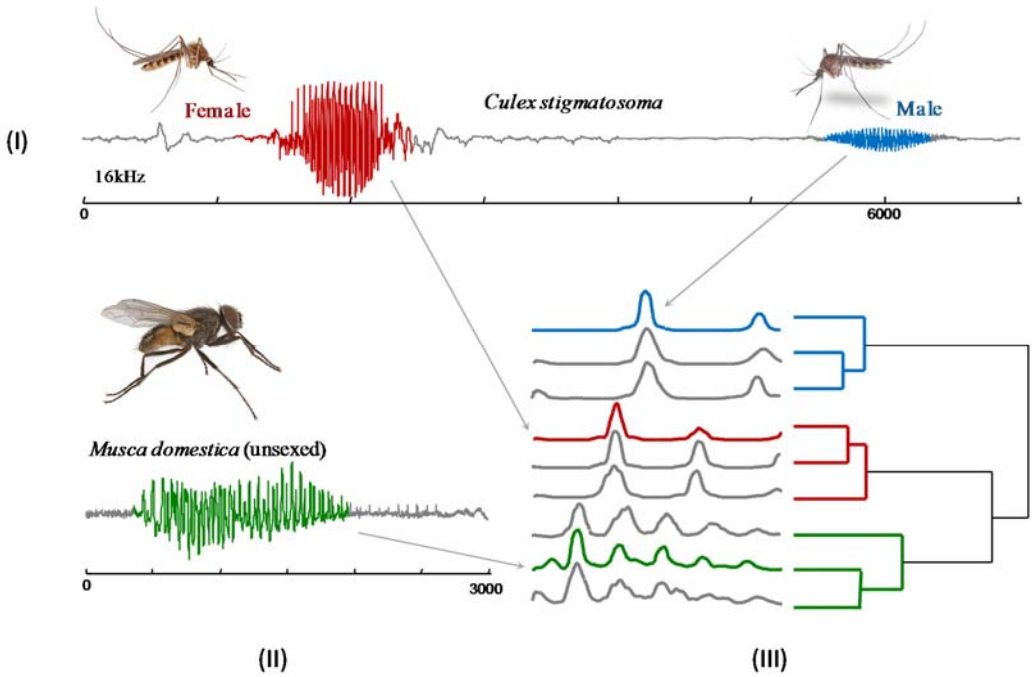


Figure 41: I) An audio snippet of a female Cx. stigmatosoma pursued by a male. II) An audio snippet of a common house fly. II) If we convert these sound snippets into periodograms we can cluster and classify the insects

With the two concepts now hard coded into our dictionary, we performed the following experiments. On day one we released 500 *Cx. Stigmatosoma* of each sex, together with two members of an *invasive species*. If we could not detect the invasive species, we increased their number for the next day, and tried again until we did detect them. After we detected the invasive species, the next day we released 500 of them with 500 *Cx. Stigmatosoma* of each sex and measured the precision/recall of detection for all three classes. We repeated the whole procedure for three different species to act as our invasive species. Table 15 shows the results.

Table 15: Our Algorithm's Ability to Detect and Then Classify Invasive Insects

Number of insects before detection		Precision / Recall		
invasive species name	triggered	*invasive species*	C*stig*♂	C*stig*♀
Aedes aegypti ♀	3	**0.91 / 0.86**	0.88/0.94	0.96/0.92
Culex tarsalis ♂	3	**0.57 / 0.66**	0.58/0.78	1.00/0.95
Musca domestica ♂ and ♀	7	**0.98 / 0.73**	0.99/0.95	0.96/0.94

Recall that the results for C*stig* ♂ and C*stig* ♀ test only the representational power of the dictionary model, as we learned these concepts offline. However, the results for the three invasive species *do* reflect our ability to learn rare concepts (just 3 to 7 sub-second occurrences in 24 hours), and having learned these concepts, we tested our ability to use the dictionary to accurately detect further instances. The only invasive species for which we report less than 0.9 precision is *Cx. tarsalis* ♂, which is a sister species of the *Cx. stigmatosoma*, and thus it is not surprising that our precision falls to a (still respectable) 0.57.

4.4 Conclusion and Future Work

In this chapter, we have introduced the first never-ending framework for real-valued time series streams. We have shown that it can be applied to detect invasive species of insects and learns new concepts fast.

Our framework has some limitations. It needs some human intervention to set initial parameters and it requires (or at least, greatly benefits from) some domain dependent

settings. In future work, we hope to remove the few assumptions/parameters we have and apply our ideas to year-plus length streams. In particular, we plan to investigate extensions to situations where the patterns may be *very* rare, perhaps making up less than one-billionth of the overall data. At the very least, it is clear that our frequent pattern mining algorithms would be greatly challenged by such domains. In addition we will consider the issue of *concept drift*. For example, recall our case study with invasive species of flying insects. It is known that the wingbeat "signatures" of an individual insect change with its age. Moreover, the wingbeat frequency changes with temperature, which itself changes with the seasons. Addressing such challenges will allow our system to be more broadly applied.

We have made all our code and data freely available at [116] and hope to see our work built upon and applied to an even richer set of domains.

Chapter 5: Conclusions

An inexpensive, noninvasive system that could to accurately classify flying insects would have important implications for entomological research, and allow for the development of many useful applications in vector control for both medical and agricultural entomology. Given this, the last sixty years have seen many research efforts devoted to this task. To date, however, none of this research has had a lasting impact.

In this dissertation, we propose a sensor/classification framework for flying insect classification. We show that pseudo-acoustic optical sensors can produce vastly superior data and we propose a Bayesian classification approach that allows to efficiently learn classification models that are very robust to over-fitting, and a general classification framework that allows to easily incorporate arbitrary number of features. We show that additional features, both intrinsic and extrinsic to the insect's flight behavior, can be exploited to improve insect classification. We demonstrate our findings with large scale experiments that dwarf all previous works combined, as measured by the number of insects and the number of species considered. The accuracies achievable by the system are good enough to allow the development of commercial products and to be a useful tool for entomological research.

In addition, when dealing with new insect species, it may be necessary to bootstrap the modeling of the species by using just a handful of annotated examples to find more (unannotated) examples in the archives, a process known as semi-supervised learning. In this work, we also introduce a simple idea that dramatically improves the quality of SSL

in time series domains. We investigate why semi-supervised learning algorithms typically fail for time series, and we introduce a simple but effective fix. The proposed fix does not require much coding effort, but only a single line of code needs to be changed. It is parameter free and can trivially be used with almost any SSL algorithm.

Moreover, we consider the scenario in many insect monitoring settings where we initially do not have a complete list of all the species of insects and have to discover new species from the data stream while doing classification. We propose a never-ending learning framework for this task. In the framework, an agent observes data streams forever, and continuously attempts to extract new concepts from the stream while doing the classification. We apply the framework to invasive insect species detection and it detects the invasive species both fast and accurately.

Finally, I would like to share some useful research philosophy that we observe during this dissertation process:

- For any bad idea, there is a dataset that makes it look good. Therefore, in all our work, we strive to test on *many* diverse real world datasets.
- Reproducibility is our religion. To encourage the adoption and extension of our ideas, we are making all code and data *freely available* at our webpage [117].

Bibliography

Chapter 1

[1] P. Chazal, M. O'Dwyer, R.B. Reilly, "Automatic classification of heartbeats using ECG morphology and heartbeat interval features," *IEEE Trans Biomed Eng*, 51: 1196–1206, 2004

[2] E. J. Keogh, Q. Zhu, B. Hu, Y. Hao, X. Xi, L. Wei, C. A. Ratanamahatana, C. A. *The UCR Time Series Classification/Clustering Homepage*, 2011

[3] SI. Hay, et al. "Forecasting, warning, and detection of malaria epidemics: A case study." *Lancet* 361: 1705–1706, 2003

[4] Y. Hao, "Supervised and Unsupervised Discovery of Structures in Large Data Archives," *PhD disseration. University of California, Riverside*, 2014.

[5] J. Hemingway, H. Ranson. "Insecticide resistance in insect vectors of human disease." *Annu Rev Entomol*. 45:371–39, 2000

[6] M. Raptis, K. Wnuk, S. Soatto, "Flexible Dictionaries for Action Recognition," *MLVMA/ECCV*, 2008

[7] P. Sykacek, S.J. Roberts, "Bayesian time series classification," *NIPS* : 937-944, 2001

[8] S. Zhai, P.O. Kristensson, C. Appert, T.H. Anderson, X. Cao. "Foundational Issues in Touch-Surface Stroke Gesture Design — An Integrative Review." *Foundations and Trends in Human-Computer Interaction.* 5, 2, 97-205, 2012

Chapter 2

[9] M. Banko, E. Brill, "Mitigating the paucity-of-data problem: Exploring the effect of training corpus size on classifier performance for natural language processing." *Proceedings of the first international conference on Human language technology research* (pp. 1-5). *Association for Computational Linguistics*, 2001

[10] GE. Batista, E. Keogh, A. Mafra-Neto, E. Rowton, "SIGKDD demo: sensors and software to allow computational entomology, an emerging application of data mining." *In Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining,* pp. 761-764, 2011

[11] P. Belton, R. Costello, "Flight sounds of the females of some mosquitoes of Western Canada." *Entomologia experimentalis et applicata*, 26(1), 105-114, 1979

[12] M. Benedict, A. Robinson "The first releases of transgenic mosquitoes: an argument for the sterile insect technique." *TRENDS in Parasitology*, 19(8): 349-355. Accessed March 8, 2012.

[13] S. Boll, "Suppression of acoustic noise in speech using spectral subtraction. Acoustics," *Speech and Signal Processing, IEEE Transactions on*, 27(2), 113-120, 1979

[14] RN. Bracewell, RN. Bracewell, "The Fourier transform and its applications." *New York: McGraw-Hill*. Vol. 31999, 1986

[15] JL. Capinera, Encyclopedia of entomology. Springer. Epsky ND, Morrill WL, Mankin R, "Traps for capturing insects." *In Encyclopedia of Entomology*, pp. 2319-2329. Springer Netherlands, 2005

[16] Y. Chen, Supporting Materials https://sites.google.com/site/insectclassification/, 2013

[17] Y. Chen, A, Why, G. Batista, A. Mafra-Neto, E. Keogh E, *supporting technique report http://arxiv.org/abs/1403.2654, 2014*

[18] Y. Chen, B. Hu, E. Keogh, G. Batista, "DTW-D: time series semi-supervised learning from a single example." *In Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 383-391, 2013

[19] MF. Cooperband, A. Hartness, JP. Lelito, AA. Cosse, "Landing surface color preferences of Spathius agrili (Hymenoptera: Braconidae), a parasitoid of emerald ash borer, Agrilus planipennis (Coleoptera: Buprestidae)." *Journal of Insect Behavior*. 26(5):721-729, 2013

[20] MA. Deakin, "Formulae for insect wingbeat frequency." *Journal of Insect Science*,10(96):1, 2010

[21] L. Devroye, "A probabilistic theory of pattern recognition." *Springer* Vol. 31, 1996

[22] C. Elkan, "The foundations of cost-sensitive learning." *In international joint conference on artificial intelligence*, vol. 17, No. 1, pp. 973-978. LAWRENCE ERLBAUM ASSOCIATES LTD, 2001

[23] Y. Ephraim, D. Malah, "Speech enhancement using a minimum-mean square error short-time spectral amplitude estimator." *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 32(6), 1109-1121, 1984

[24] TB. Frick, DW. Tallamy, "Density and diversity of non-target insects killed by suburban electric insect traps." *Entomological News*, 107, 77-82, 1996

[25] K Fukunaga, "Introduction to statistical pattern recognition." *Access Online via Elsevier*, 1990

[26] RP. Grimaldi, "Discrete and Combinatoral Mathematics: An Applied Introduction" 2nd Ed. *Addison-Wesley Longman Publishing Co., Inc*, 1989

[27] A. Halevy, P. Norvig, F. Pereira, "The Unreasonable Effectiveness of Data," *IEEE Intelligent Systems*, v.24 n.2, p.8-12, 2009

[28] Y. Hao, B. Campana and E. Keogh, "Monitoring and Mining Animal Sounds in Visual Space." *Journal of Insect Behavior:* 1-28, 2012

[29] MC. Kahn, W. Celestin, W. Offenhauser, "Recording of sounds produced by certain disease-carrying mosquitoes." *Science* 101: 335–336, 1945

[30] MC. Kahn, W. Offenhauser, "The identification of certain West African mosquitos by sound." *Amer. J. trop. IVied.* 29: 827—836, 1949

[31] E. Keogh, M. Pazzani, "Learning augmented Bayesian classifiers: A comparison of distribution-based and classification-based approaches." *In Proceedings of the seventh international workshop on artificial intelligence and statistics.* pp. 225-230, 1999

[32] R. Kohavi, "A study of cross-validation and bootstrap for accuracy estimation and model selection." In *IJCAI* , Vol. 14, No. 2, pp. 1137-1145, 1995

[33] Z. Li, Z. Zhou, Z. Shen, Q. Yao, "Automated identification of mosquito (diptera: Culicidae) wingbeat waveform by artificial neural network." *Artificial Intelligence Applications and Innovations*, 187/2009: 483–489, 2009

[34] YP. Mack, M. Rosenblatt, "Multivariate k-nearest neighbor density estimates." *Journal of Multivariate Analysis*, 9(1), 1-15, 1979

[35] RW. Mankin, R. Machan, R. Jones, "Field testing of a prototype acoustic device for detection of Mediterranean fruit flies flying into a trap." *Proc. 7th Int. Symp. Fruit Flies of Economic Importance*, pp. 10-15, 2006

[36] P. Mermelstein, "Distance measures for speech recognition, psychological and instrumental, *Pattern Recognition and Artificial Intelligence*," 116, 374-388, 1976

[37] LS. Mian, MS. Mulla, H. Axelrod, JD. Chaney, MS. Dhillon, "Studies on the bioecological aspects of adult mosquitoes in the Prado Basin of southern California." *Journal of the American Mosquito Control Association*, 6(1), 64-71, 1990

[38] A. Moore, "Artificial neural network trained to identify mosquitoes in flight." *Journal of insect behavior*, 4(3), 391-396, 1991

[39] A. Moore, RH. Miller, "Automated identification of optically sensed aphid (Homoptera: Aphidae) wingbeat waveforms." *Ann. Entomol. Soc. Am*. 95: 1–8, 2002

[40] A. Moore, JR. Miller, BE. Tabashnik, SH. Gage, "Automated identification of flying insects by analysis of wingbeat frequencies." *Journal of Economic Entomology.* 79: 1703-1706, 1986

[41] R. Novak, "The asian tiger mosquito, Aedes albopictus." *Wing Beats*, Vol. 3(3):5, 1992

[42] PA. Papathanos, HC . Bossin, MQ. Benedict, F. Catteruccia, CA. Malcolm, L. Alphey, A. Crisanti, "Sex separation strategies: past experience and new approaches." *Malar* J, 8(Suppl 2), 2009

[43] E. Parzen, "On estimation of a probability density function and mode". *The annals of mathematical statistics,* 33(3), 1065-1076, 1962

[44] L. Prechelt, "A quantitative study of neural network learning algorithm evaluation practices. "*In proceedings of the 4th Int'l Conference on Artificial Neural Networks.* pp. 223-227, 1995

[45] DR. Raman, RR. Gerhardt, JB. Wilkerson, "Detecting insect flight sounds in the field: Implications for acoustical counting of mosquitoes." *Transactions of the ASABE*, 50(4), 1481, 2007

[46] SC. Reed, CM. Williams, LE. Chadwick, "Frequency of wing-beat as a character for separating species races and geographic varieties of Drosophila." *Genetics* 27: 349-361, 1942

[47] W. Reisen, "The western encephalitis mosquito, Culex tarsalis." *Wing Beats,* Vol. 4(2):16, 1993

[48] KS. Repasky, JA. Shaw, R. Scheppele, C. Melton, JL. Carsten, LH. Spangler, "Optical detection of honeybees by use of wing-beat modulation of scattered laser light for locating explosives and land mines." *Appl. Opt*., 45: 1839–1843, 2006

[49] M. Rosenblatt, "Remarks on some nonparametric estimates of a density function." *The Annals of Mathematical Statistics,* 832-837, 1956

[50] MW. Rowland, SW. Lindsay, "The circadian flight activity of Aedes aegypti parasitized with the filarial nematode Brugia pahangi." *Physiological entomology*, 11(3), 325-334, 1986

[51] SSC. Rund, SJ. Lee, BR. Bush, GE. Duffield, "Strain- and sex-specific differences in daily flight activity and the circadian clock of Anopheles gambiae mosquitoes." *Journal of Insect Physiology* 58: 1609-19, 2012

[52] L. Sawedal, R. Hall, "Flight tone as a taxonomic character in Chironomidae (Diptera)." *Entomol. Scand.* Suppl. 10: 139-143, 1979

[53] GW. Schaefer, GA. Bent, "An infra-red remote sensing system for the active detection and automatic determination of insect flight trajectories (IRADIT)." *Bull. Entomol.* Res. 74: 261-278, 1984

[54] J. Shotton, T. Sharp, A. Kipman, A. Fitzgibbon, M. Finocchio, A. Blake, M. Cook, R. Moore, "Real-time human pose recognition in parts from single depth images." *Communications of the ACM,* 56(1), 116-124, 2013

[55] O. Sotavalta, "The flight-tone (wing-stroke frequency) of insects (Contributions to the problem of insect flight 1.)." *Acta Entomol. Fenn*. 4: 1-114, 1947

[56] B. Taylor, "Geographical range and circadian rhythm." *Nature,* 222, 296-297, 1969

[57] B. Taylor, MDR. Jones, "The circadian rhythm of flight activity in the mosquito Aedes aegypti (L.): the phase-setting effects of light-on and light-off." *Journal of Experimental Biology* 51, no. 1 (1969): 59-70, 1969

[58] A. Tsymbal, "The problem of concept drift: definitions and related work." Computer Science Department, Trinity College Dublin, 2004

[59] DM. Unwin, CP. Ellington, "An optical tachometer for measurement of the wing-beat frequency of free-flying insects." *Journal of Experimental Biology*, 82(1), 377-378, 1979

[60] VN. Vapnik, AY. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities." *Theory of Probability and Its Applications,* 16(2), 264-280, 1971

[61] G. Widmer, M. Kubat, "Learning in the presence of concept drift and hidden contexts". *Machine learning*, 23(1), 69-101, 1996

[62] C. Zhan, X. Lu, M. Hou, X. Zhou (2005) "A lvq-based neural network anti-spam email approach." *ACM SIGOPS Oper Syst Rev* 39(1):34–39 ISSN 0163-5980, 2005

Chapter 3

[63] G. Batista, E. J. Keogh, A. Mafra-Neto, E. Rowton, "SIGKDD demo: Sensors and Software to allow Computational Entomology, an Emerging Application of Data Mining." *KDD*'11: 761-764, 2011.

[64] G. Batista, X. Wang, E. J. Keogh, "A Complexity-Invariant Distance Measure for Time Series," *SDM*'11: 699-710, 2011.

[65] N. Begum, B. Hu, T. Rakthanmanon, E. Keogh, "Towards a Minimum Description Length Based Stopping Criterion for Semi-Supervised Time Series Classification." *IRI* : 333 - 340, 2013.

[66] N. Begum, B. Hu, T. Rakthanmanon, E. Keogh, "A Minimum Description Length Technique for Semi-Supervised Time Series Classification." *Integration of Reusable Systems., Springer International Publishing*: 171-192, 2014.

[67] P. Chazal, M. O'Dwyer, R.B. Reilly, "Automatic classification of heartbeats using ECG morphology and heartbeat interval features," *IEEE Trans Biomed Eng*, 51: 1196–1206, 2004

[68] S. Cheng, Y. Shi, Q.Qin, "Particle swarm optimization based semi-supervised learning on Chinese text categorization," *CEC*: 1-8, 2012

[69] M. David, C. Eugene, Johnson. Mark, "Effective Self-Training for Parsing," *HLT-NAACL*: 152-159, 2006

[70] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, E. Keogh, "Querying and mining of time series data: experimental comparison of representations and distance measures," *PVLDB* 1(2): 1542-1552, 2008

[71] A. Goldberger, et al. PhysioBank, PhysioToolkit, and PhysioNet: *New Research Resource for Complex Physiologic Signals, Circulation* 101(23): 2000

[72] P. Grünwald, "A Tutorial Introduction to the Minimum Description Length Principle." 2005.

[73] M. Guillaumin, J. Verbeek, C. Schmid, "Multimodal semi-supervised learning for image classification," *CVPR*: 902-909, 2010

[74] M. Herwig, Google's Total Library: Putting the World's Books on the Web, 2007

[75] B. Hu, Y. Chen and E. Keogh, "Time Series Classification under More Realistic Assumption," *SDM*, 2013

[76] B. Hu, T. Rakthanmanon, Y. Hao, S. Evans, S. Lonardi, E. Keogh, "Discovering the Intrinsic Cardinality and Dimensionality of Time Series using MDL," *ICDM,* 2011.

[77] E. J. Keogh, Q. Zhu, B. Hu, Y. Hao, X. Xi, L. Wei, C. A. Ratanamahatana, C. A. (2011). *The UCR Time Series Classification/Clustering Homepage*

[78] E. J. Keogh, J. Lin, "Clustering of time-series subsequences is meaningless: implications for previous and future research." *KAIS* 8(2): 154-77, 2005

[79] E. J. Keogh, W. Li, X. Xi, S. Lee, M. Vlachos, "LB_Keogh Supports Exact Indexing of Shapes under Rotation Invariance with Arbitrary Representations and Distance Measures," *VLDB* : 882–893, 2006

[80] S. Lenser, M. Veloso, "Non-Parametric Time Series Classification," *ICRA*: 3918-3923, 2005

[81] X. Li, B. Liu, "Learning to classify text using positive and unlabeled data," *IJCAI*: 587-594, 2003

[82] W. Li, E. Keogh, "Semi-supervised time series classification," *ACM SIGKDD:* 2006

[83] A. Lomsadze, et al. "Gene identification in novel eukaryotic genomes by self-training algorithm," *Nucleic Acids Res.* 33: 6494–6506, 2005

[84] U. Maulik, D. Chakraborty, **"**A self-trained ensemble with semi-supervised SVM: An application to pixel classification of remote sensing imagery," *Pattern Recognition* 44(3):615-623, 2011

[85] M. N. Nguyen, X. Li, S. Ng, "Positive unlabeled learning for time series classification," *IJCAI* (2) , 2011

[86] M. N. Nguyen, X. L. Li, and S. K. Ng, "Ensemble Based Positive Unlabeled Learning for Time Series Classification." Database Systems for Advanced Applications. Springer Berlin/Heidelberg, 2012.

[87] P. Ordóñez, et al., "Visualization of Multivariate Time Series Data in a Neonatal ICU," *IBM Journal of Research and Development*, 2012

[88] PAMAP, Physical Activity Monitoring for Aging People, www.pamap.org/demo.html , retrieved 2012-05-12.

[89] D. Preston, P. Protopapas, C. E. Brodley, "Discovering arbitrary event types in time series." *Statistical Analysis and Data Mining 2(5-6)*: 396-411, 2009

[90] M. A. Ranzato, M. Szummer, "Semi-supervised learning of compact document representations with deep networks," *ICML*: 792-799, 2008

[91] M. Raptis, K. Wnuk, S. Soatto, "Flexible Dictionaries for Action Recognition," *MLVMA/ECCV*, 2008

[92] C. A. Ratanamahatana., D. Wanichsan, "Stopping Criterion Selection for Efficient Semi-supervised Time Series Classification." *SNPD* 2012. 149: 1-14, 2008.

[93] T. M. Rath, R. Manmatha, "Word Image Matching Using Dynamic Time Warping," *CVPR* (2): 521-527, 2003

[94] C. Rosenberg, M. Hebert, H. Schneiderman, "Semi-Supervised Self-Training of Object Detection Models," *WACV*: 29-36, 2005

[95] A. Sun, R. Grishman, "Semi-supervised Semantic Pattern Discovery with Guidance from Unsupervised Pattern Clusters," *COLING (Posters)*: 1194-1202, 2010

[96] P. Sykacek, S.J. Roberts, "Bayesian time series classification," *NIPS* : 937-944, 2001

[97] D.M.J. Tax, "One-class classification: Concept-learning in the absence of counter-examples.", *ASCI dissertation series,* 2001

[98] C. J. Van Rijsbergen, "Information Retrieval," 2nd edition, London, England: *Butterworths,* 1979

[99] J. Wang, A. Balasubramanian, L. Mojica de La Vega, J.R. Green, A. Samal, B. Prabhakaran, "Word recognition from continuous articulatory movement time-series data using symbolic representations," *ACL/ISCA Interspeech Workshop on Speech and Language Processing for Assistive Technologies,* Grenoble, France, 119-127, 2013

[100] X. Wang, L.Ye, E. J. Keogh, C. R. Shelton, "Annotating Historical Archives of Images," *IJDLS* 1(2): 59-80, 2010

[101] L. Wei and E. Keogh, "Semi-Supervised Time Series Classification." *SIGKDD* 2006.

[102] X. Xi, E. J. Keogh, C. R. Shelton, L. Wei, C. A. Ratanamahatana, "Fast time series classification using numerosity reduction," *ICML*: 1033-40, 2006

[103] X. Zhu, "Semi-Supervised Learning Literature Survey," Technical report, no. 1530, *Computer Sciences*, University of Wisconsin-Madison, 2005

[104] Das Ehrenbuch der Fugger (The secret book of honour of the Fugger) -BSB Cgm 9460, *Augsburg, ca. 1545 - 1548 mit Nachträgen aus späterer Zeit*

[105] Supporting webpage: *https://sites.google.com/site/yanpingdtwd/*

Chapter 4

[106] G.Batista, E. Keogh, A. Mafra-Neto, E. Rowton, "Sensors and software to allow computational entomology, an emerging application of data mining." *KDD*: 761-764, 2011

[107] M. Borazio and K. Laerhoven. "Combining Wearable and Environmental Sensing into an Unobtrusive Tool for Long-Term Sleep Studies". *2nd ACM SIGHIT* , 2012.

[108] A.S.L.O. Campanharo, M.I. Sirer, R.D. Malgren, F.M. Ramos, L.A.N Nunes. "Duality between time series and networks." *Plos One*, 6, p. e23378, 2011

[109] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka Jr. and T.M. Mitchell. "Toward an Architecture for Never-Ending Language Learning." *In Proc' AAAI*, 2010.

[110] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, E. J. Keogh. "Querying and mining of time series data. Experimental comparison of representations and distance measures." *PVLDB* 1(2): 1542-1552, 2008.

[111] Y Hao, "Supervised and Unsupervised Discovery of Structures in Large Data Archives," PhD disseration. *University of California, Riverside*, 2014.

[112] L. Mitchell. "Time Segregated Mosquito Collections with a CDC Miniature Light Trap. " *Mosquito News*. 42: 12, 1981

[113] C. L.Morales, M. P. Arbetman, S. A. Cameron, and M. A. Aizen. "Rapid ecological replacement of a native bumble bee by invasive species." *Frontiers in Ecology and the Environment*., 2013.

[114] B. Settles. "Active Learning." *Morgan & Claypool*, 2012.

[115] S. Zhai, P.O. Kristensson, C. Appert, T.H. Anderson, X. Cao. "Foundational Issues in Touch-Surface Stroke Gesture Design — An Integrative Review." *Foundations and Trends in Human-Computer Interaction.* 5, 2, 97-205, 2012

[116] Y. Chen project webpage: *https://sites.google.com/site/nelframework/*

Conclusions

[117] Y. Chen homepage: *www.cs.ucr.edu/~ychen053/*