UNIVERSITY OF CALIFORNIA
RIVERSIDE

Supervised and Unsupervised Discovery of Structures in Large Data Archives

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

Doctor of Philosophy

in

Computer Science

by

Yuan Hao

March 2014

Dissertation Committee:
- Dr. Eamonn Keogh, Chairperson
- Dr. Neal Young
- Dr. Marek Chrobak
- Dr. Christian Shelton

The Dissertation of Yuan Hao is approved:

_____

_____

_____

_____

Committee Chairperson

University of California, Riverside

Acknowledgements

Many people have helped me during the past couple of years. My greatest and sincerest gratitude goes to my advisor Dr. Eamonn Keogh for his guidance and patience. He brought me on a wonderful journey at University of California, Riverside and I will never forget the past four years, which has allowed me to learn from scratch and consistently improve. My work and even my life were deeply influenced by his enthusiasm, passion, and hard-working attitude in research. Eamonn, thank you for giving me an opportunity to work with you and I will always remember the following lessons I have learned from you: Always try simple ideas first, demonstrate ideas more expressively (through visualization), less tuning of algorithms, and take a more general approach to easily allow others to reproduce/extend your work.

I would like to take this opportunity to thank all my committee members as well. Dr. Christian Shelton, Dr. Neal Young, and Dr. Marek Chrobak. I really enjoyed Dr. Shelton's enthusiastic teaching when I took his course in machine learning. I greatly appreciate his time discussing research ideas and helpful advice for my work. I would also like to thank Dr. Young and Dr. Chrobak for providing the most fundamental and important class, design and analysis of algorithms, during my time here. I thank them for their generous support and suggestions in compiling this dissertation.

I also owe many thanks to my friends who helped me to endure all the difficult times for the past four years. They are Qiang Zhu, Xiaoyue Wang, Lexiang Ye, Bilson Campana, Jin Shieh, Art Rakthanmanon, Abdullah Mueen, Liudmila Ulanova,

Mohammad Shokoohi-Yekta, Yan Wang, Michael Butkiewicz, Robert Halstead, Guowu Xie, and Zhen Qin.

I had two internships with Google and LinkedIn. I worked with incredible people to whom I am greatly thankful. Dr. Grant Ye was my host in Google, he fully supported me and inspired my interest in other domains, such as image hashing and spam detection. I enhanced many skills during the time working with him. I would also like to thank Dr. Benjamin Arai, who was my manager when I was a data scientist intern in LinkedIn Corporation. Dr. Arai was one of the friendliest and most inspiring managers. He helped me with topic modeling algorithms to solve a very practical problem in LinkedIn, in addition to showing me how to be a good data scientist.

Lastly, I would like to deeply thank my parents who always support me. No matter how far we are separated, where I am, I always know and always feel your caring and encouragement. You are and will be always my biggest motivation to pursue my dreams.

ABSTRACT OF THE DISSERTATION

Supervised and Unsupervised Discovery of Structures in Large Data Archives

by

Yuan Hao

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, March 2014
Dr. Eamonn Keogh, Chairperson

Most domains of human interest now generate enormous, diverse data (text, time series, image, audio and video, etc) everyday. Extracting useful knowledge from such data in an efficient manner is an essential task for data mining community. A general framework to discover useful structures without domain-dependent tuning can also mitigate the costly manual efforts for different area experts, such as biologist, neurologist, cardiologist, etc.

This dissertation first discusses definitions and representations to find useful structures (for example, audio *fingerprint*, audio *motifs*) in audio archives, and further introduces scalable algorithms to allow application to diverse massive data archives. Audio fingerprints are "prototypical" subsequences that can represent a class, differentiate from

other classes and used to identify future unknown instances. We propose a supervised approach to classify animal sounds in the visual space, by treating the texture of their spectrograms as an acoustic fingerprint using a recently introduced parameter-free texture measure as a distance measure. Our audio fingerprint discovery bioacoustic framework assists biologists in automatically classifying different species of insects, (and, in follow up work by an independent research group) detect the presence of elephants in noisy environments, etc.

Motif discovery in contrast is an *unsupervised* process to find occurrences of repeated patterns when lacking any prior knowledge of patterns, even the pattern length. The audio motif/near duplicate pairs are the most similar segments among all the subsequences in any audio stream, however, they must be carefully defined in order to prevent finding pathological solutions. We propose a novel probabilistic early abandoning approach to cast the search for audio motifs into Anytime framework. We demonstrate that our algorithm can apply to diverse domains (i.e. mice vocalization, wild animal sounds, music and human speech) without requiring any domain specific tuning.

Lastly, we propose a never-ending learning framework for time series in which an agent examines an unbounded stream of data and occasionally asks a teacher (which may be a human or an algorithm) for a label. We demonstrate the utility of our ideas with experiments that consider real world problems in domains as diverse as medicine, entomology, wildlife monitoring, and human behavior analyses.

Contents

List of Tables xii

List of Figures xiii

1 Introduction **1**

1.1 Supervised Discovery of Structures (Audio Fingerprints) 2

1.2 Unsupervised Discovery of Structures (Audio Motif) 4

1.3 Never-Ending Learning of Time Series Streams 7

2 Monitoring and Mining Animal Sound in Visual Space **10**

2.1 Related Work and Background 11

2.1.1 A Brief Review of Spectrograms 11

2.1.2 General Animal Sound Classification 12

2.1.3 Sound Classification in Visual Space 15

2.1.4 A Review of the Campana-Keogh(CK) Distance Measure 16

2.2 Definitions 17

2.3 Sound Fingerprints 23

2.3.1 The Intuition of Sound Fingerprints 23

2.3.2 Formal Problem Statement and Assumptions 27
2.3.3 A Brute-Force Algorithm 29
2.3.4 Admissible Entropy Pruning 33
2.3.5 Euclidean Distance Ordering Heuristic 36
2.3.6 Disjunctive Sound Fingerprints 39
2.4 Experimental Evaluation 43
2.4.1 CK as a Tool for Taxonomy 43
2.4.2 Insect Classification 45
2.4.3 Monitoring with Sound Fingerprints 47
2.4.4 Scalability of Fingerprint Discovery 48
2.4.5 Robustness 49
2.4.6 Insect Classification with Disjunctive Sound Fingerprints 50
2.5 Conclusion 53
3 Parameter-Free Audio Motif Discovery in Large Data Archives 54
3.1 Related Work and Background 55
3.2 Definitions 60
3.3 Finding Audio Motifs 63
3.3.1 Intuition behind Audio Motif Discovery 63
3.3.2 A Formal Description of our Algorithm 67
3.3.3 Brute-force Algorithm 68
3.3.4 Random Search Algorithm 69
3.3.5 Euclidean Distance Ordering Algorithm 70

3.3.6 Probabilistic Motif Discovery Algorithm 72
3.4 Experiments 75
3.4.1 Motif Discovery in Human Speech 75
3.4.2 Motif Discovery in Bird Songs 77
3.4.3 Motif Discovery in Music Data 78
3.4.4 Motif Discovery in Mice Vocalization 79
3.4.5 Scalability of Audio Motif Discovery 82
3.4.6 Sensitivity of User-Choice (Motif Length) 85
3.5 Conclusion 86
4 Towards Never-Ending Learning from Time Series Streams 88
4.1 Related Work 89
4.2 Algorithms 90
4.2.1 Overview of System Architecture 91
4.2.2 Class Dictionaries 94
4.2.3 Subsequence Processing 95
4.2.4 Frequent Pattern Maintenance 97
4.2.5 Active Learning System 104
4.3 Experiments 109
4.3.1 Activity Data 110
4.3.2 Invasive Species of Flying Insects 111
4.3.3 Long Term Electrocardiogram 114
4.3.4 Bird Song Classification 117

4.3.5 Understanding Sapsucking Insect Behavior 118

4.3.6 Weak Teaching Example: Elder Care 120

4.4 Conclusion 123

5 Conclusion 124

Bibliography 126

List of Tables

2.1 The CK Distance Measure 17

2.2 Insect Classification Accuracy 46

3.1 Comparison between PEAMD and Brute-Force Algorithms 84

4.1 Our Ability to Detect then Classify Invasive Insects. 113

4.2 The ground truth frequencies of beats in $BIDMC_{ch}07$. 114

4.3 Results on $BIDMC_{ch}07$. 116

List of Figures

1.1 *Top*) Seven seconds of audio produced by a male mouse. *middle*) We searched the spectrogram of this data for repeated patterns of length 0.5 seconds. *bottom left*, *right*) A zoom-in of the two repeated occurrences reveals their similarity. We will revisit this domain in Section 3.4.4. 6

1.2 The light sensors at Soda Hall produce a never-ending time series, of which we can cache only a small subset main memory. 8

1.3 *Left*) A "motif" of two patterns annotated in Figure 1.2 aligned to highlight their similarity. *center*) We imagine asking a teacher for a label for the pattern. *right*) This allows us to detect and classify a new occurrence eleven days later. 8

2.1 A spectrogram of the call of an insect. Note the highly repetitious nature of the call. In this case, capturing just two "busts" may be sufficient to recognize the insect.. 11

2.2 A candidate sound fingerprint, the boxed region in spectrogram 1, is evaluated by finding its nearest neighbor subsequence within both ***P*** and the four representatives of ***U*** and then sorting all objects on a number line..... 21

2.3 Two order lines that have the same information gain. Our tie-breaking policy reflects the intuition that the *top* line achieves less separation than the bottom line 22

2.4 A clustering of six pairs of one-second recordings of various katydids and crickets using the CK texture measure. Only one species pair {3,4} is correctly grouped. Ideally the pairs {1,2}, {5,6}, {7,8}, {9,10} and {11,12} should also be grouped together 26

2.5 A clustering of the same data used in Figure 2.4, after trimming irrelevant prefix and suffix data. All pairs are correctly grouped, and at a higher level the dendrogram separates katydids and crickets 27

2.6 A trace of value of the *bsf_Gain* variable during brute force search on the *Atlanticus dorsalis* dataset. Only sound fingerprints of length 16 are considered here for simplicity 32

2.7 The order line of all the objects in ***P*** and just the first two objects in ***U*** 34

2.8 The logically best possible order line based on the distances that have been calculated in Figure 2.7. The best split point is shown by the yellow/heavy line ... 34

2.9 A trace of the *bsf_Gain* variable during brute force and entropy pruning search on the *Atlanticus dorsalis* dataset 36

2.10 The relationship between Euclidean and CK distance for 1,225 pairs of spectrograms 38

2.11 A trace of value of the *bsf_Gain* variable during brute force, entropy pruning, and reordering optimized search on the *Atlanticus dorsalis* dataset 38

2.12 *Left*) A toy problem showing instances of crickets (*Species A*) and katydids (*Species B*). In this case a single sound fingerprint A can separate the two classes. *center*) In contrast, *Species C* is an example where a single sound fingerprint

cannot separate the classes, perhaps because the insects are sexually dimorphic with the males and females singing differently. *right*) Two sound fingerprints C_1 and C_2 connected by an *OR* operation *can* better separate the classes 40

2.13 *Top left*) An insect found in Florida: is it a *G. rubens* or *G. firmus*? *top right*) Projecting one-second snippets of songs from both insects into 2D space suggests they are almost linearly separable, a possibility reflected by their clusterability (*bottom*) 44

2.14 (Image best viewed in color) *far left*) Three insect sound fingerprints are used to monitor an eight- second audio clip. In each case, the fingerprint distance to the sliding window of audio dips below the threshold as the correct species sings, but not when a different species is singing. 47

2.15 A trace of value of the *bsf_Gain* variable during brute force, entropy pruning, and reordering optimized search on the *Atlanticus dorsalis* dataset with the 200-object universe. 48

2.16 (Image best viewed in color) Three frog sound fingerprints are again used to monitor a fourteen-second audio clip on the left. In each case, the fingerprint distance to the sliding window of audio dips below the threshold as the correct species sings, but not when a different species is singing 49

2.17 *Top*, *middle*) Sound fingerprint of *Atlanticus dorsalis* is used to monitor a twenty-second testing audio clip without (with) mislabeled data in ***P***. The fingerprint distance to the sliding window of audio dips below the threshold as the correct species sings, but not when a different species is singing. *bottom*) Forty second

sequence of audio by concatenating snippets of testing audio clips from ***P*** and ***U*** alternatively 50

3.1 An illustration of our definitions. 62

3.2 A clustering of seven pairs of two-second audio recordings of various sounds using SIFT. Only one pair {13,14} is correctly clustered (*katydids*)..... 64

3.3 A clustering of the dataset used in Figure 3.2 with CK distance measure. All pairs are correctly clustered, and the explosion sound is an outlier to animal sounds. No parameters were adjusted here. 65

3.4 *Top*) A performance of *A Dream Within A Dream* has a motif of length seven seconds. *bottom left*, *right*) A zoom-in of the two occurrences and the corresponding sentences. The reader can go to [5] to hear the original sound file and the discovered motifs. 66

3.5 *Left*) The empirical relationship between Euclidean and CK distance. *right*) As we search in Euclidean order (the y-axis order), from bottom to top. The *best-so-far* distance moves leftwards and the mean of the Gaussian distribution moves rightwards. 73

3.6 *Top*) A performance of *The Cat in the Hat* has a motif three seconds long. *bottom*) A zoom-in of the two occurrences and corresponding sentences [34]. 76

3.7 *Top*) A screenshot from [54] of a state-of-the-art human speech recognition algorithm correctly matching two utterances of "*California*" (red box). *bottom*) Our re-creation of the experiment using only the CK distance measure. 77

3.8 *Top*) A 31-second excerpt of a two-minute audio performance of a Common Scimitarbill. *bottom*) A zoom-in of the two one-second long audio motif occurrences. 78

3.9 *Top*) A performance of Bourvil's song *C'était bien* has a three-second motif. bottom) A zoom-in of the two motif occurrences and corresponding lyrics. 79

3.10 *Top*) Sample instances of a motif discovered from mice vocalizations by applying our algorithm. *middle*) Comparing the number of motifs during *S* and *R* behaviors for a sample recording of *KO* mice vocalization. *bottom*) Comparing the number of motifs during *S* and *G* behaviors for a sample recording of WT mice vocalization. 81

3.11 *Top*) A performance of "The Raven" has a motif of length seven seconds. *bottom*) A zoom-in of the two occurrences and the corresponding text. 82

3.12 A comparison of efficiency of four algorithms normalized to the 100% time taken for brute-force search. 83

4.1 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*). 92

4.2 An illustration of the expressiveness of our model. 94

4.3 *Left*) Sections of constant "flatline" signals are so common in medical domains that it is worth initializing the medical dictionaries with an example. *right*), thus suppressing the need to waste a query asking a teacher for a label for it. 95

4.4 A visual intuition of our solution to the frequent time series subsequence problem. The elements in a dense subtree (or clade) can be seen as a frequent pattern..... 99

4.5 *Left*) The (partial) dendrogram shown in Figure 4.4 has its subtrees of size four ranked by density. *right*) The observed heights of the subtrees are compared to the expected heights given the assumption of no patterns in the data..... 100

4.6 The average number of time steps required to find a repeated pattern with a desired probability, for various values of w. All curves end when they reach 99.5%..... 103

4.7 An illustration of a weak teacher. *top*) A stream *P* in which we detect three occurrences of the pattern C_1. *middle*) At the time of detection we poll a set of binary sensors to see which of them are active. *bottom*) we can use the frequency of associations between a pattern and binary "votes" to calculate probabilities for C_1's class label. 108

4.8 *Left*) A query shown to the user during a run on the activity dataset, the teacher labeled it Pushing and a new concept C_1 was added to the dictionary. *right*) About 9.6 minutes later the classifier detected a new example of the class..... 110

4.9 *Top*) An audio snippet of a female *Cx*. *stigmatosoma* pursued by a male. *bottom left*) An audio snippet of a common house fly. *bottom right*) If we convert these sound snippets into periodograms we can cluster and classify the insects. 111

4.10 A small snippet (0.0065%) of $BIDMC_{ch}07$ Lead 1. 114

4.11 *Left to right*) Three patterns discovered in our ECG experiment. *top to bottom*) The motif discovered and used to query the teacher. The learned concept. Some examples of true positives. Some examples of false positives. 115

4.12 A pattern (green/bold) show with surrounding data for context, discovered in lead 2 of $BIDMC_{ch}07$. 116

4.13 The motif discovered in the first run on the bird dataset. *right*) One snippet in three representations, bottom-to-top a spectrogram, an oscillogram and the MFCC we used. 118

4.14 *Left*) A tethered brown leafhopper. *right*) A schematic diagram of the circuit for recording EPGs. *bottom*) A snippet of data produced during one of our experiments. 119

4.15 *Top-row*) The two concepts discovered in the EPG data. *bottom-row*) Examples of classified patterns. 120

4.16 *Top*) After we have learned the concept C_1 our system monitors for future occurrences of it. Here it sees three examples in a row. *bottom*) By polling the binary RFID sensors when a "hit" for C_1 is detected, we can learn that the concept is associated with 'glove'. 121

4.17 *Top*) A zoom-out of the time series shown in Figure 4.16. *bottom*) The probability of concept C_1 been associated with various items. Of 38 possibilities only 3 have non zero entries. 122

4.18 *Top-left*) The motif discovered in our Elder care sensor experiment and averaged into concept C_1 (*bottom-left*). Examples of true positives and false positives. 122

Chapter 1

Introduction

Many (or most) domains produce rich, numerous data, including text, time series, image, audio, and video, etc. The massive amount of data produced every day is a challenge not only for domain experts but also to the data mining researchers that support them. Social networks, for example, such as Facebook, LinkedIn or Twitter handle hundreds of millions of pieces of user information. Researchers analyze environmental biodiversity by monitoring various animal sounds, which can produce gigabytes of data per day. Long term daily human behavior tracking has shown great potential application for disease prevention using sleep monitoring [37], heartbeat studies [41] etc. Data mining experts have worked on such diverse, large-scale projects for at least a decade. However, most state-of-the-art algorithms are not very general, they provide solutions only under many assumptions (stated or unstated).

Furthermore, mining even a fraction of these data using complex models (i.e. decision trees, neural networks, etc.) usually takes a long time. Secondly, most of these data is generated automatically without careful annotations (i.e. class labels), which is assumed

by most algorithms to achieve high accuracy. Therefore, building a general, online, "lightly-labeled" framework to analyze various, diverse domains data is still an open problem in many real world applications.

In the next three sections of this chapter, we briefly introduce *audio fingerprints*, *audio motif*, and a novel never-ending learning system. In Chapter 2, we describe the audio fingerprint algorithm, and a general framework for monitoring animals' behavior by the sounds they produce. In Chapter 3, we discuss audio motifs, their utility in diverse domains and how they can be found efficiently. Both of these frameworks have already attracted attention by domain experts. For example, our sound fingerprint method was used by Zeppelzauer et al [97] to detect elephants in the wild. They noted "*Experiments show that the selected templates clearly captured parts of the rumbles, which confirms that the template section works well*". In Chapter 4, we present a never ending learning system for time series data, which is an online, active learning framework to solve the problems we discuss above. Finally, we conclude with a summary and possible future work in Chapter 5.

1.1 Supervised Discovery of Structures (Audio Fingerprints)

Monitoring animals by the sounds they produce is an important and challenging task, whether the application is outdoors in a natural habitat [19], or in the controlled environment of a laboratory setting.

In the former case the density and variety of animal sounds can act as a measure of biodiversity and of the health of the environment. Algorithms are needed here not only

because they are (in the long term) cheaper than human observers, but also because in at least some cases algorithms can be more accurate than even the most skilled and motivated observers [73].

In addition to field work, researchers working in laboratory settings frequently create control and treatment groups of animals, expose them to different interventions, and test for different outcomes. One possible manifestation of different outcomes may be changes in the bioacoustics of the animals. To obtain statistically significant results researchers may have to monitor and hand-annotate the sounds of hundreds of animals for days or weeks, a formidable task that is typically outsourced to students [76].

There are also several important commercial applications of acoustic animal detection. For example, the US imports tens of billions of dollars worth of timber each year. It has been estimated that the inadvertent introduction of the Asian Longhorn Beetle (*Anoplophora glabripennis*) with a shipment of lumber could cost the US lumber industry tens of billions of dollars [75]. It has been noted that different beetle species have subtlety distinctive chewing sounds, and ultra sensitive sensors that can detect these sounds are being produced [65]. As a very recent survey of acoustic insect detection noted, "*The need for nondestructive, rapid, and inexpensive means of detecting hidden insect infestations is not likely to diminish in the near future*" [75].

With such a plethora of important applications, there have been significant efforts to build bioacoustic classification tools [19]. However, we argue that current tools are severely limited. They often require the careful tuning of many parameters (as many as eighteen [32]) and thus huge amounts of training data, they are too computationally

expensive for use with resource-limited sensors that will be deployed in the field [30], they are specialized for a very small group of species, or they are simply not accurate enough to be useful.

In this thesis, we introduce a novel bioacoustic recognition/classification framework that mitigates or solves all of the above problems. We propose to classify animal sounds in the *visual space*, by treating the texture of their spectrograms as an acoustic "fingerprint" and using a recently introduced parameter-free texture measure as a distance measure. We further show that by searching for the smallest representative acoustic fingerprint (inspired by the *shapelet* concept in time series domain [93]) in the training set, we can significantly outperform other techniques in terms of both speed and accuracy.

Note that monitoring of animal sounds in the wild opens up a host of interesting problems in sensor placement, wireless networks, resource-limited computation [30], etc. For simplicity, we gloss over such considerations, referring the interested reader to [19] and the references therein. In this work we assume all such problems have been addressed, and only the recognition/classification steps remain to be solved.

1.2 Unsupervised Discovery of Structures (Audio Motif)

The first step in most exploratory data mining endeavors is the discovery and enumeration of *repeated structure*. This has been true even for data analysis that predates computers. For example, the decipherment of documents written in ancient unknown languages first requires the discovery of *repeated* elements in the scripts [23]. Given this,

there has been significant research effort in the last decade focused on repeated pattern (motif/near-duplicate) discovery in text, DNA, graphs, time series, images, and video [47][55][58]. In contrast, the discovery of *audio* motifs, with the sole exception of music data, has not received much attention [49]. However, identifying structure in general audio sequences is an important and challenging task with applications in many diverse domains. Some representative examples include:

- Acoustic wildlife monitoring has been shown to allow effective and non-invasive measurement of the health of ecosystems [88].
- A powerful tool for investigating the role of genetics in human disorders modifies ("knocks out") various genes in mice and examines their vocalizations for changes that may be linked to those genes, and hence the analogue genes in humans [83][95]. Figure 1.1 hints at the utility of this idea, which we will revisit in Section 3.4.4. In recent years this framework has emerged as an extremely promising tool for understanding human cognitive and memory disorders.
- Audio content analysis has been shown to assist with *video* segmentation and summarization [47][55][77].

The above are in addition to the more obvious applications in the music domain, such as analysis, thumbnailing, retrieval, and summarization [11].

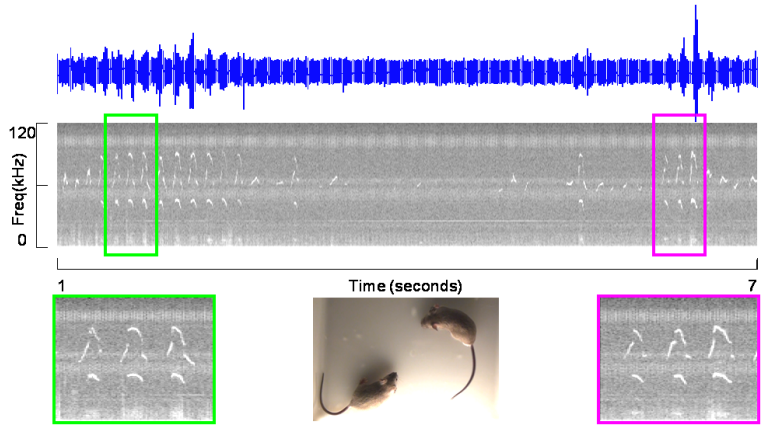

Figure 1.1: *top*) Seven seconds of audio produced by a male mouse. *middle*) We searched the spectrogram of this data for repeated patterns of length 0.5 seconds. *bottom left*, *right*) A zoom-in of the two repeated occurrences reveals their similarity. We will revisit this domain in Section 3.4.4.

Thus far virtually all research efforts aimed at finding repeated patterns in audio sequences use feature extraction algorithms to produce low cardinality symbolic representations of the data and use suffix trees, hashing, or similar techniques to search these symbolic strings for approximately repeated elements [11]. The problem with this approach is that the feature extraction step must be highly tuned to the domain. For example, Zakaria et. al [95] demonstrate a technique to find motifs in vocalizations of a specific strain of lab mice called *Fmr1-KO*. However, it is not clear if this multi-stage algorithm (which requires significant human intervention) generalizes to other strains of mice, much less to other rodents.

In contrast to these efforts, we propose an algorithm which is completely general, makes zero assumptions about the data, and is essentially parameter-free. We achieve this by leveraging off the growing realization that for at least some audio similarity problems, we can best measure similarity when the data is transformed into the *image space* (i.e.

spectrograms) [58]. Image processing algorithms themselves are not generally devoid of the need for feature extraction. However, we propose to use the CK distance measure [24], a recently introduced compression-based measure that avoids explicitly extracting any features, thus remains parameter-free. We will show that the CK distance measure is so efficient that even a brute-force implementation can run in about real-time for a typical song. For longer audio sequences we introduce two ideas to mitigate the time complexity. First, we show that we can cast the search for audio motifs into an anytime framework [61]. Second, we can derive confidence bounds that allow searches to return the optimal audio motifs with some bounded probability of error. As we shall show, even if we allow a very conservative probability of error, we can achieve a massive speedup.

1.3 Never-Ending Learning of Time Series Streams

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. 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 [27], 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 1.2 we show a time series produced by a light sensor at Soda Hall in Berkley. 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.

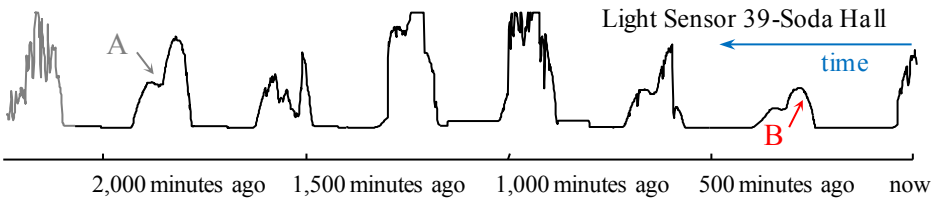

Figure 1.2: The light sensors at Soda Hall produce a never-ending time series, of which we can cache only a small subset main memory.

As we can see in Figure 1.3.*left* and Figure 1.3.*center*, these patterns are even more similar after we z-normalize them **[**33**]**. Suppose that the appearance of these two similar patterns (or "motif") causes an agent to query a teacher as to their *meaning*.

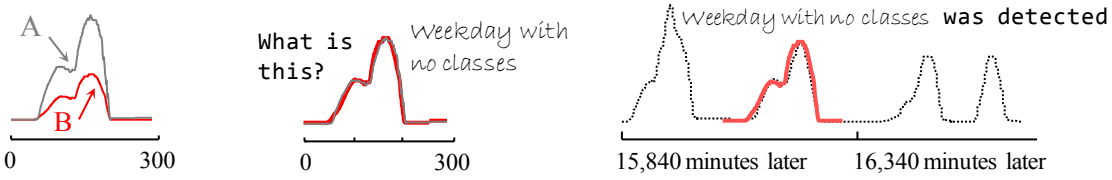

Figure 1.3: *left*) A "motif" of two patterns annotated in Figure 1.2 aligned to highlight their similarity. *center*) We imagine asking a teacher for a label for the pattern. *right*) This allows us to detect and classify a new occurrence eleven days later.

This query could be implemented in a number of ways; moreover the teacher need not necessarily be human. Let us assume here 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 **[**25**]**.

This example elucidates our aims, but suggested 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? We will discuss all these questions in Chapter 4.

Chapter 2

Monitoring and Mining Animal Sound in Visual Space

This chapter discusses building a general bioacoustic framework to eliminate huge amount of manual effort to monitor animals by the sounds they produce. It is an important and challenging task, whether the application is outdoors in a natural habitat, or in the controlled environment of a laboratory setting.

In the former case the density and diversity of animal sounds can act as a measure of biodiversity. In the latter case, researchers often create control and treatment groups of animals, expose them to different interventions, and test for different outcomes. One possible manifestation of different outcomes may be changes in the bioacoustics of the animals.

With such a plethora of important applications, there have been significant efforts to build bioacoustic classification tools. However, we argue that most current tools are severely limited. They often require the careful tuning of many parameters (and thus huge amounts of training data), they are too computationally expensive for deployment in

resource-limited sensors, they are specialized for a very small group of species, or they are simply not accurate enough to be useful.

In this work we introduce a novel bioacoustic recognition/classification framework that mitigates or solves all of the above problems. We propose to classify animal sounds in the *visual space*, by treating the texture of their spectrograms as an acoustic fingerprint using a recently introduced parameter-free texture measure as a distance measure. We further show that by searching for the most representative acoustic fingerprint we can significantly outperform other techniques in terms of speed and accuracy.

2.1 Related Work and Background

2.1.1 A Brief Review of Spectrograms

As hinted at in Chapter 1.1, we intend to do recognition/classification in the visual space, by examining the spectrogram of the animal sounds. As shown in Figure 2.1, a spectrogram is a time-varying spectral representation that shows how the spectral density of a signal varies with time.

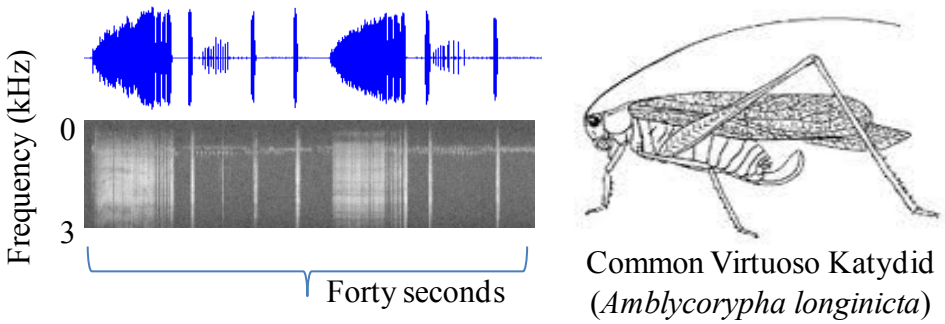

Figure 2.1: A spectrogram of the call of an insect. Note the highly repetitious nature of the call. In this case, capturing just two "busts" may be sufficient to recognize the insect.

There is a huge amount of literature leveraging off *manual* inspection of such spectrograms; see [48] and the references therein for some examples. However, as we shall see, algorithmic analysis of spectrograms remains an open problem, and an area of active research. Beyond the problems that plague attempts to define a distance measure in any domain, including invariance to offset, scaling, uniform scaling, non-uniform warping, etc., spectrograms almost always have significant noise artifacts, even when obtained in tightly controlled conditions in a laboratory setting. One avenue of research is to "clean" the spectrograms using various techniques [16], and then apply shape similarity measures to the cleaned shape primitives. Some types of specialized cleaning may be possible; for example, removing the 60Hz noise is commonly encountered[1]. However, algorithms to robustly clean general spectrograms seem likely to elude us for the foreseeable future.

As we shall see in Section 2.3, our solution to this problem is to avoid any type of data cleaning or explicit feature extraction, and use the raw spectrogram directly.

2.1.2 General Animal Sound Classification

The literature on the classification of animal sounds is vast; we refer the interested reader to [6][70] for useful surveys. At the highest level, most research efforts advocate extracting sets of features from the data, and using these features as inputs to standard classification algorithms such as a decision tree, a Bayesian classifier or a neural

[1] American domestic electricity is at 60Hz (most of the rest of the world is 50Hz) and inadequate filtering in power transformers often allows some 60Hz signal to bleed into the sound recording.

network. As a concrete representative example, consider [78], which introduces a system to recognize *Orthoptera* (the order of insects that includes grasshoppers, crickets, katydids[2] and locusts). This method requires that we extract multiple features from the signal, including *distance-between-consecutive-pulses, pulse-length, frequency-contour-of-pulses, energy-contour-of-pulses, time-encoded-signal-of-pulses*, etc. However, robustly extracting these features from noisy field recordings is non-trivial, and while these features seem to be defined for many *Orthoptera*, it is not clear that they generalize to other insects, much less to other animals. Moreover, a significant number of parameters need to be set, both for the feature extraction algorithms, and the classification algorithm.

For more complex animal sounds (essentially all non-insect animals), once again features are extracted from the raw data; however, because the temporal transitions between features are themselves a kind of meta-feature, techniques such as Hidden Markov Models are typically used to model these transitions [6] [22][70]. This basic idea has been applied with varying degrees of success to birds [60], frogs and mammals [22].

One major limitation of Hidden Markov Model-based systems is that they require careful tuning of their many parameters. This in turn requires a huge amount of labeled training data, which may be difficult to obtain in many circumstances for some species.

Many other approaches have been attempted in the last decade. For example, in a series of papers, Dietrich et al. introduce several classification methods for insect sounds,

[2] In British English, *katydids* are known as *bush-crickets*.

some of which require up to eighteen parameters, and which were trained on a dataset containing just 108 exemplars [32].

It is important to note that our results are *completely* automatic. Numerous papers report high accuracies for the classification of animal sounds, but upon careful reading it appears (or it is explicitly admitted) that human effort was required to extract the right data to give to the classifier. Many authors do not seem to fully appreciate that "extracting the right data" is *at least* as difficult as the classification step.

For example, a recent paper on the acoustic classification of Australian anurans (frogs and toads) claims a technique that is "*capable to identify the species of the frogs with an average accuracy of 98%.*" [46]. This technique requires extracting features from syllables, and the authors note, "*Once the syllables have been properly segmented, a set of features can be calculated to represent each syllable*" (our emphasis). However, the authors later make it clear that the segmentation is done by careful human intervention.

In contrast, we do not make this unrealistic assumption that all the data has been perfectly segmented. We do require sound files that are labeled with the species name, but nothing else. For example, most of the sound files we consider contain human voiceover annotations such as "*June 23th, South Carolina, Stagmomantis carolina, temperature is ...*" and many contain spurious additional sounds such as distant bird calls, aircraft, the researcher tinkering with equipment, etc. The raw unedited sound file is the input to our algorithm; there is no need for costly and subjective human editing.

2.1.3 Sound Classification in Visual Space

A handful of other researchers have suggested using the visual space to classify sounds (see [66][67]). However, this work has mostly looked at the relatively simple task of recognizing musical instruments or musical genres [94], etc. More recent work has considered addressing problems in bioacoustics in the visual space. In [67] the authors consider the problem of recognizing whale songs using spectrograms. The classification of an observed acoustic signal is determined by the maximum cross-correlation coefficient between its spectrogram and the specified template spectrogram [67]. However, this method is rather complicated and indirect: a "correlation kernel" is extracted from the spectrogram, the image is divided into sections which are piecewise constant, and a cross-correlation is computed from some subsets of these sections and thresholded to obtain a detection event. Moreover, at least ten parameters must be set, and it is not clear how best to set them, other than using a brute force search through the parameter space. This would require a huge amount of labeled training data. In [66] the authors propose similar ideas for bird calls. However, beyond the surfeit of parameters to be tuned, these methods have a weakness that we feel severely limits their applicability. Both these efforts (and most others we are aware of) use *correlation* as the fundamental tool to gauge similarity. By careful normalization, correlation can be made invariant to shifts of pitch and amplitude. However, because of its intrinsically linear nature, correlation cannot be made invariant to global or local differences in time (in a very slightly different context, these are called *uniform scaling* and *time warping*, respectively [38]). There is significant evidence that virtually all real biological signals have such

distortions, and that unless it is explicitly addressed in the representation or classification algorithm, we are doomed to poor accuracy. As we shall show empirically in the experimental section below, our proposed method is largely invariant to *uniform scaling* and *time warping*.

2.1.4 A Review of the Campana-Keogh (CK) Distance Measure

The CK distance measure is a recently introduced measure of texture similarity [24]. Virtually all other approaches in the vast literature of texture similarity measures work by explicitly extracting features from the images, and computing the distance between suitably represented feature vectors. Many possibilities for features have been proposed, including several variants of wavelets, Fourier transforms, Gabor filters, etc. [18]. However, one drawback of such methods is that they all require the setting of many parameters. For example, at a minimum, Gabor filters require the setting of scale, orientation, and filter mask size parameters. This has led many researchers to bemoan the fact that "*the values of (Gabor filters parameters) may significantly affect the outcome of the classification procedures*..."[18].

In contrast, the CK distance measure does not require any parameters, and does not require the user to create features of any kind. Instead, the CK measure works in the spirit of Li and Vitanyi's idea that two objects can be considered similar if information garnered from one can help compress the other [59][62]. The theoretical implications of this idea have been heavily explored over the last eight years, and numerous applications for discrete data (DNA, natural languages) have emerged.

The CK measure expands the purview of the compression-based similarity measurements to *real-valued images* by exploiting the compression technique used by MPEG video encoding [24]. In essence, MPEG attempts to compress a short video clip by taking the first frame as a template, and encoding only the *differences* of subsequent frames. Thus, if we create a trivial "video" consisting of just the two images we wish to compare, we would expect the video file size to be small if the two images are similar, and large if they are not. Assuming x and y are two equally-sized images; Table 2.1 shows the formula to achieve this.

dist = ((mpegSize(*x*,*y*) + mpegSize(*y*,*x*)) /(mpegSize(*x*,*x*) + mpegSize(*y*,*y*))) - 1;

Table 2.1: The CK Distance Measure.

It is worth explicitly stating that this is not pseudo code, but the *entire* actual Matlab code needed to calculate the CK measure.

The CK measure has been shown to be very effective on images as diverse as moths, nematodes, wood grains, tire tracks, etc. [24]. However, this is the first work to consider its utility on spectrograms.

2.2 Definitions

In this section we define the necessary notation for our sound fingerprint finding algorithm. We begin by defining the data type of interest, an *audio sequence*:

Definition 2.1 [AUDIO SEQUENCE] A *audio sequence* **A** of length $m > 0$ is a sequence $\mathbf{A} = (A_1, A_2, \ldots, A_m)$ of m real-valued numbers corresponding to the amplitude

at that time stamp. The data points are typically generated in temporal order and spaced at uniform time intervals.

As with other researchers [66][67], we are interested in the *audio sequence* representation in the *visual space*, which is called the *spectrogram*.

Definition 2.2 [SOUND SPECTROGRAM] A *sound spectrogram* **S** is an image of time-varying spectral representation, produced by applying the Short Fast Fourier Transform to successive overlapping frames of an *audio sequence*. The horizontal dimension corresponds to time and the vertical dimension corresponds to frequency. The relative spectral intensity of a *sound* at any specific time and frequency is indicated by the color/grayscale intensity of the *image*.

A more detailed discussion of spectrograms is beyond the scope of this thesis, so we refer the reader to [6] and the references therein.

We are typically interested in the *local* properties of the audio sequence rather than the *global* properties, because the entire audio sequence may be contaminated with extraneous sounds (human voice annotations, passing aircraft, etc.). Moreover, as we shall see, our ultimate aim is to find the *smallest* possible sound snippet to represent a species. A local subsection of a spectrogram can be extracted with a *sliding window*:

Definition 2.3 [SLIDING WINDOW] A *sliding window* (*W*) contains the latest *w* data points (S_{t-w+1}, S_{t-w+2},…, S_t) in the sound spectrogram **S**.

Within a sliding window, a local subsection of the audio sequence we are interested in is termed as a *subsequence*.

Definition 2.4 [AUDIO SUBSEQUENCE] An *audio subsequence* of length n of an *audio* sequence $\mathbf{A} = (A_1, A_2, \ldots, A_m)$ is a time series $\mathbf{A}_{i,n} = (A_i, A_{i+1}, \ldots, A_{i+n-1})$ for all integers i, where $0<i<m-n+1$.

Since our algorithm attempts to find the prototype of an audio sequence **A**, ultimately, a local audio subsequence $\mathbf{A}_{i,n}$ should be located with a distance comparison between **A** and $\mathbf{A}_{i,n}$, which may be of vastly different lengths. Recall that the CK distance is only defined for two images of the same size.

Definition 2.5 [DISTANCE] The *distance* ***d*** between a *subsequence* $\mathbf{A}_{i,n}$ and a longer *audio sequence* **A** is the minimum distance between $\mathbf{A}_{i,n}$ and all possible subsequences in **A** that are the same length as $\mathbf{A}_{i,n}$.

Our algorithm needs some evaluation mechanism for splitting datasets into two groups (*target class*, denoted as ***P***, *everything else*, denoted as ***U***). We use the classic machine learning idea of *information gain* to evaluate candidate splitting rules. To allow discussion of *information gain*, we must first review *entropy*:

Definition 2.6 [ENTROPY] The *entropy* for a given *audio sequence* dataset **D** is $E(\mathbf{D}) = -p(X)\log(p(X))-p(Y)\log(p(Y))$, where X and Y are the positive and universe classes in **D**, $p(X)$ is the proportion of objects in class X and $p(Y)$ is the proportion of objects in class Y.

The *information gain* is for a given splitting strategy and is just the difference in entropy before and after splitting. More formally:

Definition 2.7 [INFORMATION GAIN] The *information gain* of a partitioning of dataset **D** is:

$$Gain = E(\mathbf{D}) - E'(\mathbf{D}),$$

where $E(\mathbf{D})$ and $E'(\mathbf{D})$ are the entropy before and after partitioning $\mathbf{D}$ into $\mathbf{D_1}$ and $\mathbf{D_2}$, respectively.

$$E'(\mathbf{D}) = f(\mathbf{D_1})E(\mathbf{D_1}) + f(\mathbf{D_2})E(\mathbf{D_2}),$$

where $f(\mathbf{D_1})$ is the fraction of objects in $\mathbf{D_1}$, and $f(\mathbf{D_2})$ is the fraction of objects in $\mathbf{D_2}$.

As noted above, we wish to find a sound fingerprint such that most or all of the objects in ***P*** of the dataset have a subsequence that is similar to the *fingerprint*, whereas most of the audio sequences in ***U*** do not. To find such a *fingerprint* from all possible candidates, we compute the distance between each candidate and every subsequence of the same size in the dataset, and use this information to sort the objects on a number line, as shown in Figure 2.2. Given such a linear ordering, we can define the best splitting point for a given sound fingerprint:

Definition 2.8 [BEST SPLITTING POINT] Given an annotated (by one of two classes, ***P*** and ***U***) linear ordering of the objects in $\mathbf{D}$, there exists *at most*[3] $|\mathbf{D}|$-1 distinct splitting points which divide the number line into two distinct sets. The splitting point which produces the largest information gain is denoted as the *best splitting point*.

In Figure 2.2 we illustrate the *best splitting point* with a bold/yellow vertical line.

We are finally in a position to define the sound fingerprint using the above definitions:

Definition 2.9 [SOUND FINGERPRINT] The *sound fingerprint* for a species is the *subsequence* from ***P***, together with its corresponding *best splitting point*, which produces the largest information gain when measured against the universe set ***U***.

[3] Note that there can be duplicate values in the ordering.

Note that we may expect ties, which must be broken by some policy. We defer a discussion of tie-breaking policies to later in this section.

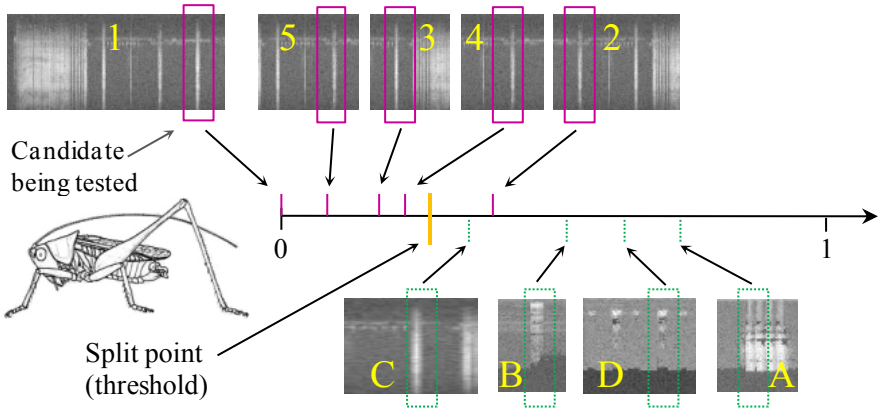

Figure 2.2: A candidate sound fingerprint, the boxed region in spectrogram 1, is evaluated by finding its nearest neighbor subsequence within both ***P*** and the four representatives of ***U*** and then sorting all objects on a number line.

We can concretely illustrate the definition of a sound fingerprint using the example shown in Figure 2.2. Note that there are a total of nine objects, five from ***P***, and four from ***U***. This gives us the entropy for the unsorted data of:

$$[-(5/9)\log(5/9)-(4/9)\log(4/9)] = 0.991$$

If we used the split point shown by the yellow/bold vertical bar in Figure 2.2, then four objects from ***P*** are the only four objects on the left side of the split point. Of the five objects to the right of the split point we have four objects from ***U*** and just one from ***P***. This gives us an entropy of:

$$(4/9)[-(4/4)\log(4/4)]+(5/9)[-(4/5)\log(4/5)-(1/5)\log(1/5)] = 0.401$$

Thus, we have an information gain of 0.590 = 0.991-0.401. Note that our algorithm will calculate the information gain many times as it searches through the candidate space, and ties are very likely. Thus, we must define a tie-breaking policy. Here we have several

options. The intuition is that we want to produce the maximum separation ("margin") between the two classes. We could measure this margin by the absolute distance between the *rightmost* positive and the *leftmost* universe distances. However, this measure would be very brittle to a single mislabeled example. To be more robust to this possibility (which frequently occurs in our data) we define the margin as the absolute distance between the *medians* of two classes. Figure 2.3 illustrates this idea.

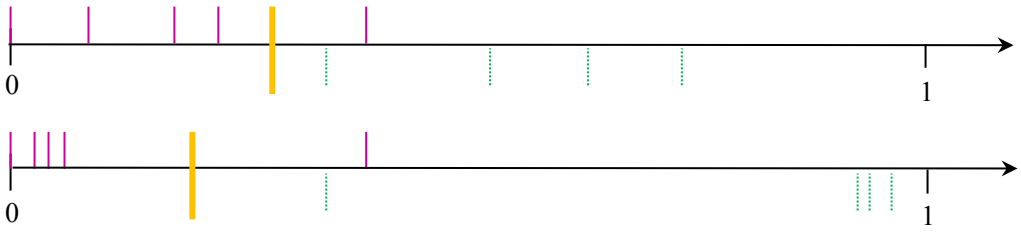

Figure 2.3: Two order lines that have the same information gain. Our tie-breaking policy reflects the intuition that the *top* line achieves less separation than the bottom line.

Even though these two fingerprints have the same information gain of 0.590 as the example shown in Figure 2.2, the bottom one is preferable, because it achieves a larger margin between ***P*** and ***U***.

Before moving on, we preempt a possible question from the reader. Why optimize the information gain, rather than just optimizing the tie-breaking function itself? The answer is twofold. Just optimizing the tie- breaking function allows pathological solutions that do not generalize well. More critically, as we shall see later, optimizing the information gain will allow admissible pruning techniques that can make our algorithm two orders of magnitude faster.

2.3 Sound Fingerprints

As the dendrogram we will later show in Figure 2.5 hints at, the CK measure can be very accurate in matching (carefully extracted) examples of animal sounds. However, our task at hand is much more difficult than this. We do *not* have carefully extracted prototypes for each class, and we do not have a classification problem where every sound must correspond to some animal we have previously observed.

Rather, for each species we have a collection of sound files which contain within them one or more occurrences of a sound produced by the target. We do not know how long the animal call is, or how many occurrences of it appear in each file. Moreover, since most of the recordings are made in the wild, we must live with the possibility that some of the sound files are "contaminated" with other sounds. For example, a twenty-second recording of a frog we examined also contains a few seconds of Strigiform (owl) calls and several cricket chirps.

In addition, as we later use our sound fingerprints to monitor audio streams we must generally expect that the vast majority of sounds are not created by any of the target species, and thus we have a large amount of data that *could* produce false positives.

2.3.1 The Intuition of Sound Fingerprints

We begin by expanding on the intuition behind sound fingerprints. For ease of exposition we will give examples using discrete text strings, but the reader will appreciate that we are really interested in streams of real-valued sounds. Assume we are giving a set of three

observations that correspond to a particular species, let us say *Maua affinis* (a cicada from South West Asia):

$$Ma = \{\text{rrbbcxcfbb, rrbbfcxc, rrbbrrbbcxcbcxcf}\}$$

We are also given access to the universe of sounds that are known *not* to contain examples of a *Maua affinis*.

$$\neg Ma = \{\text{rfcbc, crrbbrcb, rcbbxc, rbcxrf,..,rcc}\}$$

In practice, the universe set may be so large that we will just examine a small fraction of it, perhaps just sounds that are likely to be encountered and could be confused for the target insect. Our task is to monitor an audio stream (or examine a large offline archive) and flags any occurrences of the insect of interest.

Clearly it would be quite naive to examine the data for *exact* occurrences of the three positive examples we have been shown, even under a suitably flexible measure such as edit distance. Our positively labeled data is only guaranteed to have one or more samples of the insect call, and it may have additional sections of sounds from other animals or anthropogenic sounds before and/or after it.

Instead, we can examine the strings for shorter substrings that seem diagnostic of the insect. The first candidate template that appears promising is T_1 = **rrbb**, which appears in every *Ma* insect example. However, this substring also appears in the second example in ¬*Ma*, in c**rrbb**rcb, and thus this pattern is not unique to *Maua affinis.*

We could try to *specialize* the substring by making it longer; if we use T_2 = **rrbbc**, this does not appear in ¬*Ma*, removing that false positive. However, **rrbbc** only appears in two out of three examples in *Ma*, so using it would incur a risk of false negatives. As

it happens, the substring template T_3 = **cxc** *does* appear in all examples in *Ma* at least once, and never in ¬*Ma*, and is thus the best candidate for a prototypical template for the class.

As the reader may appreciate, the problem at hand is *significantly* more difficult that this toy example. First, because we are dealing with real-value data we cannot do simple tests for equality; rather, we must also learn an accept/reject threshold for the template. Moreover, we generally cannot be sure that every example in the positive class really has one true high-quality example call from the target animal. Some examples could be mislabeled, of very low quality, or simply atypical of the species for some reason. Furthermore, we cannot be completely sure that ***U*** does not contain any example from ***P***. Finally, because strings are discrete, we only have to test all possible substrings of length one, then of length two, etc, up to the length of the shortest string in the target class. However, in the real-valued domain in which we must work, the search space is *immensely* larger. We may have recordings that are minutes in length, sampled at 44,100Hz.

Thus far we have considered this problem abstractly: is it really the case that small amounts of spurious sounds can dwarf the similarity of related sounds? To see this we took six pairs of recording of various *Orthoptera* and visually determined and extracted one-second similar regions. The group average hierarchical clustering of the twelve snippets is shown in Figure 2.4.

The results are very disappointing, given that only one pair of sounds is correctly grouped together, in spite of the fact that human observers can do much better.

We believe this result is exactly analogous to the situation elucidated above with strings. Just as **rrbbcxcfbb** must be stripped of its spurious prefix and suffix to reveal **cxc**, the pattern that is actually indicative of the class, so too must we crop the irrelevant left and right edges of the spectrograms.

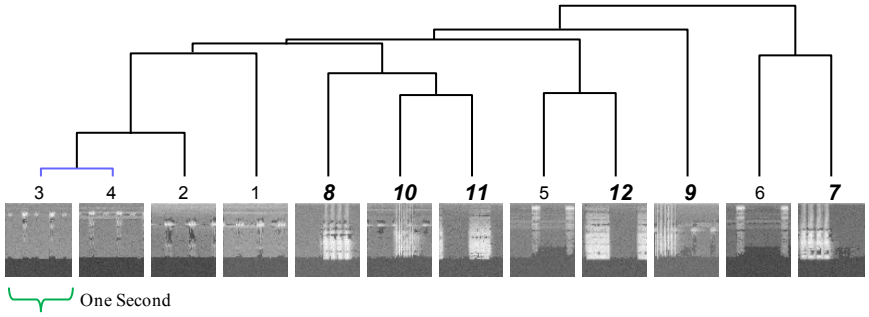

Figure 2.4: A clustering of six pairs of one-second recordings of various katydids and crickets using the CK texture measure. Only one species pair {3,4} is correctly grouped. Ideally the pairs {1,2}, {5,6}, {7,8}, {9,10} and {11,12} should also be grouped together.

For the moment, let us do this by hand. As the resulting images may be of different lengths, we have to slightly redefine the distance measure. To compute the distance between two images of different lengths, we slide the shorter one along the longer one (i.e. definition 2.5), and report the minimal distance. Figure 2.5 shows the resulting clustering.

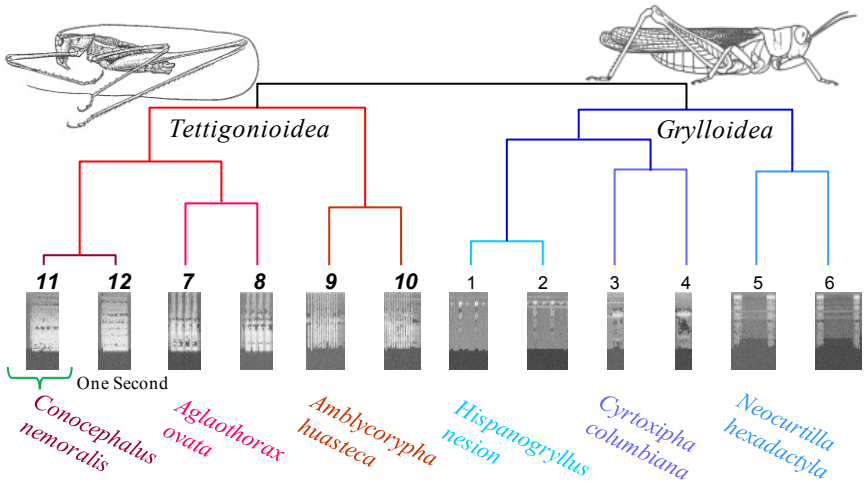

Figure 2.5: A clustering of the same data used in Figure 2.4, after trimming irrelevant prefix and suffix data. All pairs are correctly grouped, and at a higher level the dendrogram separates katydids and crickets.

The trimming of spurious data produces a dramatic improvement. However, it required careful human inspection. In the next section we will show our novel algorithm which can do this automatically.

2.3.2 Formal Problem Statement and Assumptions

Informally, we wish to find a snippet of sound that is most representative of a species, on the assumption that we can use this snippet as a template to recognize future occurrences of that species. Since we cannot know the exact nature of the future data we must monitor, we will create a dataset which contains representatives of ***U***, non-target species sounds.

Given this heterogeneous dataset ***U***, and dataset ***P*** which contains only examples from the "positive" species class, our task reduces to finding a subsequence of one of the

objects in ***P*** which is close to at least one subsequence in *each* element of ***P***, but far from all subsequences in *every* element of ***U***. Recall that Figure 2.2 shows a visual intuition of this.

This definition requires searching over a large space of possibilities. How large of a space? Suppose the dataset ***P*** contains a total of k audio sequences. Users have the option to define the minimum and maximum (L_{min}, L_{max}) length of sound fingerprint candidates. If they decline to do so we default to L_{max}= infinity and to L_{min}= 16, given that 16 by 16 is the smallest size video MPEG-1 is defined for [24]. Assume for the moment that the following relationship is true:

$$L_{max} \leq min(M_i)$$

That is to say, the longest sound fingerprint is no longer than the shortest object in ***P***, where M_i is the length of S_i from ***P***, $1 \leq i \leq k$.

The total number of sound fingerprint candidates of all possible lengths is then:

$$\sum_{l=L_{\min}}^{L_{\max}} \sum_{S_i \in \{P\}} (M_i - l + 1)$$

where l is a fixed length of a candidate. It may appear that we must test every integer pixel size from L_{min} to L_{max}; however, we know that the "block size" of MPEG-1 [24] in a CK measure is eight-by-eight pixels, and pixels remaining after tiling the image with eight-by-eight blocks are essentially ignored. Thus, there is no point in testing non-multiples of eight image sizes. As a result, the above expression can be modified to the one below:

$$\sum_{l=L_{\min}+8(i-1)}^{L_{\max}} \sum_{S_i \in \{P\}} (M_i - l + 1), i = 1, 2, ..., \lceil (L_{\max} - L_{\min}) / 8 \rceil$$

While this observation means we can reduce the search space by a factor of eight, there is still a *huge* search space that will require careful optimization to allow exploration in reasonable time.

For concreteness, let us consider the following small dataset, which we will also use as a running example to explain our search algorithms in the following sections. We created a small dataset with ***P*** containing ten two-second sound files from *Atlanticus dorsalis* (Gray shieldback), and ***U*** containing ten two-second sound files from other random insects. If we *just* consider fingerprints of length 16 (i.e. L_{min}= L_{max} = 16), then even in this tiny dataset there are 830 candidate fingerprints to be tested, requiring 1,377,800 calls to the CK distance function.

2.3.3 A Brute-Force Algorithm

For ease of exposition, we begin by describing the brute force algorithm for finding the sound fingerprint for a given species and later consider some techniques to speed this algorithm up.

The brute force algorithm is described in Algorithm 2.1. We are given a dataset **D**, in which each audio sequence is labeled either class ***P*** or class ***U***, and a user defined length L_{min} to L_{max} (optional: we default to the range sixteen to infinity).

The algorithm begins by initializing *bsf_Gain*, a variable to track the best candidate encountered thus far, to zero in line 1. Then all possible sound fingerprint candidates $S_{k,l}$ for all legal subsequence lengths are generated in the nested loops in lines 2, 4, and 5 of the algorithm.

As each candidate is generated, the algorithm checks how well each candidate $S_{k,l}$ can be used to separate objects into class ***P*** and class ***U*** (lines 2 to 9), as illustrated in Figure 2.2. To achieve this, in line 6 the algorithm calls the subroutine *CheckCandidates*() to compute the information gain for each possible candidate. If the information gain is larger than the current value of *bsf_Gain*, the algorithm updates the *bsf_Gain* and the corresponding sound fingerprint in lines 7 to 9. The candidate checking subroutine is outlined in the algorithm shown in Algorithm 2.2.

Algorithm 2.1 *SoundFPDiscovery*(D, L_{min}, L_{max})

Require: A dataset D (P and U) of audio sequence's spectrogram, user defined minimum length and maximum length of sound fingerprint.
Ensure: Return the *sound fingerprint*.

```
 1: bsf_Gain ← 0
 2: for i ← 1 to |P| do{every spectrogram in P}
 3:     S ← P_i
 4:     for l ← L_min to L_max  do {every possible length}
 5:         for k ← 1 to |S| - l + 1 do {every start position}
 6:             gain ← CheckCandidates(D, S_{k,l})
 7:             if gain > bsf_Gain then
 8:                 bsf_Gain ← gain
 9:                 bsfFingerprint ← S_{k,l}
10: return bsfFingerprint
```

In the subroutine *CheckCandidates*(), shown in Algorithm 2.2, we compute the *order line* ***L*** according to the distance from the *audio sequence* to the candidate computed in *minCKdist*() procedure, which is shown in Algorithm 2.3. In essence, this is the procedure illustrated in Figure 2.2. Given ***L***, we can find the optimal split point (definition 2.8) in lines 10 to 15 by calculating all possible splitting points and recording the best.

While the splitting point can be any point on the positive real number line, we note that the information gain cannot change in the region between any two adjacent points. Thus, we can exploit this fact to produce a finite set of possible split positions. In particular, we need only test |**D**|-1 locations.

In the subroutine *CheckCandidates*() this is achieved by only checking the mean value (the "halfway point") of each pair of adjacent points in the distance ordering as the possible positions for the split point. In *CheckCandidates*(), we call the subroutine *minCKdist*() to find the best matching subsequence for a given candidate under consideration.

Algorithm 2.2 *CheckCandidates*(*D* or *Dist*, *candidate* $S_{k,l}$)

Require: A dataset *D* of spectrogram (or distance ordering), *sound fingerprint* candidate $S_{k,l}$.
Ensure: Information Gain *gain*.

```
1:  L ← Ø
2:  if first input is D
3:      for j ← 1 to |D| do {compute distance of every spectrogram to the candidate
        sound fingerprint S_{k,l}}
4:          dist ← minCKdist(D_j, S_{k,l})
5:          insert D_j into L by the key dist
6:  else
7:      dist ← Dist
8:  I(D) ← new information gain after split computed by def' 7
9:  for split ← 1 to |D|-1 do
10:     Count N_1, N_2 for both the partitions
11:     I'(D).split ← new information gain after split computed by def' 7
12:     gain(D) = max(I(D) – I'(D).split)
13: return gain(D)
```

We do this for every spectrogram in **D**, including the one from which the candidate was culled. This explains why in each order line at least one subsequence is at zero (c.f. Figure 2.2 and Figure 2.7). In *minCKdist*() (Algorithm 2.3), we use the CK measure [24]

as the distance measurement between a candidate fingerprint and a generally much longer spectrogram.

Algorithm 2.3 *minCKdist*(D_j,*candidate* $S_{k,l}$)

Require: A audio sequence's spectrogram D_j, *sound fingerprint* candidate $S_{k,l}$.
Ensure: Return the minimum *distance* computed by CK.

```
1: minDist ← Infinity
2: for i ← 1 to | D_{j,i} | - |S_{k,l}| + 1 do {every start position}
3:     CKdist ← CK(D_{j,i} ,S_{k,l})
4:     if CKdist < minDist
5:         minDist ← CKdist
6: return minDist
```

In Figure 2.6 we show a trace of the brute force algorithm on the *Atlanticus dorsalis* problem.

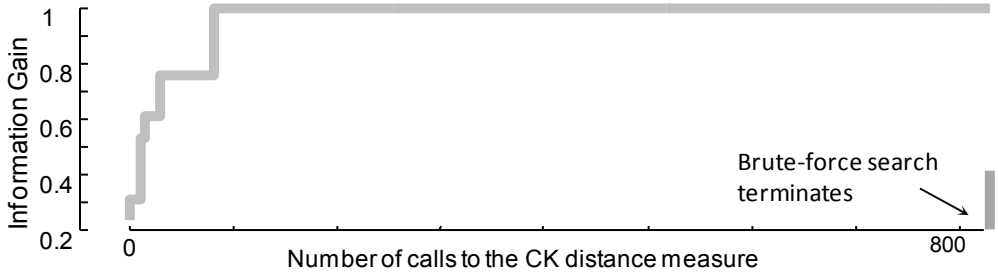

Figure 2.6: A trace of value of the *bsf_Gain* variable during brute force search on the *Atlanticus dorsalis* dataset. Only sound fingerprints of length 16 are considered here for simplicity.

Note that the search continues even after an information gain of one is achieved in order to break ties. The 1,377,800 calls to the CK function dominate the overall cost of the search algorithm (99% of the CPU time is spent on this) and require approximately 8 hours. This is not an unreasonable amount of time, considering the several days of effort needed for an entomologist to collect the data in the field. However, this is a tiny dataset.

We wish to examine datasets that are orders of magnitude larger. Thus, in the next section we consider speedup techniques.

2.3.4 Admissible Entropy Pruning

The most expensive computation in the brute force search algorithm is obtaining the distances between the candidates and their nearest matching subsequences in each of the objects in the dataset. The information gain computations (including the tie breaking computations) are inconsequential in comparison. Therefore, our intuition in speeding up the brute force algorithm is to eliminate as many distance computations as possible.

Recall that in our algorithm, we have to obtain the annotated linear ordering of all the candidates in ***P***. As we are incrementally doing this, we may notice that a particular candidate looks very unpromising. Perhaps when we are measuring the distance from the current candidate to the first object in ***U*** we find that it is a *small* number (recall that we want the distances to ***P*** to be small and to ***U*** large), and when we measure the distance to the next object in ***U*** we again find it to be small. Must we continue to keep testing this unpromising candidate? Fortunately, the answer may be "no". Under some circumstances we can admissibly prune unpromising fingerprints; without having to check all the objects in the universe ***U***.

The key observation is that we can cheaply compute the *upper bound* of the current partially computed linear ordering at any time. If the *upper bound* we obtain is less than the *best-so-far* information gain (i.e. the ***bsf_Gain*** of Algorithm 2.1), we can simply

eliminate the remaining distance computations in ***U*** and prune this particular fingerprint candidate from consideration.

To illustrate this pruning policy, we consider a concrete example. Suppose that during a search the *best-so-far* information gain is currently 0.590 and we are incrementally beginning to compute the sound fingerprint shown in Figure 2.2. Assume that the partially computed linear ordering is shown in Figure 2.7. We have computed the distances to all five objects in ***P***, and to the first two objects in ***U***.

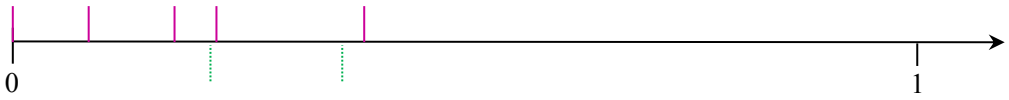

Figure 2.7: The order line of all the objects in ***P*** and just the first two objects in ***U***.

Is it possible that this candidate will yield a score better than our *best-so-far*? It is easy to see that the most optimistic case (i.e., the upper bound) occurs if all of the remaining objects in ***U*** map to the far right, as we illustrate in Figure 2.8.

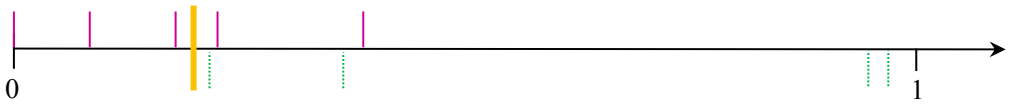

Figure 2.8: The logically best possible order line based on the distances that have been calculated in Figure 2.7. The best split point is shown by the yellow/heavy line.

Note that of the three objects on the left side of the split point, all three are from ***P***. Of the six objects on the right side, two are from ***P*** and four are from ***U***. Given this, the entropy of the hypothetical order line shown in Figure 2.8 is:

$$(3/9)[-(3/3)\log(3/3)]+(6/9)[-(4/6)\log(4/6)-(2/6)\log(2/6)] = 0.612$$

Therefore, the best possible information gain we could obtain from the example shown in Figure 2.7 is just 0.612, which is lower than the *best-so-far* information gain. In this case, we do not have to consider the ordering of the remaining objects in ***U***. In this toy example we have only pruned two invocations of the *CheckCandidates*() subroutine shown in Algorithm 2.2. However, as we shall see, this simple idea can prune more than 95% of the calculations for more realistic problems.

The formal algorithm of admissible entropy pruning is shown in Algorithm 2.4. After the very first sound fingerprint candidate check, for all the remaining candidates, we can simply insert *EntropyUBPrune*() in line 4 of Algorithm 2.2, and eliminate the remaining CK distance and information gain computation if the current candidate satisfies the pruning condition, as we discussed in this section. *EntropyUBPrune*() takes the *best-so-far* information gain, current distance ordering from class ***P*** and class ***U***, and remaining objects in ***U***, and returns the fraction of the distance measurements computed to see how much elimination we achieved.

Algorithm 2.4 *EntropyUBPrune* (U_m, *currentDist*, $S_{k,l}$, *bsf_Gain*)

Require: A audio sequence's spectrogram U_m, current distance ordering, *sound fingerprint* candidate $S_{k,l}$, best-so-far information gain.
Ensure: Return fraction of distance computations in U.

```
1:  fraction ← 0
2:  counter ← 0
3:  rightmostDist ← largest distance value in currentDist + 1
4:  bestDist ← Add rightmostDist for U_m to currentDist
5:  gain ← CheckCandidates(bestDist, S_{k,l})
6:  if gain > bsf_Gain
7:      return false and increment counter
8:  else
9:      return true
10: return fraction ← counter/|U|, gain
```

We can get a hint as to the utility of this optimization by revisiting the *Atlanticus dorsalis* problem we considered above. Figure 2.9 shows the difference entropy pruning makes in this problem.

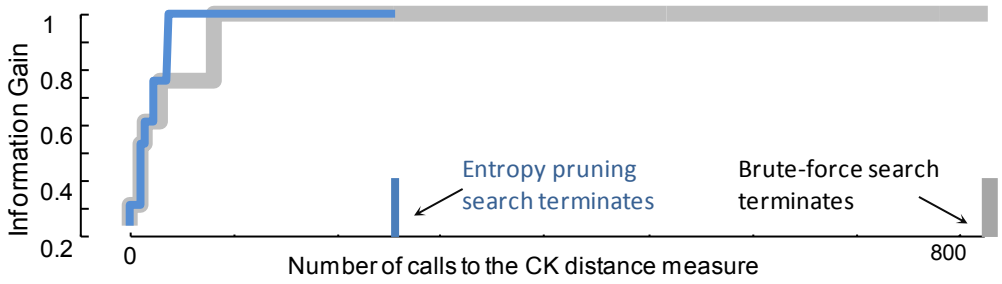

Figure 2.9: A trace of the *bsf_Gain* variable during brute force and entropy pruning search on the *Atlanticus dorsalis* dataset.

Note that not only does the algorithm terminate earlier (with the exact same answer), but it *converges* faster, a useful property if we wish to consider the algorithm in an anytime framework [92].

2.3.5 Euclidean Distance Ordering Heuristic

In both the brute force algorithm and the entropy-based pruning extension introduced in the last section, we generate and test candidates; from left to right; and top to bottom based on the given lexical order of the objects' label (i.e., the file names used by the entomologist).

There are clearly other possible orders we could use to search, and it is equally clear that for entropy-based pruning, some orders are better than others. In particular, if we find a candidate which has a relatively high information gain early in the search, our pruning strategy can prune much more effectively.

However, this idea appears to open a "*chicken and egg*" paradox. How can we know the best order; until we have finished the search? Clearly, we cannot. However, we do not need to find the *optimal* ordering; we just need to encounter a relatively good candidate relatively early in the search. Algorithm 2.5 outlines our idea to achieve this. We simply run the entire brute force search using the Euclidean distance as a proxy for the CK distance, and sort the candidates based on the information gain achieved using the *Euclidean distance*.

Concretely, we can insert *EuclideanOrder*() between lines 4 and 5 in Algorithm 2.1 to obtain a better ordering to check all the candidates.

Algorithm 2.5 *EuclideanOrder* (*D*, *minLen*, *maxLen*)

Require: A dataset *D* (*P* and *U*) of audio sequence's spectrogram, user defined minimum/maximum length of *sound fingerprint*.
Ensure: Return the *new order* of candidates.
1: Replace CK measure with Euclidean distance measure
2: *newGain* ← *CheckCandidates* (*D* or *Dist*, candidate $S_{k,l}$)
3: *newOrder* ← sort the candidates by decreasing *newGain*
4: **return** *newOrder*

Running this preprocessing step adds some overhead; however, it is inconsequential because the Euclidean distance is at least two orders of magnitude faster than the CK distance calculation. For this idea to work well, the Euclidean distance must be a good proxy for the CK distance calculation. To see if this is the case, we randomly extracted 1,225 pairs of insect sounds and measured the distance between them under both measures, using the two values to plot points in a 2D scatter plot, as shown in Figure 2.10. The results suggest that Euclidean distance is a very good surrogate for CK distance.

To measure the effect of this reordering heuristic we revisited our running example shown in Figure 2.6/Figure 2.9.

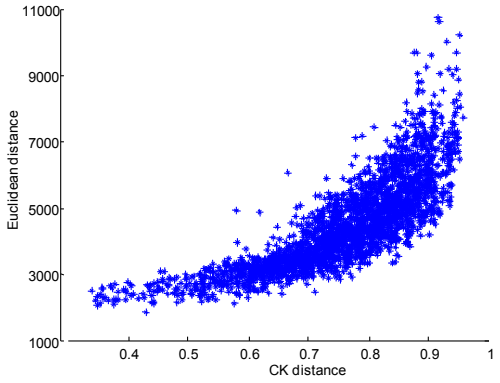

Figure 2.10: The relationship between Euclidean and CK distance for 1,225 pairs of spectrograms.

The Euclidean distance reordering heuristic is shown in Figure 2.11.

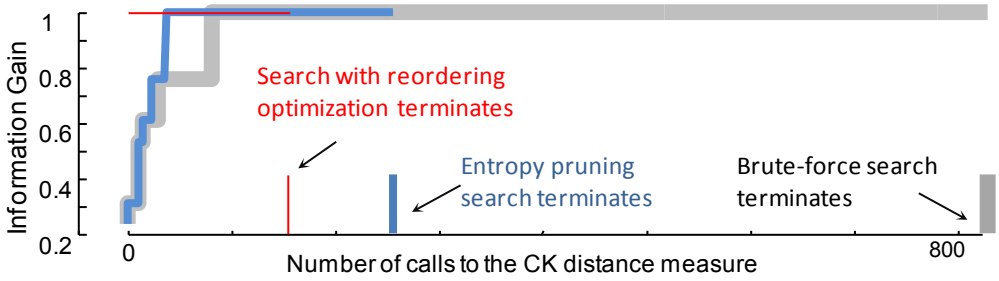

Figure 2.11: A trace of value of the *bsf_Gain* variable during brute force, entropy pruning, and reordering optimized search on the *Atlanticus dorsalis* dataset.

As we can see, our heuristic has two positive effects. First, the absolute time to finish (with the identical answer as a brute force search) has significantly decreased. Secondly, we converge on high quality solution faster. This is a significant advantage if we wanted to cast the search problem as an anytime algorithm [92]. As impressive as the speedup

results are, as we shall show in the next section, they are pessimistic due to the small size of our toy problem.

2.3.6 Disjunctive Sound Fingerprints

A single species of insect can produce a variety of sounds for different functions, such as attracting mates, claiming territory, repelling aggressors, etc. In addition, their sounds can be affected by external factors such as ambient temperature [36][52]. Thus far, we have only considered using a *single* sound fingerprint to represent a class. However, it is easy to imagine situations where it may not be possible to use a *single* sound fingerprint to separate classes.

To illustrate this, we consider a toy example as shown in Figure 2.12.*left*. We can easily differentiate *Species A* from *Species B* with a *single* sound fingerprint A. In particular, every object within threshold T_A belongs to *Species A*, otherwise, it belongs to *Species B*. In contrast, as we show in Figure 2.12.*center*, we may have a situation in which both the males and females of a species sing (In most Orthoptera only the males sing, but in some cases, such as the *Magicicada septendecim*, the females also sing [36]), as in *Species C* in our toy example. Moreover, as insects are often sexually dimorphic [36], these songs may be rather different.

In that case, *Species C* contains two well-defined distributions, if we attempt to use only one sound fingerprint to separate *Species C* from *Species D*, almost half of the instances could be misclassified into *Species D* as shown in Figure 2.12.*center*.Note that if we increase the threshold (i.e. expand the radius of the gray circle), we will capture

more true positives only at the expense of capturing more false positives (labeling *Species D* as into *Species C*).

How can we solve this non-linear separation problem? Inspired by the similar nearest centroid algorithm [64], we propose to use multiple fingerprints to represent each species (when necessary). As shown in Figure 2.12.*right*, we can use *multiple* fingerprints (C_1 and C_2) to represent the single concept of *Species C*.

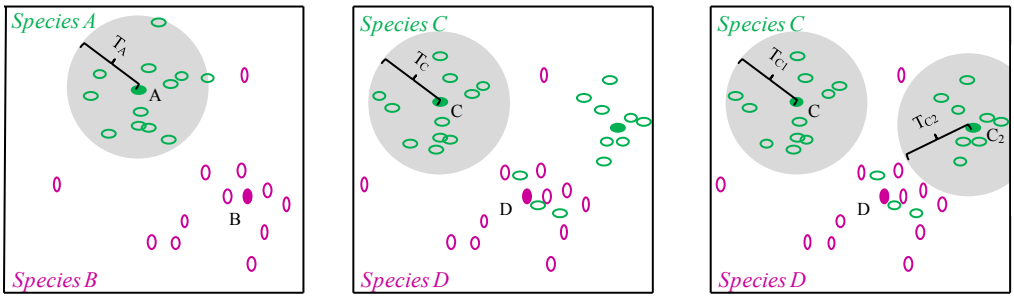

Figure 2.12: *left*) A toy problem showing instances of crickets (*Species A*) and katydids (*Species B*). In this case a single sound fingerprint A can separate the two classes. *center*) In contrast, *Species C* is an example where a single sound fingerprint cannot separate the classes, perhaps because the insects are sexually dimorphic with the males and females singing differently. *right*) Two sound fingerprints C_1 and C_2 connected by an *OR* operation *can* better separate the classes.

The intuition behind this is that we can augment the representational power of sound fingerprints with the logical **OR** operator. Intuitively, we say that a candidate sound is of class X, if it is similar to *sound fingerprint*$_1$ **OR** *sound fingerprint*$_2$ …**OR** *sound fingerprint*$_k$. We call such a representation *Disjunctive Sound Fingerprints*. Each sound fingerprint may correspond to a different behavior for the animal, e.g., mating calls, aggressive utterances, mother-pup interactions, etc. In addition, for insects, the multiple sound fingerprints may correspond to different external conditions which influence their

sound production apparatus. For instance, for some insects, especially in *Orthoptera*, it is well known that the external temperature greatly affects the sounds the insects produce [36]. Thus, in this case, three or four sound fingerprints could represent the sound a particular insect produces at, say 10 to 20 degrees, 21 to 30 degrees, etc. While we have some such domain knowledge, it is important to note that we do not propose to "hand-craft" the disjunctive sound fingerprints. We wish to preserve the black-box nature of our algorithms.

During the search process, we wish to avoid finding *redundant* fingerprints, that is to say, minor-variations of previously discovered sound fingerprints. To achieve this, objects that are already "explained" (i.e., correctly classified) by a fingerprint are removed before rerunning the search on a now smaller set of data objects. This continues until either we cannot find any sound fingerprint to separate ***P*** from ***U*** or there are less than two objects left in ***P***.

For instance, in Figure 2.12.*center*, our algorithm learned C_1 as the first sound fingerprint, however, almost half of the objects are still not classified correctly. Thus a second sound fingerprint C_2 is learned after removing objects which can be described by C_1, as shown in Figure 2.12.*right*. However, there are still three remaining objects, which are not encompassed by the gray circles as shown in Figure 2.12.*right*. We can keep searching for another sound fingerprint, but this opens the possibility of overfitting, given there is so little data remaining. If these three objects cannot be described by another sound fingerprint that separates them from *Species D*, they are unclassified and the search for sound fingerprints of *Species C* is terminated. Thus, our algorithm finds between zero

to K sound fingerprints for a given class. Finding zero fingerprints signals the user that the dataset cannot be represented by a single sound fingerprint and has no exploitable structure.

The formal algorithm to learn multiple fingerprints is shown in Algorithm 2.6.

In D*isjuntFPDiscovery*(), we call the *SoundFPDiscovery*() procedure with entropy-based pruning and Euclidean distance ordering heuristic speed up techniques to find the first sound fingerprint (line 2). When the first sound fingerprint is found, we check to see if $\boldsymbol{P}$ has at least two objects to the right of the distance threshold, because that means using a single sound fingerprint cannot represent $\boldsymbol{P}$ sufficiently to separate it from $\boldsymbol{U}$.

Given the CK distance ordering *currentDist* and the *threshold*, if there is more than *one* object of $\boldsymbol{P}$ with distances larger than the threshold, another sound fingerprint is required to learn to represent these objects. After finding the new sound fingerprint we test the information gain to see if it can separate the two classes further. If the separation improves, we look for a new $\boldsymbol{P}_j$ and continue as before. Once there are less than two objects left in $\boldsymbol{P}$ or we cannot achieve better separation, we terminate *SoundFPDiscovery*() routine and return sound fingerprint(s).

Algorithm 2.6 *DisjuntFPDiscovery* (D, L_{min}, L_{max})

Require: A dataset D (P and U) of audio sequence's sonogram, user defined minimum length and maximum length of sound fingerprint.
Ensure: Return a set of sound fingerprints(can be an empty set).

```
1:  for index = 1 to |P|+|U|-1
2:      soundFP, currentDist, threshold, bsf_infoGain = SoundFPDiscovery(D, L_min,
        L_max) with speedup techniques
3:  for j = 1 to |currentDist|
4:      if currentDist(j) > threshold
5:          newP = P_j
6:      if |newP|>1 and |P_j|-|newP|>1 and bsf_infoGain will not increase {number of
        objects in newP, number of objects whose distance to the current fingerprint less
        than threshold should larger than 1}
7:          recursive call SoundFPDiscovery(newP&U, L_min, L_max)
8:          increment index
9:          append new sound fingerprint to soundFP
10:     else
11:         break
12: return soundFP
```

2.4 Experimental Evaluation

We have created a supporting webpage [5], which contains all code/data used in this work. Moreover, the webpage contains addition experiments, along with videos and sounds files from [3] that allow the interested reader to get a better appreciation of the scale and complexity of the data we are working with.

2.4.1 CK as a Tool for Taxonomy

We begin by noting that beyond the utility of our ideas for monitoring wildlife, the CK measure may be useful as a taxonomic tool. Consider the insect shown in Figure 2.13. As noted in a National Geographic article, "*the sand field cricket (Gryllus firmus) and the southeastern field cricket (Gryllus rubens) look nearly identical and inhabit the same*

geographical areas" [4]. Thus, even if handling a living specimen, most entomologists could not tell them apart without resorting to DNA analysis.

We suspected that we might be able to tell them apart by sound[4]. While we do not have enough data to do forceful and statistically significant experiments, we can do two tentative tests. As shown in Figure 2.13, we projected twenty-four examples from the two species into two-dimensional space using multi-dimensional scaling, and we also clustered eight random examples, four from each class.

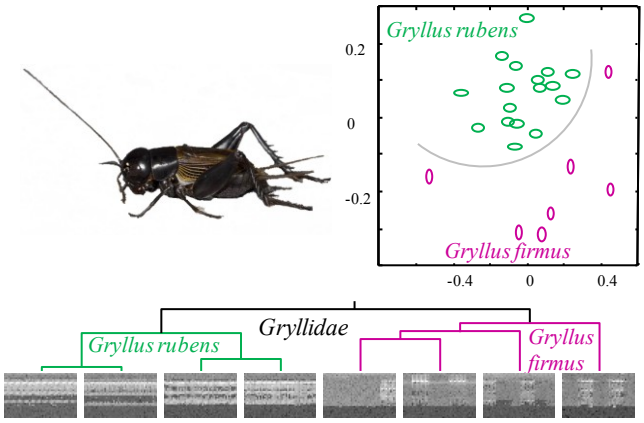

Figure 2.13: *top left*) An insect found in Florida: is it a *G. rubens* or *G. firmus*? *top right*) Projecting one-second snippets of songs from both insects into 2D space suggests they are almost linearly separable, a possibility reflected by their clusterability (*bottom*).

The results suggest that these congeneric[5] species are almost linearly separable in two-dimensional space (they *are* linearly separable in three-dimensional space).

[4] In brief, it is well known that the acoustic behavior of insects is important in insect speciation, especially for sympatric speciation, where new species evolve from a single ancestral species while inhabiting the same geographic region [91].

[5] Species belonging to the same genus are *congeneric*.

2.4.2 Insect Classification

There are currently no benchmark problems for insect classification. Existing datasets are either too small to make robust claims about accuracy, or were created by authors unwilling to share their data. To redress this we created and placed into the public domain a large classification dataset [5]. The data consists of twenty species of insects, eight of which are *Gryllidae* (crickets) and twelve of which are *Tettigoniidae* (katydids)[6]. Thus, we can treat the problem as either a twenty-class *species* level problem, or two-class *genus* level problem. For each class we have ten training and ten testing examples. It is important to note that we assembled these datasets *before* attempting classification, explicitly to avoid cherry-picking. Note that because of convergent evolution, mimicry and the significant amounts of noise in the data (which were collected in the field) we should not expect perfect accuracy here. Moreover, this group of insects requires some *very* subtle distinctions to be made; for example, *Neoconocephalus bivocatus*, *Neoconocephalus retusus*, and *Neoconocephalus maxillosus* are obviously in the same genus, and are *visually* indistinguishable at least to our untrained eye. Likewise, we have multiple representatives from both the *Belocephalus* and *Atlanticus* genera.

We learned twenty sound fingerprints using the algorithm in Section 2.3. We then predicted the testing exemplars class label by sliding each fingerprint across it and recording the fingerprint that produced the minimum value as the exemplar's nearest neighbor. The classification accuracies are shown in Table 2.2.

[6] A full description of the data is at [5].

	species-level problem		genus-level problem	
	default rate	fingerprint	default rate	fingerprint
10 species	0.10	0.70	0.70	0.93
20 species	0.05	0.44	0.60	0.77

Table 2.2: Insect Classification Accuracy.

The results are generally impressive. For example, in the ten-class species-level problem the default accuracy rate is only 10%, but we can achieve 70%. It is worth recalling the following when considering these results.

- The testing data does not consist of carefully extracted *single* utterances of an insect's call. Rather, it consists of one or two-minute sound files known to contain at least one call, together with human voice annotations and miscellaneous environmental sounds that can confuse the classification algorithm.
- As noted above, our dataset has multiple congeneric species; that, at least to our eyes and ears, look and sound identical. This is an *intrinsically hard* problem.
- The reader can be assured that the results are not due to overfitting, because we did not *fit* any parameters in this experiment. These are "black box" results.
- We can do a little better by weighting the nearest neighbor information with the threshold information (which we ignore in the above). Since this *does* introduce a (weighting) parameter to be tuned, in the interest of brevity, given page limits and our already excellent results, we defer such discussions to future work.

2.4.3 Monitoring with Sound Fingerprints

To test our ability to monitor an audio stream in real time for the presence of a particular species of insects, we learned the sound fingerprints for three insect species of insects native to Florida. In each case we learned from training sets consisting of ten insects.

To allow visual appreciation of our method, as shown in Figure 2.14 we produced an eight-second sequence of audio by concatenating snippets of four different species, including holdout (i.e. *not* seen in the training set) examples from our three species of interest. While each fingerprint has a different threshold, for simplicity and visual clarity we show just the averaged threshold. As we can see in Figure 2.14, this method achieves three true positives, and more remarkably, no false positives. Recall that the CK distance measure exploits the compression technique used by MPEG video encoding, which is among the most highly optimized computer code available. Thus, we can do this monitoring experiment in real time, even on an inexpensive laptop.

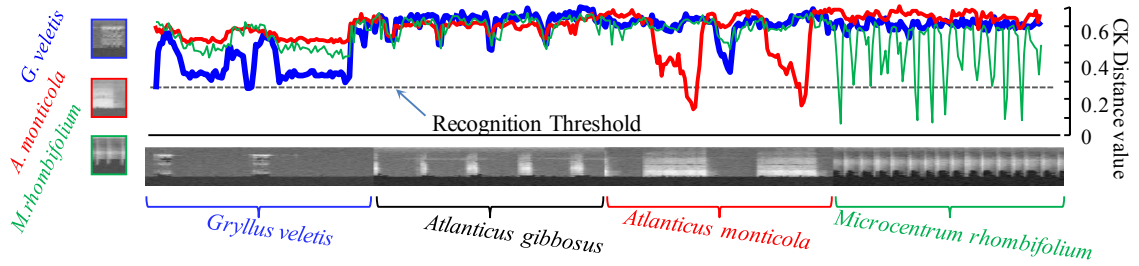

Figure 2.14: (Image best viewed in color) *far left*) Three insect sound fingerprints are used to monitor an eight- second audio clip. In each case, the fingerprint distance to the sliding window of audio dips below the threshold as the correct species sings, but not when a different species is singing.

2.4.4 Scalability of Fingerprint Discovery

Recall the experiments shown in Section 2.3, when our toy example had only ten objects in both ***P*** and ***U***. We showed a speedup of about a factor of five, although we claimed this is pessimistic because we expect to be able to prune more aggressively with larger datasets. To test this, we reran these experiments with a more realistically-sized ***U***, containing 200 objects from other insects, birds, trains, helicopters, etc. As shown in Figure 2.15, the speedup achieved by our reordering optimization algorithm is a factor of 93 in this case.

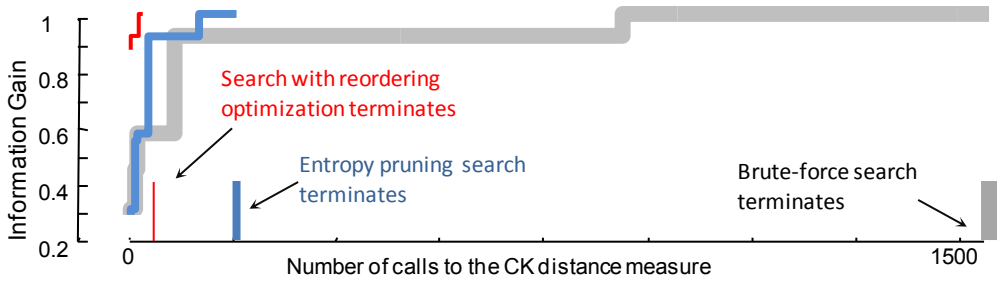

Figure 2.15: A trace of value of the *bsf_Gain* variable during brute force, entropy pruning, and reordering optimized search on the *Atlanticus dorsalis* dataset with the 200-object universe.

Similar to the experiment above, we also learned the sound fingerprints for three frog species, *Bufo alvarius*, *Bufo canorus*, and *Pseudacris crucifer*. We learned sound fingerprints from the training sets consisting of ten frogs for each species.

To allow visual appreciation of our method, as shown in Figure 2.16 we produced a fourteen-second sequence of audio by concatenating snippets of seven different species, including holdout (i.e. *not* seen in the training set) examples from our three species of interest. Again, while each fingerprint has a slightly different threshold, for simplicity

and visual clarity we show just the averaged threshold. As we can see in Figure 2.16, this method also achieves three true positives, and more significantly, no false positives.

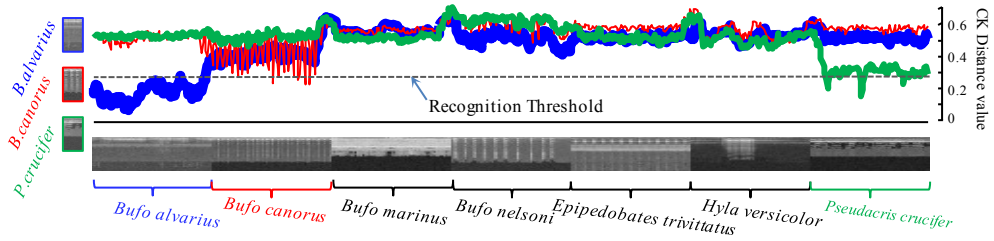

Figure 2.16: (Image best viewed in color) Three frog sound fingerprints are again used to monitor a fourteen-second audio clip on the left. In each case, the fingerprint distance to the sliding window of audio dips below the threshold as the correct species sings, but not when a different species is singing.

2.4.5 Robustness

No matter how carefully the animal audio dataset is obtained, we must resign ourselves to the possibility of mislabeled data. We claim our algorithm is robust to mislabeled data so long as the *majority* of data in ***P*** is correctly labeled. To test our algorithm's robustness to mislabeled training data, we will give a simple example using one katydid data (*Atlanticus dorsalis*).

There are ten objects in ***P*** and ten objects in ***U***; we ran our algorithm and found the sound fingerprints shown on the left of Figure 2.17.*top*. To simulate mislabeled training data, we randomly swapped two objects in ***P*** with two objects in ***U***, creating 20% mislabeled training data. We relearned sound fingerprints from this mislabeled data and discovered the sound fingerprint as shown on the left of Figure 2.17.*middle*. As in Section 2.3, we concatenated the testing data in ***P*** and ***U*** in a round-robin fashion to

obtain a forty-second long audio clip to test the monitoring accuracy of the two learned sound fingerprints.

As shown in Figure 2.17, the fingerprint learned from the correctly labeled data achieves nine (out of ten) true positives and one false negative. The fingerprint learned from the mislabeled data also achieves nine (out of ten) true positives and a single false negative, although the location of false negative is different.

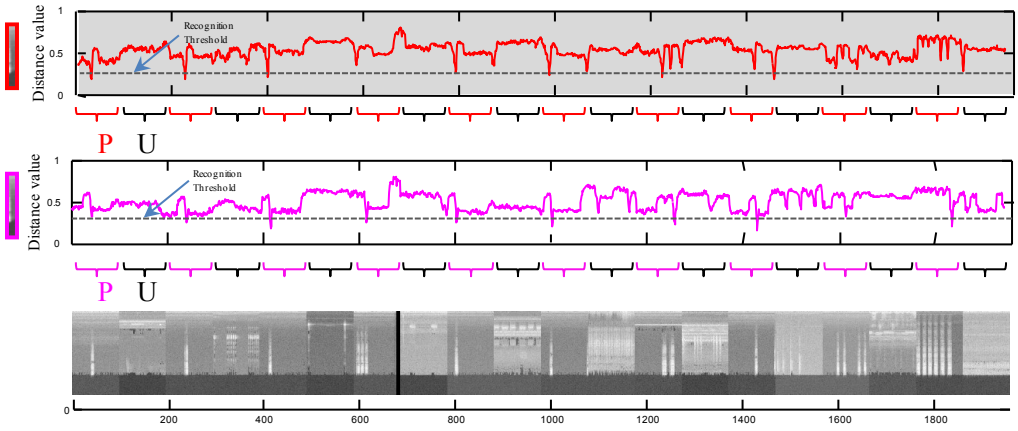

Figure 2.17: *top*, *middle*) Sound fingerprint of *Atlanticus dorsalis* is used to monitor a twenty-second testing audio clip without (with) mislabeled data in ***P***. The fingerprint distance to the sliding window of audio dips below the threshold as the correct species sings, but not when a different species is singing. *bottom*) Forty second sequence of audio by concatenating snippets of testing audio clips from ***P*** and ***U*** alternatively.

These results suggest that our algorithm is largely invariant to small amounts of mislabeled data.

2.4.6 Insect Classification with Disjunctive Sound Fingerprints

To this point, we have evaluated *single* sound fingerprint representation. We will now test the expressive power of *disjunctive* sound fingerprints. Recall the toy example in

Figure 2.12. We showed that our dataset may have multiple "clusters" of sound within a single species, and further proposed a technique to learn multiple sound fingerprints to represent the class concept. We will consider two examples to demonstrate the utility of the disjunctive sound fingerprint discovery algorithm for improving classification accuracy.

Insect Dataset I

The first dataset consists of two classes, which are Crickets (*Anaxipha litarena* and *Anurogryllus celerinictus*) and Katydids (*Amblycorypha carinata* and *Neoconocephalus bivocatus*). Note that there are intrinsically two subclasses in each class, although our algorithm is not "aware" of this.

There are twenty training and twenty testing objects for each class, all of which are two-seconds long. We ran the disjunctive sound fingerprints discovery algorithm on this problem. We find that two sound fingerprints described the class Crickets. Gratifyingly, the two sound fingerprints correspond to the two subclasses of crickets, the first came from *A.celerinictus* and the second from *A.litarena*. However, we only find one sound fingerprint to describe Katydids, presumably this is because the two subclasses of Katydids were more similar to each other.

We predicted the testing data class labels by sliding each fingerprint across each object and recording the fingerprint that produced the minimum value as the exemplar's nearest neighbor. Disjunctive sound fingerprints reduced the classification error rate from 0.325 (with a single sound fingerprint, as in Section 2.3.) to 0.175.

Insect Dataset II

The second dataset also has two classes, which we again labeled as Crickets (*A.litarena*, *A. celerinictus*, *Gryllus fultoni*, and *Velarifictorus micado*) and Katydids (*A.carinata*, *Atlanticus dorsalis*, *Belocephalus sabalis*, and *Neoconocephalus retusus*). There are intrinsically four subclasses in each class, but as before our algorithm does not have access to this information.

We have forty training and forty testing two-second instances for each class. After running the disjunctive sound fingerprint discovery algorithm in Algorithm 2.6, we find that four sound fingerprints represented the class Crickets. The first one is from *A.litarena*, the second one is from *A. celerinictus*, the third one is from *G. fultoni*, and the fourth one is from *V. micado*. However, we only find three sound fingerprints to represent Katydids. The first one is from *A.carinata*, the second one is from *N.retusus*, and the third one is from *B.sabalis*. As before we suspect that the reason is that two species *A.carinata* and *A. dorsalis* are similar to each other. Once again the disjunctive sound fingerprints reduced the classification error rate, this time from 0.45 (with a single sound fingerprint, as in Section 2.3) to 0.325.

Insect Dataset III

To test the disjunctive sound fingerprints discovery algorithm to find an appropriate number of sound fingerprints for a given class, we will provide a simple example to show that our algorithm did not find redundant fingerprints. This dataset also has two classes, Crickets (*A.litarena*) and Kaytdids (*A. dorsalis*). We have ten training and ten testing objects for each class. By visual, manual inspection alone, the diversity among each class

is not large so a single sound fingerprint should be able to represent one class. To test our prediction, we run the disjunctive sound fingerprint discovery algorithm in Algorithm 2.6. We found only one sound fingerprint for each class and the classification error rate was 0.15. The result suggests that our disjunctive sound fingerprint algorithm can both avoid overfitting and help to reduce the classification error rate.

Beyond improving the classification accuracy, disjunctive fingerprints may also be useful for exploratory data mining, telling us something about the data that would otherwise be difficult to discover.

2.5 Conclusion

In this chapter we have introduced a novel bioacoustic recognition/classification framework. We feel that unlike other work in this area, our ideas have a real chance to be adopted by domain practitioners, because our algorithm is essentially a "black box", requiring only that the expert can label some data. We have shown through extensive experiments that our method is accurate, robust and efficient enough to be used in real time in the field.

Chapter 3

Parameter-Free Audio Motif Discovery in Large Data Archives

The discovery of repeated structure, i.e. motifs/near-duplicates, is often the first step in exploratory data mining. As such, the last decade has seen extensive research efforts in motif discovery algorithms for text, DNA, time series, protein sequences, graphs, images, and video. Surprisingly, there has been less attention devoted to finding repeated patterns in audio sequences, in spite of their ubiquity in science and entertainment. While there is significant work for the special case of motifs in music, virtually all this work makes many assumptions about data (often to the point of being genre specific) and thus these algorithms do not generalize to audio sequences containing animal vocalizations, industrial processes, or a host of other domains that we may wish to explore.

In this work we introduce a novel technique for finding audio motifs. Our method does not require any domain-specific tuning and is essentially parameter-free. We demonstrate our algorithm on very diverse domains, finding audio motifs in laboratory mice vocalizations, wild animal sounds, music, and human speech. Our experiments

demonstrate that our ideas are *effective* in discovering objectively correct or subjectively plausible motifs. Moreover, we show our novel probabilistic early abandoning approach is *efficient*, being two to three orders of magnitude faster than brute- force search, and thus faster than real-time for most problems.

The rest of this chapter is organized as follows. In Section 3.1 we review related work. In Section 3.2 we introduce the necessary notation to formalize our algorithm in Section 3.3. Section 3.4 sees a detailed empirical evaluation of our ideas on diverse domains, and we offer conclusions and directions for future work in Section 3.5.

3.1 Related Work and Background

Researchers have devoted an enormous amount of effort to the acoustic analysis domain for decades, mostly focused on human speech recognition and music analysis [82]. For example, Automatic Speech Recognition (ASR) has been an active research field and that accomplished much throughout the last decades (i.e. the success of Apple's Siri). Apart from work on mitigating resource limitations (time and memory), most of research can be divided four disciplines: preprocessing, representations and similarity measurement, feature extraction, and advanced algorithms [12][42]. The performance levels of most current ASR degrade significantly in a noisy environment, which is common in real world recordings [42]. Therefore, preprocessing is a necessary for audio processing. Energy normalization (i.e. apply pre-emphasis radiation filter in order to equalize the effect of the propagation of speech though air), various signal processing filters to improve signal to noise ratio, domain transformation (discrete Fourier Transformation

(DFT), mel-scaled cepstrum coefficients (MFCC)), in which the combinations of signal and noise sources can be considered additive, therefore, relatively easier to separate them. A more detailed survey can be found [42].

As noted in Section 2.1.3, in addition to extracting features directly after domain transformation, analyzing audio streams in visual space allows people to consider domains that take advantage by mature computer vision techniques [58][95]. In speech recognition domain, there are four general features: continuous (pitch-related, formants, energy-related, timing features, etc), Qualitative (voice level, voice pitch, phrase, phoneme, temporal structures, etc), spectral (linear predictor coefficients (LPC), MFCC, Log-frequency power coefficients (LFPC)), etc. It is also common to apply feature selection and transformation afterwards, for example, dimension reduction, principle component analysis, linear discriminate analysis, etc.

Lastly, researchers typically use traditional algorithms to the extracted features, such as Hidden Markov Model (HMM), Gaussian Mixture model (GMM), neural networks, and support vector machines, or multiple classifiers techniques [12][87]. Among all these classifiers, HMM is most commonly used more advanced applications, such as speech emotion recognition, speech recognition for mobile devices, etc. [12][96].

Complex animal sounds (birds, whales, insects, etc) have been receiving more attention because they can be used to measure the health of the ecosystem and its biodiversity [40][86]. However, most of the current work simply took off-the-shelf techniques from human speech, music analysis, and focus on some specific target animal songs, requiring with huge amount of manual effort [97] and domain expertise.

Now we discuss more related work about the specific task for this thesis, finding repeated patterns in audio streams. By far the most common approach to finding repeated patterns in audio is to "*use string-matching techniques on a symbolic representation learned from the data*" [11]. Given a high quality symbolic representation of the data, the problem becomes much simpler; we can just use an off-the-shelf symbolic repeated pattern discovery tool. This approach has been used in music [11] and in mice vocalizations [95]. However, it is obvious that the symbol extraction algorithms used for pop songs are unlikely to generalize to classical music, much less mice or insects [95]. Likewise, it is not clear that the symbol extraction method discussed in [95] will generalize to other strains of mice, much less other mammals. It is difficult to overstate how poorly existing audio motifs discovery algorithms can be expected to generalize. For example, [81] introduces an algorithm that is specialized for just Hindustani vocal music compositions.

There is, however, research work in the speech recognition community that is very close in spirit to our work. For example, in a recent but highly cited paper, the authors ask how well we can do in finding repeated speech elements with "*zero resources*" [53]. By zero resources they mean that they assume no "*models or training data for the target language*." Note, however, that even here researchers assume human speech. We would like to remove even that assumption, and have a completely unsupervised, parameter-free, and zero resource algorithm that can detect repeated sounds in sources as diverse as human speech, wildlife surveillance, music, and industrial applications.

It is important to recognize that the framing of our problem precludes many *apparent* solutions. For example, there is significant work in very fast audio search for commercial music applications. Such work is often called audio thumbnailing or audio fingerprinting [26]. One might imagine that such techniques and representations could be adapted to the task at hand. However, most such methods assume that there is a "platonic ideal" sound snippet, say a master recording of a song. The instances matching this idealized template might not be bit-for-bit identical due to different encodings, or in the case of Shazam/SoundHound[7], corruption by background noise as the user records the music with a mobile device. Nevertheless, the problem reduces to matching two objects that are essentially identical, except that one has minor noise/distortions. Most critically, the two snippets are assumed to be aligned perfectly in time.

In contrast, we consider the more general case where the two similar sounds are different *physical* (not digital) instantiations of a "process". Here the process could be two bird calls, two belt slip screeches from an overloaded industrial machine [98], or two repetitions in a mouse's song (cf. Figure 1.1). Thus, we are interested in finding repetitions in the face of a much broader set of noise and distortions, including time scaling (*global* shrinking/stretching), *local* time warping, pitch shifts, and echos, etc.

The most important difference between our work and previous audio motif discovery approaches is that our algorithm finds repeated objects in audio sequences without making *any* assumptions about the intrinsic properties of the objects. For example, we did

[7] Shazam/SoundHound is commercial mobile phone based music identification services. A cell phone's built-in microphone is used to gather a brief sample of music being played. An acoustic fingerprint is created based on the sample, and is compared against a central database for a match.

not need domain knowledge of rodent physiology to find the motifs discovered in the mice vocalizations shown in Figure 1.1 [95]. For the task of *music* motif discovery, researchers have considered a huge number of possible features, including static music information (key, beat, and tempo), acoustic information (loudness, duration, pitch, bandwidth, and brightness), thematic features (melodies, rhythms, and chords), and higher-level composite features (i.e. hierarchical rules, Markov models) [49]. These features may be helpful for motif discovery, but they require a huge amount of feature engineering and there is evidence that they do not generalize across music genres [81], much less generalize to the diverse domains we consider.

Non human-produced sounds offer no fewer difficulties. For example, in [40], researchers attempt to find repeated patterns in bird songs. Their algorithm requires extracting features from syllables, and the authors bemoan the effort of human intervention: "*Syllable templates were formed by aligning and averaging four to five manually chosen clips corresponding to each syllable…*," "*…manually chosen based on visual inspection*," etc. It is exactly this kind of manual tweaking that we wish to avoid.

Our algorithm leverages off the idea of analyzing sounds directly in the image space (i.e. spectrograms). This idea has been increasing in popularity recently [58][95]. For example, [58] analyzes music data by computer vision techniques; however, current work is limited to query-by-content, not motif discovery, and is explicitly specialized to music data.

3.2 Definitions

Before describing our audio motif discovery algorithm, we provide the necessary definitions based on the definitions in Section 2.2.

Informally, audio motifs are the most *similar* subsequences within a longer audio sequence. Thus, we must compute similarity with some measure of *distance*:

Definition 3.1 [DISTANCE BETWEEN SUB AUDIO SEQUENCE] The *distance* between a subset of sound spectrogram **S**, comprised of spectrogram subset $\mathbf{S}_{i,n}$ and another subset $\mathbf{S}_{j,n}$ is the CK distance [24], denoted $\boldsymbol{dist}(\mathbf{S}_{i,n}, \mathbf{S}_{j,n})$.

The CK distance measure is a relatively new, compression-based similarity measure, which exploits MPEG video encoding to measure the similarity between *real-valued* images [24]. The distance between two equal-sized images (denoted as **x** and **y**) is calculated as:

$dist(\mathbf{x},\mathbf{y})=((\text{mpegSize}(\mathbf{x},\mathbf{y})+\text{mpegSize}(\mathbf{y},\mathbf{x}))/(\text{mpegSize}(\mathbf{x},\mathbf{x})+\text{mpegSize}(\mathbf{y},\mathbf{y})))-1$

In Section 3.4, we will explain and justify the choice of this particular distance function [24].

We are finally in a position to define audio motifs more formally. To find the audio motif pair of (a *user* given) length *w* in a long audio sequence, we consider the pair-wise distances between each subsequence and all others. The pair with the smallest distance is the *audio motif*:

Definition 3.2 [AUDIO MOTIF] The *audio motif* of an *audio sequence* is the unordered pair of subsequences $\{\mathbf{A}_{i,n}, \mathbf{A}_{j,n}\}$ of a long audio sequence **A** of length *n* that is

the most similar. More formally, $\exists i,j\ \forall a,b,$ the pair $\{\mathbf{A}_{i,n}, \mathbf{A}_{j,n}\}$ is the *audio motif* iff $\boldsymbol{dist}(\mathbf{A}_{i,n}, \mathbf{A}_{j,n}) \leq \boldsymbol{dist}(\mathbf{A}_{a,n}, \mathbf{A}_{b,n})$, $|i-j| \geq w$ and $|a-b| \geq w$ $(i \neq j,\ a \neq b)$for $w>0$, where w is the audio motif length.

$$\{(A_{i*,n}, A_{j*,n})\} = \underset{\substack{(A_{i,n}, A_{j,n}) \in A \\ |i-j| \geq w}}{\arg\min}\ dist(A_{i,n}, A_{j,n})$$

Note that we refer to *audio motifs* even though we are searching the image space (the spectrograms). In Figure 3.1, we illustrate an example of an audio motif pair (leftmost and rightmost boxes) together with the other concepts introduced in this section.

Note that our audio motif definition excludes trivial matches of *audio subsequences* that match a part of a sound with itself, such that $i=j$ or $|i-j|< w$. Thus the motif pair must be strictly non-overlapping.Our experience analyzing real world audio has shown us that sections of pure silence (which we denote as $\mathbf{S}_{\mathbf{ps}}$, shown as pure black section in Figure 3.1) are quite common in scientific data [1]. These silent elements (not to be confused with simply *quiet* but non-zero time periods) may be caused by disconnected wires or data deliberately written with zero energy to denote the beginning/ending of an event. These sections can be problematic since all silences “sound” the same, and thus allow for perfect yet meaningless audio motifs.

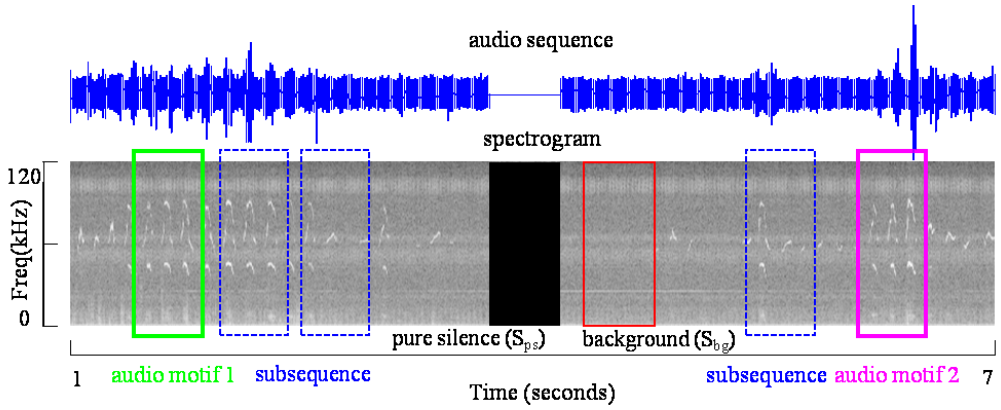

Figure 3.1: An illustration of our definitions.

Another special case we have to consider is that of a constant background sound (denoted as $\mathbf{S_{bg}}$, and illustrated in the area surrounded by the red box in Figure 3.1), which occurs *everywhere* in audio sequence. For example, if we are interested in finding audio motifs near a highway on a rainy day, then the *entire* background will have the sound of rain, which we would like to ignore. We exclude both pure silence and regions containing *only* the background sound if they last more than one motif length *w*.

It is important to note that our definition of *closest pair* does not preclude other possible definitions. For example, for some applications it might be convenient to consider the *K* closest pairs, or *all* objects within a user-given radius *R*. As noted in [72] in the context of time series motifs, if one can solve the closest pair problem efficiently, then the *K* closest pairs and user-given radius variants can also be solved using the *closest pair* subroutine with some linear time post processing. In particular, we have explored finding the top *K* motifs (for *K* equals up to 10) in the birds dataset discussed in Section 3.4.2; this required just a few minutes modifying the code, and took less than twice as

long as finding the *closest pair* (or $K = 2$). Nevertheless for clarity of presentation and consistency we limit discussion and experiments to the *closest pair* case in this work.

3.3 Finding Audio Motifs

We outline a detailed formal explanation and statement of our audio motif discovery algorithm in Section 3.3.2. However, for simplicity and clarity, we give some simple intuitions behind our ideas in the next section.

3.3.1 Intuition behind Audio Motif Discovery

Our entire approach is predicated on the following assumption. Similar sounds will produce similar images when transformed into spectrograms, and we can efficiently and effectively compute the similarity in the *visual space*. The idea that audio patterns can be revealed and measured in the image space has been exploited in some specialized domains [13][58][95]. However, these works require domain-specific feature extraction techniques to allow the similarity computation, a step we are anxious to avoid in order to create a universal and highly usable tool.

To be clear, using the spectrogram representation is, by itself, *not* the solution to our problem. To see this, we performed a simple clustering experiment using Scale-Invariant Transform Features (SIFT) [6][63] on a small dataset. SIFT is arguably the state-of-art for image matching, and the most obvious strawman to compare against [51].

In Figure 3.2 , we show seven pairs of two-second audio snippets of diverse sounds produced by *coyotes*, *crickets*, *squirrels*, *katydids*, *ravens*, *owls*, and *explosions*. For all

images we extract SIFT features to form a feature description using Lowe's algorithm [63]. We use the number of matched keypoints/features points as a similarity measure between two images[8]. Figure 3.2 shows the clustering of the seven pairs using SIFT; the results are only slightly better than random.

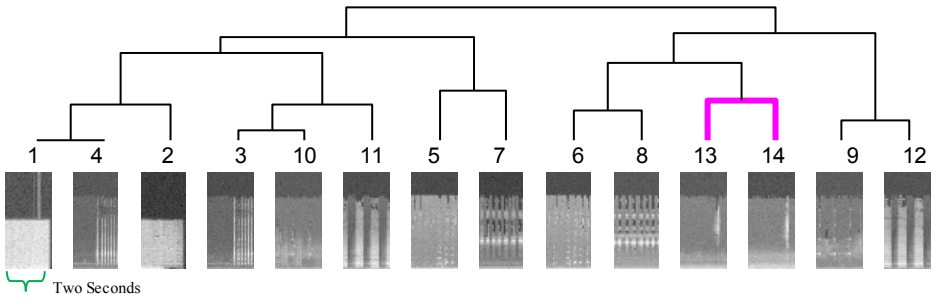

Figure 3.2: A clustering of seven pairs of two-second audio recordings of various sounds using SIFT. Only one pair {13,14} is correctly clustered (*katydids*).

These results are not promising. In contrast, we tested the same dataset shown in Figure 3.2 using CK distance measure, and the result is shown in Figure 3.3.

[8] This (carefully annotated) code, along with *all* code and data used in this work is archived at [5].

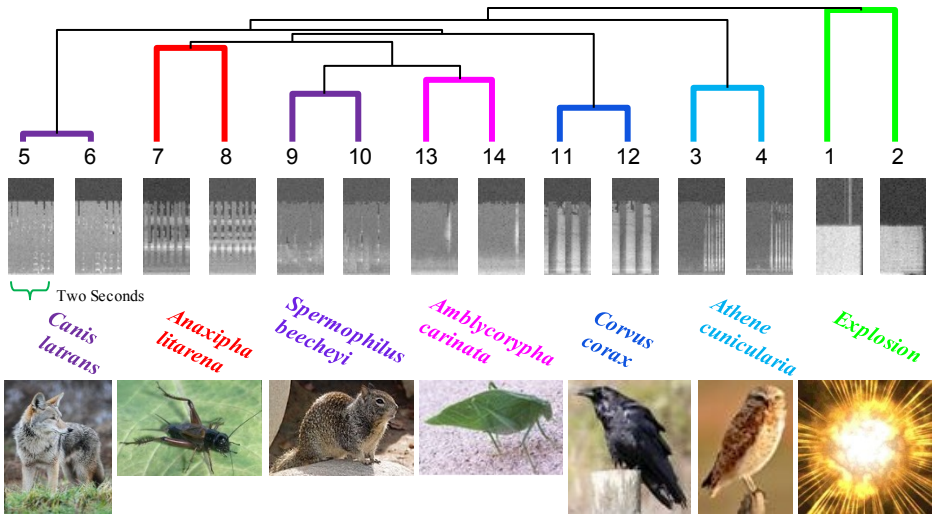

Figure 3.3: A clustering of the dataset used in Figure 3.2 with CK distance measure. All pairs are correctly clustered, and the explosion sound is an outlier to animal sounds. No parameters were adjusted here.

This result highly suggests that the CK distance measure on spectrogram images is measuring similarity in a way that maps to human notions of sound similarity.

Surprisingly, the CK distance measure can be very effective even on *human speech*, which is obviously the most studied audio source [26][45][53]. To see this, in Figure 3.4, we show a reading of *A Dream within a Dream* by Edgar Allan Poe. We naturally expect repeated structure in most poetry [39], and although this short poem only has 24 lines in two stanzas, we do find two obvious repetitions as the audio motif (the last line of both verses).

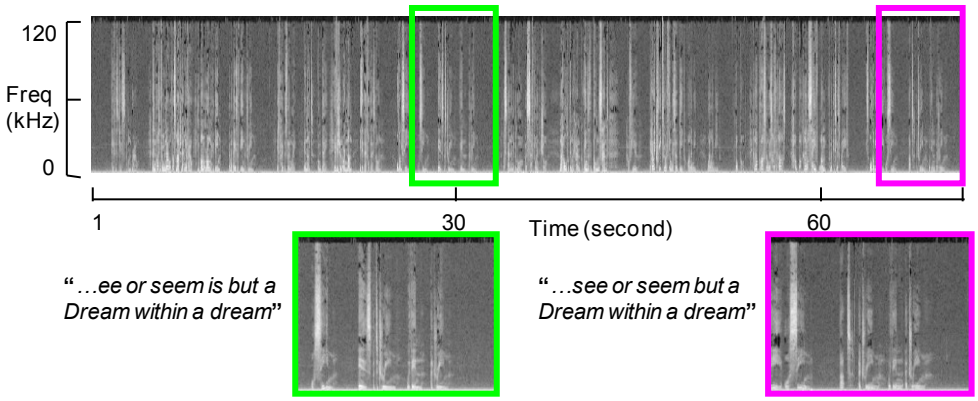

Figure 3.4: *top*) A performance of *A Dream Within A Dream* has a motif of length seven seconds. *bottom left*, *right*) A zoom-in of the two occurrences and the corresponding sentences. The reader can go to [5] to hear the original sound file and the discovered motifs.

As the audio motif shown in Figure 3.4 demonstrates, the CK distance measure can accurately find the *only* repeated pattern ("…*see or seem* (*is*) *but a dream within a dream*") of this audio recording. Note that the distance measure was *exactly* the same as used in Figure 3.3, no tuning or adjustments were necessary to go from (mostly) animals to poetry.

In this example, generating and testing *all* possible motifs to find the best one (cf. definition 3.2) requires about 15 minutes, although the audio clip is barely a minute long. Such languor may be tenable for music and other relatively short audio files; however, in scientific domains we need to be able to find motifs in datasets that are orders of magnitude longer. In the next section, we will outline our strategy for making motif discovery tenable even in such massive datasets.

3.3.2 A Formal Description of our Algorithm

Given a spectrogram S transformed from a long *audio sequence* A, and a user specified length *w*, our goal is to find audio motifs as described in definition 3.2.

For ease of exposition we will begin by explaining the generic search algorithm, and then we will introduce the modifications that make it more tractable.

In Algorithm 3.1 Line 1, our algorithm begins by initializing the *best-so-far* CK distance corresponding to the audio motif pair to infinity. In Line 2 we generate all *N* subsections *D*. This consists of *all* subsections except those excluded because they are $\mathbf{S_{ps}}$ or $\mathbf{S_{bg}}$ (cf. Section 3.2). One pair (with indices differing by at least *w*) from this set will eventually become the motif pair.

In Line 4 we are finally in a position to test the $N(N\text{-}1)/2$ possible pairings of subsections for the pair that minimizes the CK distance, our audio motif. However, in what order should we search? Clearly, if we search exhaustively then the order makes no difference. However, there are two reasons why we may want to avoid exhaustive search and terminate early. The first is to respond to a user request to stop, so the user can treat the algorithm as an *Anytime* search algorithm [10]. The other reason is that we may wish to frame the search probabilistically, supporting a user request of the type "*stop searching when there is only a one in a thousand chance that the current best-so-far is not the true motif*."

As we will shortly show, we can support these useful variants by using different heuristic functions as defined in Sections 3.3.3, 3.3.4, 3.3.5 and 3.3.6.

Algorithm 3.1 *AduioMotifDiscovery* (*S*, *w*, *p*)

Require: A Spectrogram *S* transformed from original audio archive A, Audio motif length *w*, Early abandoning probability threshold *p*.
Ensure: Return Audio motif pair *D*.

```
 1: best-so-far ← Inf
 2: Discard Sps and Sbg area from S and generate only meaningful subsequences into D. (i.e. no
    silence and no common background sounds)
 3: N ← |D|
 4: for loopCnt ← 1 to N(N-1)/2 do
 5:    [i, j, stopFlag] ← heuristicFunction(loopCnt, type, p, best-so-far)
 6:    distance ←dist(Di, Dj)
 7:    if distance < best-so-far and |i-j| ≥ w then
 8:       best-so-far ← distance
 9:       Pos1 ← i
10:       Pos2 ← j
11:    if stopFlag == True then
12:       break out of the loop
13: return DPos1, DPos2
```

Returning for a moment to the generic version of our algorithm, in Line 6 we measure the CK distance between the two candidate subsections and in Line 7 we check to see if this pair of *audio subsequences* has a smaller distance than the current *best-so-far* distance. If that is the case, we update the *best-so-far* distance, and record the relevant locations (Lines 8 to 10).

Having seen the generic algorithm we now consider four variants produced by using different heuristic functions and stopping criteria.

3.3.3 Brute-force Algorithm

The brute-force heuristic is outlined in Algorithm 3.2. This heuristic simply lists every possible combination of pairs of audio subsequences $\{Pos_1, Pos_2\}$ in lexical order and

allows search to exhaustion. Note the last two arguments are just place-keeping dummy variables.

Algorithm 3.2 *heuristicFunction* (*index*, *bruteForce*, *dummy*, *dummy*)

Require: A Spectrogram *S* transformed from original audio archive A, Audio motif length *w*, Early abandoning probability threshold *p*.
Ensure: Return Audio motif pair *D*.
1: Generate testing candidates in a lexical order, which is from left to right with the sliding window
2: [idx_i, idx_j, **False**] ← Return an array containing the candidates' positions

Run to completion this heuristic clearly lists the pair of audio subsequences {(D_{Pos1}, D_{Pos2})} that are optimal.

Note that for many real world problems there may be *many* motifs that are of high quality, and finding *any* pair may be sufficient. For example, if a full day's recording in a forest in Kenya has dozens of the stereotypical calls of the *Common Scimitarbill* (cf. Section 3.4.1.1) then reporting any pair as a motif will suffice for many applications. However, if the recording started at midnight, and the bird is most vocal just before dusk, then the linear-ordered brute-force search will not produce a good motif (i.e. have a low *best-so-far* value) until very late in the search process. As we show in the next section, we can mitigate this with a random ordered search.

3.3.4 Random Search Algorithm

In contrast to lexical-ordered search discussed in the previous section, we can consider random ordered search, which has long been used to guard against pathological situations where an iterative improvement algorithm (i.e. *best-so-far* linear search) does not improve much until the last few iterations [57].

The speed at which the *best-so-far* decreases does not matter if we run to completion. But if we allow the user to interrupt the search and peek at the best current motif, then we can expect that random search as shown in Algorithm 3.3 to do better on average, as its *best-so-far* value will converge much faster.

Algorithm 3.3 *heuristicFunction* (*index*, *Random*, *dummy*, *dummy*)

Require: A Spectrogram *S* transformed from original audio archive A, Audio motif length *w*, Early abandoning probability threshold *p*.
Ensure: Return Audio motif pair *D*.
1: Generate testing candidates in a random order, which is produced by random permutation.
2: [idx_i, idx_j, **False**] ← Return an array containing the candidates' positions

---

The idea of supporting interruptions (possibly followed by continuations) of an algorithm is known as creating an *anytime algorithm*, and anytime algorithms have seen a recent surge of interest in the data mining community [6][61]. As we shall empirically show in Section 3.4, random ordering does greatly improve the "*early returns*" property of our search. However, in the next section we show that we can do even better.

3.3.5 Euclidean Distance Ordering Algorithm

Anytime algorithms for searching tend to work best if they can test promising solutions early. This seems to open a *chicken-and-egg* type paradox, since we do not know if a pair of subsequences will make a good motif until *after* we test them. However, if we had an approximate test of quality that was much faster than the CK distance itself, then we could sort by that measure and increase our chances of seeing good solutions early on. In the most general case, CK distance measure has resisted attempts at fast approximation [51]. However, in our previous work we have shown that in the special case of

spectrograms, the Euclidean distance between the images is a reasonable approximation of CK distance measure, but can be computed at least three orders of magnitude faster. Moreover, the Euclidean distance computations are amenable to many tried-and-tested speedup techniques including early abandoning, triangular inequality, and indexing.

Given a spectrogram image **S** with size M×N, **S** can be written as $\mathbf{S} = \{\mathbf{S}^1, \mathbf{S}^2, \ldots, \mathbf{S}^{MN}\}$ according to the gray levels of each pixel. The Euclidean distance $dist_E(\mathbf{S}_1, \mathbf{S}_2)$ between two images $\mathbf{S}_1$ and $\mathbf{S}_2$ is defined as

$$dist_E(\mathbf{S}_1, \mathbf{S}_2) = \sum_{k=1}^{MN} (\mathbf{S}_1^k - \mathbf{S}_2^k)^2 = (\mathbf{S}_1 - \mathbf{S}_2)^T (\mathbf{S}_1 - \mathbf{S}_2)$$

Thus, as shown in Algorithm 3.4 we propose to sort the pairs returned by the heuristic in ascending order (denoted as *Z*) of their Euclidean distance.

Algorithm 3.4 *heuristicFunction* (*index*, *ED*, *dummy*, *dummy*)

Require: A Spectrogram *S* transformed from original audio archive A, Audio motif length *w*, Early abandoning probability threshold *p*.
Ensure: Return Audio motif pair *D*.
1: *sortOrder* $\tau \leftarrow$ Sort all the subsection pairs by the E*uclidean distance* between them in an ascending order
2: *stopFlag* ← DidUserRequestAnInteruption()
3: [idx_i, idx_j, **False**] ← Return an array containing the candidates' positions based on τ and *stopFlag*.

The high degree of correlation between the Euclidean distance and CK distance is hinted at in Figure 3.5.*left*.

Before moving on, it is critical to note that while the Euclidean distance and the CK distance are correlated, we cannot simply use the Euclidean distance to directly find motifs. For example, it does not produce the correct answers for the examples shown in Figure 3.3 and Figure 3.4 (we shown this in [5]). Nevertheless, as we show in the next

section, we can use the Euclidean distance as a heuristic to both guide the search order, and to tell us when we can abandon the search with a small, user-defined probability of missing the optimal answer.

3.3.6 Probabilistic Motif Discovery Algorithm

As noted above, the anytime algorithm framework is gaining increasing acceptance by both the data mining community and domain practitioners. However, at least some of the latter may be reluctant to use anytime algorithms as intended. Nevertheless, most biologists are much more comfortable with the idea of statistical significance, the idea of considering if a result could be explained by a chance at a given probability cutoff (i.e. the *significance level*). We can support this type of worldview by allowing the user to specify the probability of returning a non optimal motif pair. In essence, we propose to allow queries of the form "*stop searching when there is only a one in a million chance that the current best-so-far is not the true motif*."

By exploiting the Euclidean distance ordering heuristic, introduced in the previous section, we can support such queries. As we shall see later in our experimental section, we can trade a *small* probability of a *slightly* suboptimal result for several orders of magnitude speedup.

The intuition behind the Probabilistic Early Abandoning Audio Motif Discovery (PEAMD) algorithm is to internally estimate the likelihood that the current *best-so-far* motif is optimal, and signal to abandon the search once this likelihood exceeds the user's tolerance for a sub-optimal result. This signal is passed into the generic search algorithm

in Line 5 of Algorithm 3.1. How can we estimate this probability? Figure 3.5 gives a visual intuition. In Figure 3.5.*left* we show the relationship between the Euclidean distance and CK (estimated from the dataset shown in Figure 3.4).

Let P_d(*best-so-far*) be the probability that the remaining pairs of subsequences in the Euclidean searching order Z (the y-axis ordering of Figure 3.5.*left*) contains a better match than the match represented by the current *best-so-far*. Given that the two measures are highly correlated, we can estimate P_d(*best-so-far*), which is monotonically non-decreasing as we search because the *best-so-far* can only decrease by definition (i.e. the red *bsf_dist* bar shown in Figure 3.5.*right* can only move *leftwards*), and the positive correlation means that the mean of the distribution of estimated values of untested pairs can only move *rightwards*.

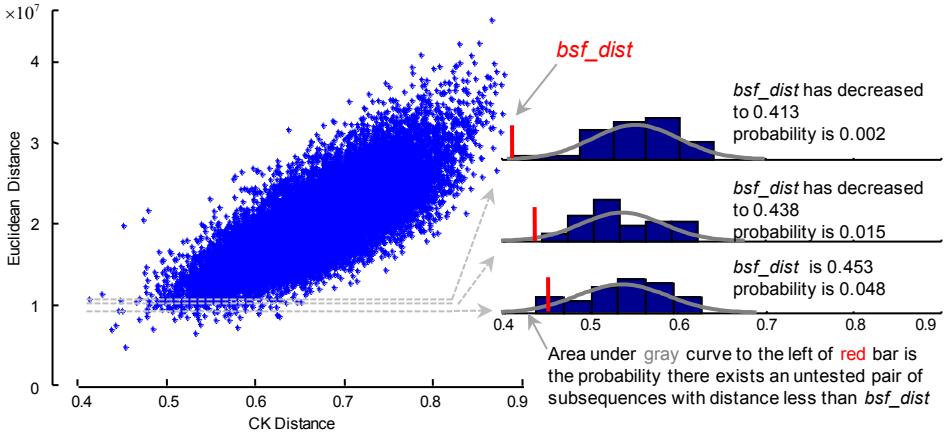

Figure 3.5: *left*) The empirical relationship between Euclidean and CK distance. *right*) As we search in Euclidean order (the y-axis order), from bottom to top. The *best-so-far* distance moves leftwards and the mean of the Gaussian distribution moves rightwards.

Concretely, we compute d, the CK distance for the ε items below the *lowest* dash-line in Figure 3.5.*left* to form the histogram shown at the bottom right of Figure 3.5. Here $\boldsymbol{\varepsilon}$ is a small number, enough to learn a Gaussian (we use $\boldsymbol{\varepsilon} = 50$).

$$d_k = dist(Z_{((k-1)\cdot\varepsilon+1:k\cdot\varepsilon,1)}, Z_{((k-1)\cdot\varepsilon+1:k\cdot\varepsilon,2)})$$

This property of the distance distribution can be realized by a Gaussian process (GP). The probability vector $\{\varphi_k\}$ is drawn from a GP as $\varphi_k \sim N(\mu_k, \sigma_k^2)$, where μ_k is the mean of and variance σ_k^2, shown as the gray "bell" curve. For example, the *best-so-far* distance decreases from 0.453 to 0.413 and the corresponding P_d(*best-so-far*) of the distance distribution changes from 0.048 to 0.002 as shown in Figure 3.5.*right*. The area below the gray curve, left of the *best-so-far* marker, is the probability that there exists an untested pair of subsections with distance less than the *best-so-far* distance. If P_d(*best-so-far*) is less than the user threshold (denoted as p) then we simply set the *stopFlag* to be true, and the invoking generic search algorithm will terminate. The formal algorithm of PEAMD is outlined in Algorithm 3.5.

Algorithm 3.5 *heuristicFunction* (*index*, *PEAMD*, ***p***, *best-so-far*)

Require: A Spectrogram S transformed from original audio archive A, Audio motif length w, Early abandoning probability threshold p.
Ensure: Return Audio motif pair D.

```
1:  Call the procedure Euclidean distance measure order search in Algorithm 3.4
2:  Z ← heuristicFunction(index, ED)
3:  for k ← |Z| do
4:      for j ← 1 to ε do
5:          d_k ← Compute CK distance for pair k+j-1 based on Z
6:      d_k ~ N(μ, σ²) //Build Gaussian distribution over d_k
7:      prob ← P_d(best-so-far) // CDF of the current best-so-far distance
7:      if prob < p then
8:          stopFlag ← True
11: [idx_i, idx_j, stopFlag] ← Return an array containing the candidates' positions and
    stopFlag
```

Our probabilistic framework makes some assumptions that are strongly empirically warranted (i.e., that "slices" of the cloud of data points in Figure 3.5.*left* are approximately Gaussian), and some that are less realistic (i.e. the independence of the "slices"). However, all such assumptions tend to make our algorithm err *only* on the conservative side.

3.4 Experiments

We have designed all our experiments to ensure that they are *very* easy to reproduce. A supporting webpage [5] contains *all* the code, datasets, and raw data spreadsheets used in this work. Moreover, although this work is completely self-contained, the webpage contains additional experiments and video/sound files to allow an interested reader to directly see and hear the motifs discovered and the original source sounds.

3.4.1 Motif Discovery in Human Speech

Human speech is surely the most studied sound source [28]. Recurrences in human speech have implications for studying linguistics, cognitive disorders, and pragmatic applications in indexing speech [54], etc. Thus, we will test our algorithm with a familiar audio book in Section 3.4.1.1 and show a comparison with a state-of-the-art tool in section 3.4.1.2.

3.4.1.1 Motif Discovery in an Audio Book

A famous example of reoccurring text can be found in the book *The Cat in the Hat* by Dr. Seuss [34]. It is an impressive feat of wordplay that this **1629** word book contains only **236** distinct words, and this is suggestive of significant repetition. The experiment shown in Figure 3.6 demonstrates that our algorithm finds a meaningful motif pair (three seconds long) from an audio performance of story (professional male actor). Note that our algorithm is robust to the fact that one occurrence contains an additional word ("*ball*").

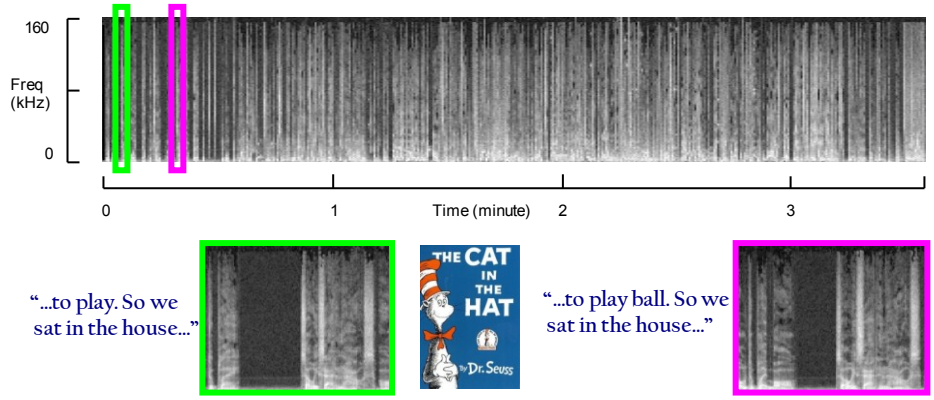

Figure 3.6: *top*) A performance of *The Cat in the Hat* has a motif three seconds long. *bottom*) A zoom-in of the two occurrences and corresponding sentences [34].

3.4.1.2 Comparison with state-of-the-art Work

The utility of "black-box" CK distance on human speech may be surprising, given that most human speech processing algorithms are highly optimized with domain knowledge of linguistics, phonetics, etc. To further explore this, we attempted to reproduce a result in a recent state-of-the-art work [54]. Here the data is a nine second snippet of telephone

quality audio. We take the same spoken query of word "*California*" as [54], and build the same type of dot-plot, but use the CK distance measure as shown in Figure 3.7.

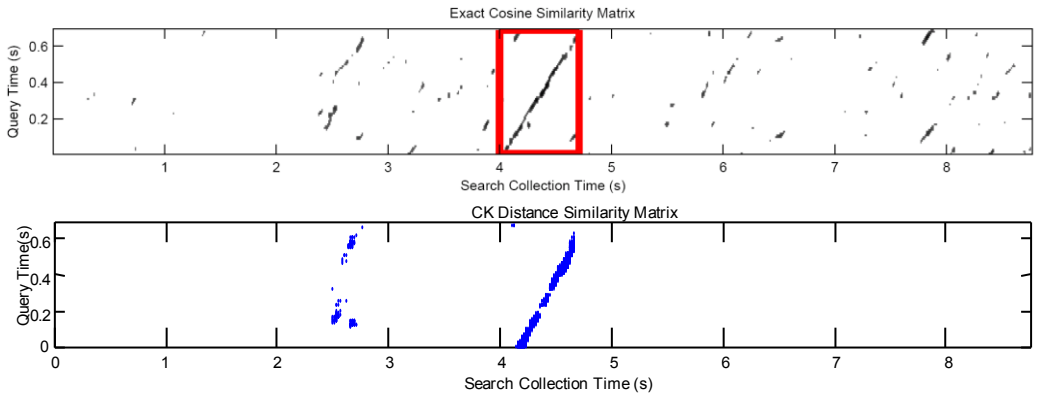

Figure 3.7: *top*) A screenshot from [54] of a state-of-the-art human speech recognition algorithm correctly matching two utterances of "*California*" (red box). *bottom*) Our re-creation of the experiment using only the CK distance measure.

While the interpretation of the results is somewhat subjective, our simple approach does seem *at least* competitive with current human speech processing methods without the need for tuning the *nine* parameters used in [54]. Note that we are only comparing on *effectiveness* here; [54] does not make claims on *efficiency*.

3.4.2 Motif Discovery in Bird Songs

Complex songs produced by animals (bats, whales, mice, birds) have been receiving increasing attention because summaries of these sounds can be a measure of the health of the ecosystem and its biodiversity. For example, The Long Island Sound Study, a six-year research project, is a notable effort devoted to protecting the environment [90]. Birds, though still a common sight even in cities, are facing threats from habitat reduction.

While bird songs have been explored in several research efforts [9][14], like human sound processing, the algorithms tend to be very specialized and parameter-laden. How well can we do with no parameters? We tested our algorithm on audio sequences of the *Common Scimitarbill* from *xeno-canto* [7]. One representative experiment is shown in Figure 3.8. We obtained similarly intuitive results for many other diverse species. We encourage the interested reader to hear/see them at [5].

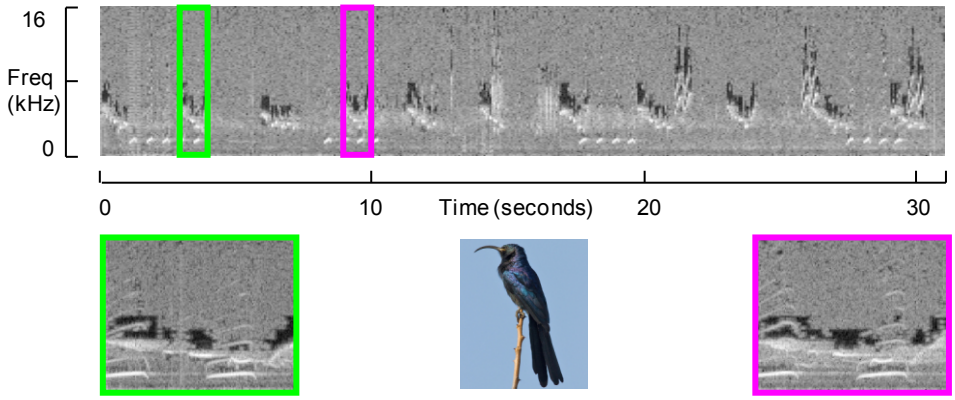

Figure 3.8: *top*) A 31-second excerpt of a two-minute audio performance of a Common Scimitarbill. *bottom*) A zoom-in of the two one-second long audio motif occurrences.

3.4.3 Motif Discovery in Music Data

Algorithms for automatic discovery of repeated patterns in music data can be very useful; they have a number of applications for content-based retrieval, indexing, and audio-thumbnailing (summarization) [11][49][58]. In the absence of formal benchmarks for music motif discovery, we will attempt to reproduce an experiment in a highly cited paper [11].

We attempted to find motifs in André Bourvil's song *C'était bien* [21]. As with [11], we set the motif length to three seconds and as shown in Figure 3.9 we discover a motif

phase *Et c'était bien*, the song title itself. In contrast to the string-matching techniques on a derived symbolic representation used by [21] and almost all music motif efforts, we do not need to extract explicit features or tune any parameters. As before, the same algorithm works on mice and men, on birds and whales, with no human intervention.

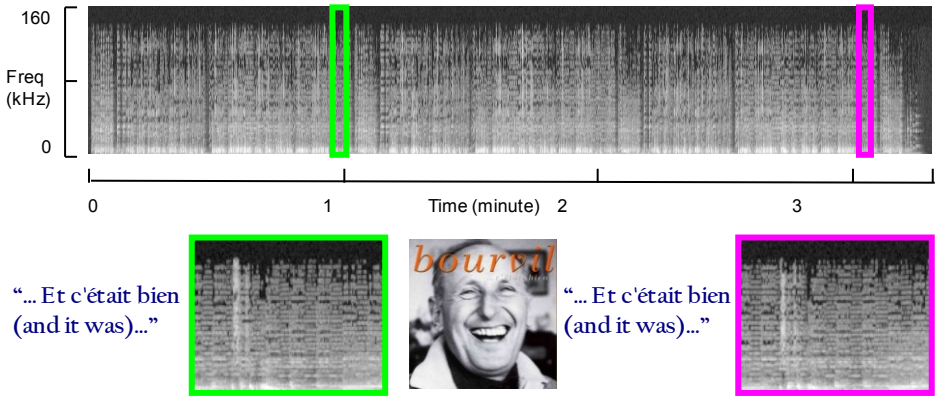

Figure 3.9: *top*) A performance of Bourvil's song *C'était bien* has a three-second motif. bottom) A zoom-in of the two motif occurrences and corresponding lyrics.

As a sanity check, we tested to see if *music* motif algorithms could be made to work for our bird/mice/whale data (to be fair, no one has claimed they might). After all, biologists do speak of mice courtship *songs* [43], bird *choruses*, and whale *melodies*, etc. However, in spite of significant effort, we could not make the music motif algorithms work for any biological datasets [5].

3.4.4 Motif Discovery in Mice Vocalization

Mice have been extensively used as genetic models of human disease for almost four decades. They can produce ultrasonic vocalizations, inaudible to humans. These vocalizations are important to researchers who study *human* pathologies by testing the

effects of manipulating homologue genes in *mice* [95]. Analyzing vocal behaviors of mice models in this manner has lead to the discovery of the genetic cause of Autism [44], and has shown great promise for the study of Alzheimer's disease [68].

We applied our audio motif discovery algorithm to various subsets of the mice vocalization dataset studied in [95]. One such result is shown in Figure 1.1. We find that we can obtain similar results to [95] (and [43]) but *without* the need for explicitly extracting syllables, a painstaking and time consuming step. This experiment speaks volumes to the generalizability of our algorithm.

In this section we hint at the *actionability* of audio motif discovery by showing that motifs, once discovered, can be used to test for changes in vocal repertoire that may be attributable to genes that were deliberately *deleted* (in genetics parlance "*knocked out*" or "*KO*") from the mouse genome.

We obtained six hours of vocalizations recorded during courtship/mating of various pairs of mice (only males vocalize). These sessions were annotated by the mice behaviors, from the set: {`Defensive` (*D*), `Ejaculate` (*E*), `Grooming` (*G*), `Intromission` (*I*), `Mounting` (*M*), `No Contact` (*N*), `Rooting` (*R*) and `Sniffing` (*S*)}. Neuroscience researchers at UCR want to know if vocal repertoire or frequency during these behaviors differ for different mice genomes. Below we hint at the answer to this question.

We applied our algorithm to the data and found many instances of motif shown in Figure 3.10.*top*. Having discovered this motif, we used a sliding window to calculate its density over time. As shown in Figure 3.10.*middle*, this particular motif occurs about 4.1

times more frequently during `Sniffing` than during `Rooting` for this particular strain of *KO* mice. Moreover, because we are able to automate this process (most similar research efforts resort to manual counting [43][68][95]) we can automatically search through a large space of motifs × behaviors × genomes, scoring the frequency differences by significant tests.

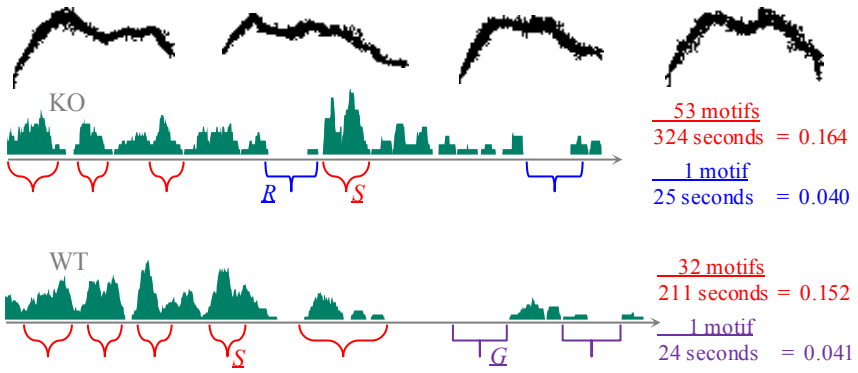

Figure 3.10: *top*) Sample instances of a motif discovered from mice vocalizations by applying our algorithm. *middle*) Comparing the number of motifs during *S* and *R* behaviors for a sample recording of *KO* mice vocalization. *bottom*) Comparing the number of motifs during *S* and *G* behaviors for a sample recording of WT mice vocalization.

In Figure 3.10.*bottom*, we show another example of a similarly significant contrasting pattern, this time in WT (wild type) mice. In this case we noted a dearth of the motif during `Grooming`. Note in [5] we show that the same algorithm that finds motifs in mice, expressed in about the 40 to 110 kHz range, also works for whales, expressed in a completely disjoint range of about 20 Hz to 24 kHz. The only difference in the two experiments was the suggested length of the motif was increased for the much larger whales, as suggested by allometry of vocal production [31].

3.4.5 Scalability of Audio Motif Discovery

After demonstrating the *utility* of our algorithm, we now show the *scalability* of finding audio motifs with an example of human speech data, a ten-minute performance of "The Raven" as shown in Figure 3.11.

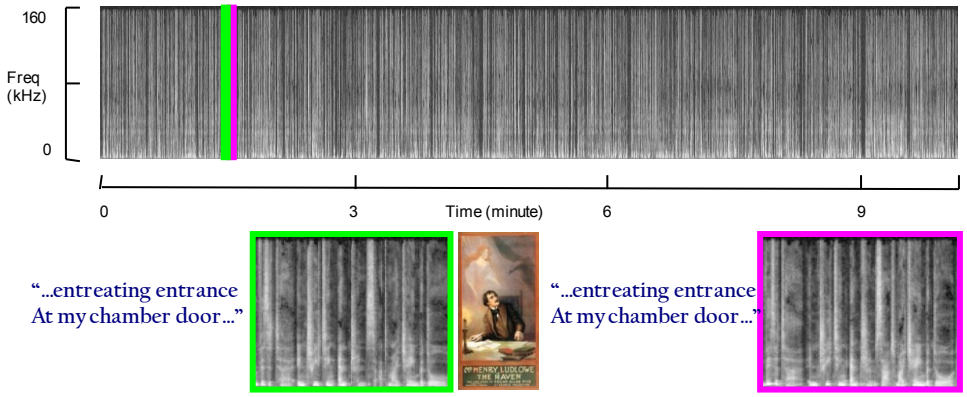

Figure 3.11: *top*) A performance of "The Raven" has a motif of length seven seconds. *bottom*) A zoom-in of the two occurrences and the corresponding text.

We compared the four algorithms (brute-force, random, Euclidean distance reordering heuristic, and PEAMD search) shown in Figure 3.12. The brute-force search takes **twenty-two** hours. Random search takes the same time, but converges more quickly, so if "anytime" interrupted after **fourteen** minutes it would have had the correct result [6][61].

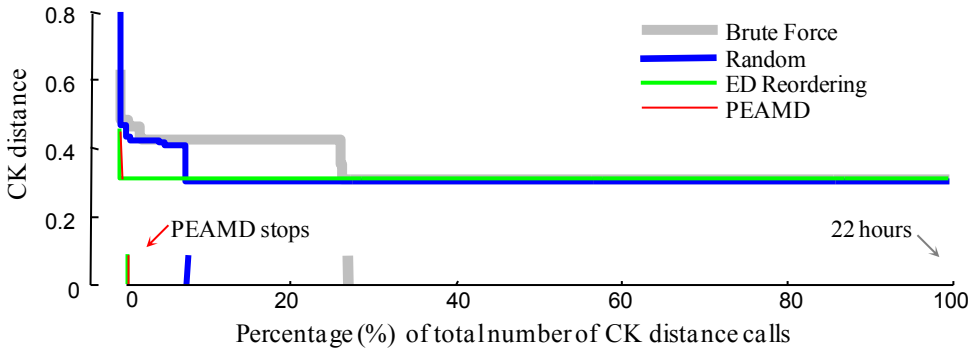

Figure 3.12: A comparison of efficiency of four algorithms normalized to the 100% time taken for brute-force search.

Euclidean ordered search converges to the optimal motif in a few minutes, and about a minute later PEAMD (allowing a 1 in 10,000 chance of a non-optimal answer) is confident enough to abandon the search and return the correct answer. Thus, using the PEAMD we can search audio files that are on the order of *hours* in real time. Examining *day*-long audio recordings does require more than a day (unless there are large time periods of silence, cf. Section 3.2).

We have *informally* shown the accuracy of our algorithm in an intuitive fashion. For example in the ability to find the chorus of a poem/song (cf. Figure 3.6, Figure 3.9 and Figure 3.11). However, to *formally* evaluate the accuracy of the PEAMD algorithm, we compute the ratio of the motif distance returned by the brute-force linear search algorithm over the motif distance algorithm returned by our algorithm. Numbers approaching one indicate that there is little difference in the two algorithms output. As we can see in the second column in Table 3.1, this is strongly the case.

In addition, we compute the *speedup* of the PEAMD algorithm compared to the brute-force algorithm for all the datasets we tested. The results are shown in the third column of Table 3.1.

Dataset (Figure Number)	$\frac{BF\_dist}{PEAMD\_dist}$	$\frac{time\ BF}{time\ PEAMD}$
A Dream within a Dream (3.5)	0.94	4,100
Mice vocalization (3.1,3.11)	0.92	179
The Cat in the Hat (3.7)	1.00	6,591
Scimitarbill (3.9)	1.00	267
C'était bien (3.10)	1.00	605
The Raven (3.12)	1.00	525

Table 3.1: Comparison between PEAMD and Brute-Force Algorithms.

The results show a significant speedup for our PEAMD algorithm across diverse audio archives. Moreover, in four out of six cases the results returned are *identical* to the brute-force algorithm.

It is worth considering the two cases where our algorithm failed to return the optimal answer; in particular we can ask how badly did we fail? The answer can be seen directly in Figure 3.4. Here the two motifs are *very* slightly misaligned. One begins with "*See or seem...*" and the other begins "*ee or seem...*". Thus the answer is semantically correct. Similar remarks can be made for the mice vocalization dataset. Due to the page limitations, we refer to [5] for more scalability analyses and experiments.

3.4.6 Sensitivity of User-Choice (Motif Length)

The results shown in previous sections demonstrate the efficiency and effectiveness of our algorithm in finding motifs for a *given* user-defined length. However, the reader may wonder how critical this user choice is. Clearly, motifs can exist on different scales, for example repeated *words*, and repeated *phrases* in speech. However, it would be very undesirable if the results returned were very sensitive to tiny changes in this user choice.

It is hard to make any strong claims about this issue, as one could construct an artificial dataset for which the motifs of length w-ε, w, and w + ε are disjoint. However, on real data we generally find that our algorithm will report the same essential concept when the motif length is within [w-σ, w+σ] for values of σ which are a significant fraction of w.

For example, let us revisit the *The Cat in the Hat* dataset. The motif length used in Section 3.4.1.1 was three seconds; however, we found that our algorithm allows us to set σ anywhere in the range of [1.7 sec, 3.9 sec] (-43%~ 30%) to obtain motifs that correspond to the same basic phrase. This result suggests a simple way to explore a dataset for which one poor intuition about possible motif lengths. We can simply set w to be a small number and find motifs of length w, 2w, 4w, 8w, etc. The efficiency of the PEAMD algorithm makes such iterative doubling search tenable.

3.5 Conclusion

In this chapter we introduced a scalable and extremely general framework for finding audio motifs. We have demonstrated the utility of audio motifs analysis in diverse domains including music, human speech, mice vocalizations, and bird songs. By comparisons to existing work (Figure 3.7, Figure 3.9) we have shown that the representative power of our general purpose distance measure is typically competitive with domain specialized measures. While there is no obvious rival strawman to compare to in terms of *efficiency*, we have shown that by using probabilistic early abandoning we can examine most realistic length scientific recordings in much less than real time.

For brevity we have hinted at the *utility* of audio motifs only in the mice genetics domain; however, in data types as diverse as text, DNA, time series, and video, motif discovery is often leveraged for diverse types of analyses [47][55][58]. We believe this work has the potential to enable analogous analyses for audio.

We have claimed that our method is essentially parameter free. The reader might object to this claim, noting for example that the algorithm that converts audio to a spectrogram representation requires several parameters to be set. This is true, but in most cases the best parameters have been determined by the community decades ago. For example, virtually all mouse researchers truncate below 20 Hz and above 100 kHz [40][68][95]. The best parameters for *human* vocalization research are even better understood [53].

In future work, we hope to leverage off the Minimum Description Length principle to automatically find the natural length for motifs, thus removing the need for this user input, the only true parameter we need to set. However, we note that even here, as we showed in Section 3.4.6, our algorithm is not particularly sensitive to this setting.

Finally, we have made all code and data freely available in perpetuity so others can confirm, use, and extend our work [5].

Chapter 4

Towards Never-Ending Learning from Time Series Streams

Time series classification has been an active area of research in the data mining community for over a decade, and significant progress has been made in the tractability and accuracy of learning. However, virtually all work assumes a one-time training session in which labeled examples of all the concepts to be learned are provided. This assumption may be valid in a handful of situations, but it does not hold in most medical and scientific applications where we initially may have only the vaguest understanding of what concepts can be learned. Based on this observation, we propose a never-ending learning framework for time series in which an agent examines an unbounded stream of data and occasionally asks a teacher (which may be a human or an algorithm) for a label. We demonstrate the utility of our ideas with experiments that consider real world problems in domains as diverse as medicine, entomology, wildlife monitoring, and human behavior analyses.

The rest of this chapter is organized as follows. In Section 4.1 we briefly discuss related work, before explaining our system architecture and algorithms in Section 4.2. We provide an empirical evaluation on a host of diverse domains in Section 4.3, and in Section 4.4 we offer conclusions and directions for future work.

4.1 Related Work

The task at hand requires contributions from, and an understanding of, many areas including; frequent item mining [29], time series classification [33], hierarchical clustering, crowdsourcing, active learning [84], semi-supervised learning, etc. It would be impossible to consider all these areas with appropriate depth in this work, thus we refer the reader to [5] where we have a detailed bibliography of the many research efforts we draw from.

However, it would be remiss of us not to mention the groundbreaking NELL project lead by Tom Mitchell at CMU [27] which is the inspiration for the current work. Note however that the techniques used by NELL are informed by very different assumptions and goals. NELL is learning *ontologies* from *discrete* data that it can crawl *multiple* times. In contrast our system is learning prototypical time series *templates* from *real-valued* data that it can only see *once*.

The work closest in spirit to ours in the *time series* domain is [17]. Here the authors are interested in a (human) activity inference system with an application to psychiatric patient monitoring. They use time series streams from a wrist worn sensor to detect *dense motifs*, which are used in a periodic (every few weeks) retrospective interview/assessment

of the patient. However, this work is perhaps best described as a *sequence* of *batch* learning, rather than a true *continuous* learning system. Moreover, the system requires at least seven parameters to be set, and significant human intervention. In contrast, our system requires few (and relatively non-critical) parameters, and where humans are used as teachers, we limit our demands of them to providing labels only.

4.2 Algorithms

The first decision facing us is which base classifier to use. Here the choice is easy, there is near universal agreement that the special structure of time series lends itself particularly well to the nearest neighbor classifier [33][50][71]. This only leaves the question of which *distance measure* to use. There is increasing empirical evidence that the best distance measure for time series is either Euclidean Distance (ED), or its generalization to allow time misalignments, Dynamic Time Warping (DTW) [33]. DTW has been shown to be more accurate than ED on some problems; however it requires a parameter, the *warping window width*, to be carefully set using training data, which we do not have.

Because ED is parameter-free, computationally more tractable, and allows several useful optimizations in our framework (triangular inequality etc), and works *very* well empirically [33][71], we use it in this work. However, nothing in our overarching architecture specifically precludes other measures.

4.2.1 Overview of System Architecture

We begin by stating our assumptions:

- We assume we have a never-ending[9] 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**.

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 **[**20**]**, which manually attempts some of the goal 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, **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 1.2.

- We assume we have access to a teacher (or *Oracle* [84]), possibly at some cost.

[9] 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 [33].

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 the sparseness of our assumptions and the 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 4.1 which attempts to divorce the domain dependent and domain independent elements.

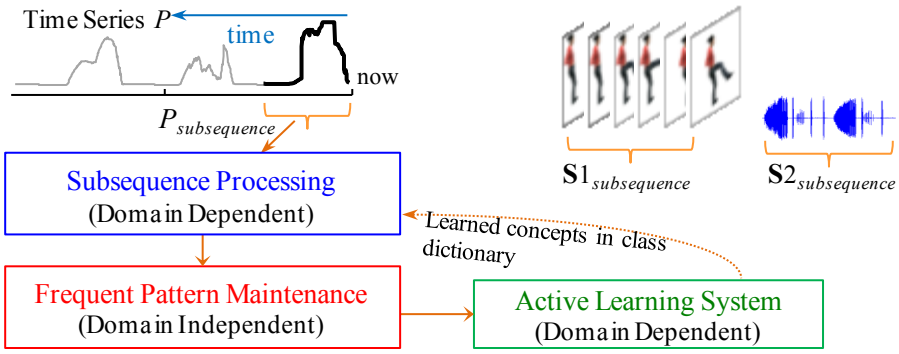

Figure 4.1: An overview of our system architecture. The time series P which is being processed may actually be a proxy for a more complex data source such as audio or video (*top right*).

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 4.1.*top.right*.

Our framework is further explained at a high level in Algorithm 4.1. We begin in Line 1 by initializing the class dictionary, in most cases just to `empty`. The dictionary format is defined in Section 4.2.2. We then initialize a dendrogram of size *w*. We will explain the

motivation for using a dendrogram in Section 4.2.4. 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), then 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 (cf. Section 4.2.3) 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.

Algorithm 4.1 *NeverEndingLearning* (*S*, *P*, *w*)

Require: A high dimensional data stream *S*, proxy stream *P*, and number of time series contains in memory *w*.
Ensure: Return dictionary containing concepts *dict*.

```
1: dict ← initializeClassDictionary ()
2: global dendro = createRandomDendrogramOfSize(w)
3: for ever
4:     sub ← getSubsequenceFromP()
5:     sub ← subsequenceProcessing(sub, dict)
6:     top ← frequentPatternMaintenance(sub)
7:     dict ← activeLearningSystem(top, dict)
8: return dict
```

In the next four subsections we expand our discussion of the class dictionary and the three major modules introduced above.

4.2.2 Class Dictionaries

We limit our representation of a class concept *i*, to a triple containing: a prototype time series C_i, its associated threshold, T_i, and $Count_i$, a counter to record how often we see sequences of this class. As shown in Figure 4.2.*right*, a class dictionary is a set of such concepts, represented by *M* triples.

Unlabeled objects that are within T_i of C_i under the Euclidean distance are classified as belonging to that class. Figure 4.2.*left* illustrates the representational power of our model. Note that because a single class could be represented by two or more templates with different thresholds (i.e. Weekend in the Figure 4.2.*right*), this representation can in principle approximate any decision boundary. It has been shown that for time series problems this simple model can be *very* completive with more complex models [50], at least in the case where both C_i and T_i are carefully set.

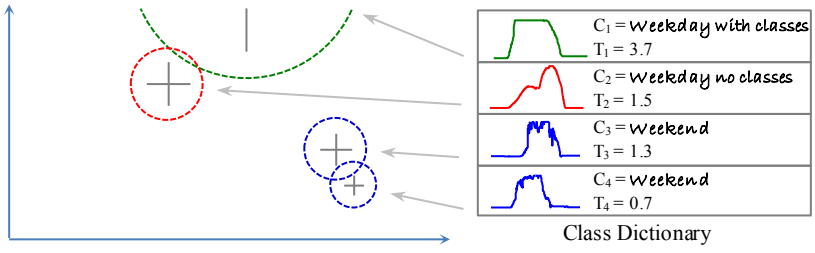

Figure 4.2: An illustration of the expressiveness of our model.

It is possible that the volumes that define two *different* classes could overlap (as C_1 and C_2 slightly do above), and that an unlabeled object could fall into the intersection. In

this case we assign the unlabeled object to the nearest center. We reiterate that this model is adopted for *simplicity*; nothing in our overall framework precludes more complex models, using different distance measures [33], using logical connectives, etc [71].

As shown in Algorithm 4.1 Line 1 our algorithm begins by initializing the class dictionary. In most cases it will be initialized as `empty`, however, in some cases we may have some domain knowledge we wish to "prime" the system with. For example, as shown in Figure 4.3, our experience in medical domains suggests that we should initialize our system to recognize and ignore the ubiquitous flat lines caused by battery/sensor failure, patient bed transfers, etc.

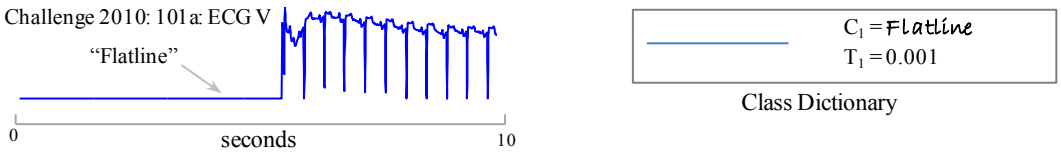

Figure 4.3: *left*) Sections of constant "flatline" signals are so common in medical domains that it is worth initializing the medical dictionaries with an example. *right*), thus suppressing the need to waste a query asking a teacher for a label for it.

Whatever the size of the *initial* dictionary, it can only increase by being appended to by the active learning module, as suggested in Line 7 of Algorithm 4.1 and explained in detail in Section 4.2.5.

4.2.3 Subsequence Processing

Subsequence processing refers to any domain specific preprocessing that must be done to prepare the data for the next stage (frequent pattern mining). We have already seen in Figure 1.2 and Figure 1.3 that z-normalization may be necessary [33]. More generally

this step could include downsampling, smoothing, wandering baseline removal, taking the derivative of the signal, filling in missing values, etc. In some domains very specialist processing may take place. For example, for ECG datasets, robust beat extraction algorithms exist that can detect and extract full individual heartbeats, and as we show in Section 4.3.3, converting from the time to the frequency domain may be required [15].

As shown in Algorithm 4.2-Line 3, after processing we attempt to classify the subsequence by comparing it to each time series in our dictionary, and assigning its class label to its nearest neighbor, if and only if it is within the appropriate threshold. If that is the case we increment the class counter, and the subsequence is simply discarded without passing it to the next stage.

Algorithm 4.2 **subsequnceProcessing** (*sub*, *dict*)

Require: subsequence time series *sub* from *P* and initialized dictionary *dict*.
Ensure: Return subsequence time series *sub* after processed.

```
1: sub ← domainDependentProcessing()
2: [dist, index] ← nearestNeighborInDictionary(sub, dict)
3: if dist < T_index     // Item can be classified
4:     disp ('An instance of calss 'index' was detected')
5:     count_index ← count_index + 1
6:     sub ← null    // return null to signal that no further processing is needed
7: return sub
```

Assuming the algorithm processes the subsequence and finds it is unknown, it passes it onto the next step of frequent pattern maintenance, which *is* completely domain independent.

4.2.4 Frequent Pattern Maintenance

As we discuss in more detail in the next section, any attempt to garner a label must have some cost, even if only CPU time. Thus, as hinted at in Figure 1.2/Figure 1.3 we plan to only ask for labels for patterns which appear to be repeated with some minimal fidelity. This reflects the intuition that a repeated pattern probably reflects some conserved concept that *could* be learned.

The need to detect repeated time series patterns opens a host of problems. Note that the problem of maintaining *discrete* frequent items from unbounded streams in bounded space is known to be unsolvable in general, and thus has opened up an active area of research in approximation algorithms for this task [29]. However, we have the more difficult task of maintaining *real-valued* and high dimensional frequent items. The change from discrete to real-valued causes two significant difficulties.

- **Meaningfulness**: We never expect two real-valued items to be equal, so how can we define a *frequent* time series?
- **Tractability**: The high dimensionality of the data objects, combined with the inability to avail of common techniques and representations for *discrete* frequent pattern mining (hashing, graphs, trees and lattices [29]) seems to bode ill for our hopes to produce a highly tractable algorithm.

Fortunately these issues are not as problematic as they may seem. Frequent item mining algorithms for discrete data must handle million-plus Hertz arrival rates [29]. However, most medical/human behavior domains have arrival rates are rarely more than

a few hundred Hertz. Likewise, for *meaningfulness*, a small Euclidean distance between two or more time series tells us that a pattern has been (approximately) repeated.

We begin with the intuition of our solution to these problems. For the moment imagine we can relax the space and time limitations, and we could buffer *all* the data seen thus far. Further imagine, as shown in Figure 4.4, that we build a dendrogram for all the data. Under this assumption, frequent real-valued patterns would show up as dense subtrees in the dendrogram.

Given this intuition, we have just two problems to solve. The first is to produce a concrete definition of "unusually dense subtree". The second problem is to efficiently maintain a dendrogram in constant space with unbounded streaming data.

While our constant space dendrogram can only approximate the results of the idealized ever-growing dendrogram, we have good reason to suspect this will be a good approximation. Consider the dense subtree shown in Figure 4.4; even if our constant space algorithm had thrown out any two of the four sequences in this clade, we would *still* have a dense subtree of size two that would be sufficient to report the existence of a repeated pattern. We will revisit this intuition with more rigor below.

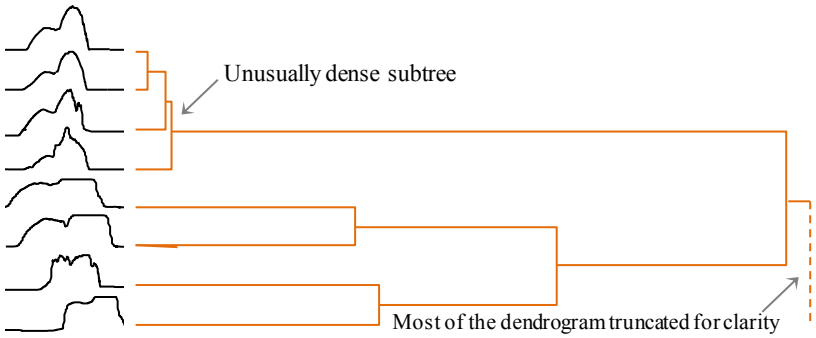

Figure 4.4: A visual intuition of our solution to the frequent time series subsequence problem. The elements in a dense subtree (or clade) can be seen as a frequent pattern.

We will maintain a dendrogram of size w in a buffer, where w is as large as possible given the space or (more likely) time limitations imposed by the domain. At most once per each time step[10], the Subsequence Processing Module will hand over a subsequence for consideration. After this happens a subsequence from the dendrogram will be randomly chosen to be discarded in order to maintain constant space. At all times our algorithm will maintain the top most significant patterns in the dendrogram, and it is *only* this top-1 motif that will be visible to the active learning module discussed below.

In order to define *most significant motif* more concretely we must first define one parameter, *MaxSubtreeSize*. The dense subtree shown in Figure 4.4 has four elements; a dense subtree may have fewer elements, as few as two. However, what should be the maximum allowed number of elements? If we allowed the maximum to be a significant fraction of w, the size of the dendrogram, we can permit pathological solutions as a subtree is only dense relative to the rest of the tree. Thus, we define *MaxSubtreeSize* to be

[10] Recall from Section 4.2.3 that the Subsequence Processing Module may choose to *discard* a subsequence rather than pass it to Frequent Pattern Maintenance.

a small constant. Empirically, the exact value does not matter, so we simply use six throughout this work.

We calculate the significance of the top motif in the following way. Offline, we take a sample time series from the domain in question and remove existing patterns by permuting the data. We use this "patternless" data to create multiple dendrograms with the same parameters we intend to monitor P under. We examine these dendrograms for all possible sizes of subtrees from two to *MaxSubtreeSize*, and as shown in Figure 4.5 we record the mean and standard deviation of the heights of these subtrees.

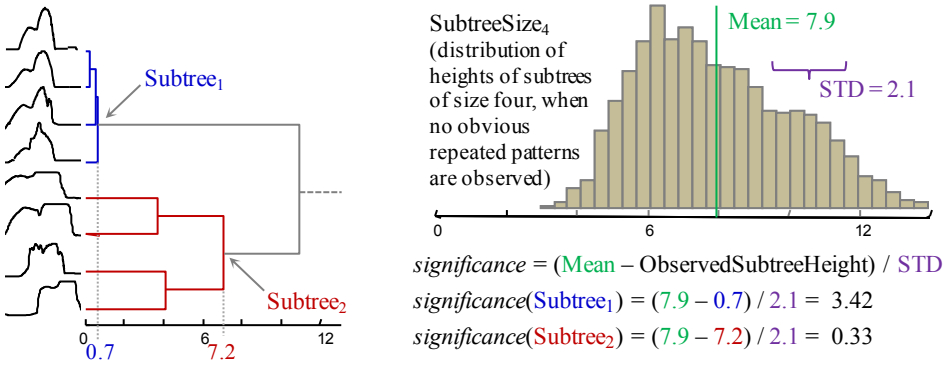

Figure 4.5: *left*) The (partial) dendrogram shown in Figure 4.4 has its subtrees of size four ranked by density. *right*) The observed heights of the subtrees are compared to the expected heights given the assumption of no patterns in the data.

These distributions tell us what we should expect to see if there are *no* frequent patterns in the new data stream P, as clusters of frequent patterns will show up as unusually dense subtrees. These distributions allow us to examine the subtrees of the currently maintained dendrogram and rank them according to their *significance*, which is simply defined as the number of standard deviations less than the mean is the height of the ancestor node. Thus, the *significance* of subtree$_i$ which is of size j is:

$$significance(Subtree_i) = \frac{mean(SubtreeSize_j) - SubtreeHeight_i}{STD(SubtreeSize_j)}$$

For example, in Figure 4.5.*right* we see that $Subtree_1$ has a score of 3.42, suggesting it is much denser than expected. Note that this measure makes different sized subtrees commensurate.

There are two issues we need to address to prevent pathological solutions.

- **Redundancy**: Consider Figure 4.5.*left*. If we report $Subtree_1$ as the most significant pattern, it would be fruitless to report a contained subtree of size two as the next most significant pattern. Thus, once we find the i^{th} most significant subtree, all its descendant and ancestor nodes are excluded from consideration for the i^{th}+1 to *K* most significant subtrees.
- **Overflow**: Suppose we are monitoring an accelerometer on an individual's leg. If she goes on a long walk, we might expect that single gait cycles might *flood* the dendrogram, and diminish our ability to detect other behaviors. Thus, we allow any subtree in the current list of the top *K* to grow upto *MaxSubtreeSize*. After that point, if a new instance is inserted into this subtree, we test to see which of the *MaxSubtreeSize* + 1 items can be discarsded to create the tightest subtree of size *MaxSubtreeSize*, and the outlying object is discarded.

In Algorithm 4.3 we illustrate a high level overview of the algorithm.

Algorithm 4.3 **frequentPatternMaintenance** (*sub*)

Require: subsequence time series *sub* from *P*.
Ensure: Return dictionary containing top k most significant concepts.

```
1: if sub == null     // If null was passed in, do nothing return
2:     top ← null; return
3: else
4:     dendro ← insert (dendro, sub)    // |dendro| is now w + 1
5:     top ← findMostSignificantSubtree(dendro)
6:     dendro ← discardALeafNode(dendro)    // |dendro| back to size w
7: return top
```

Our frequent pattern mining algorithm has only a single value, *w* the number of objects we can keep in the buffer, which affects its performance. This is not really a free *parameter*, as *w* should be set as large as possible, *given* the more restrictive of the time *or* space constraints. However, it is interesting to ask how large *w* needs to be to allow successful learning. A detailed analysis is perhaps worthy of its own paper, so we will content ourselves here with a brief intuition. Imagine a version of our problem, simplified by the following assumptions. One in one hundred subsequences in the data stream belong to the same pattern, everything else is random data. Moreover, assume that we can unambiguously recognize the pattern the moment we see any two examples of it. Under these assumptions how does the size of *w* effect how long we expect to wait to discover the pattern? Figure 4.6 shows this relationship for several values of *w*.

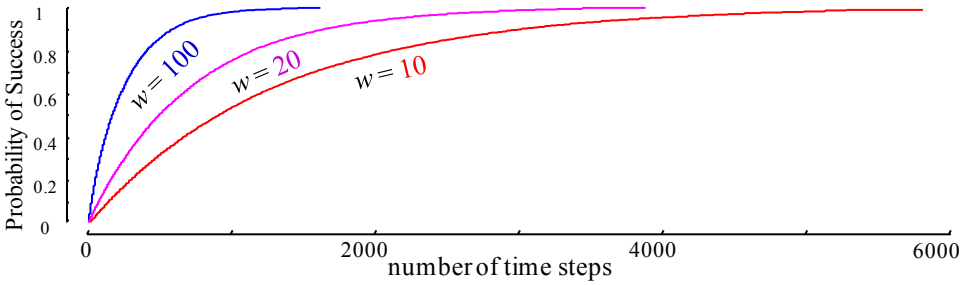

Figure 4.6: The average number of time steps required to find a repeated pattern with a desired probability, for various values of w. All curves end when they reach 99.5%.

If w is set to 10 we must wait about 5,935 time steps to have at least a 99.5% chance of finding the pattern. If we increase w by a factor of ten, our wait time does decrease, but only by a factor of 3.6. In other words, there are rapidly diminishing returns for larger and larger values of w. These results are borne out by experiments on real datasets (cf. Section 4.3). A pathologically small value for w, say $w = 2$, will almost never stumble on a repeated pattern. However, once we make w large enough, we can easily find repeated patterns, and making w larger again makes no perceptible difference. The good news is that "large enough" seems to be a surprisingly small number, of the order of a few hundred for the many diverse domains we consider. Such values are easily supported by off-the-shelf hardware or even smartphones. In particular, *all experiments* in this thesis are performed in real time on cheap commodity hardware.

Finally, we note that there are clearly exists real-world problems with extraordinarily rare patterns that would push the limits of our current naive implementation. However, it is important to note that our description was optimized for clarity of presentation and brevity, *not* efficiency. We can take advantage of recent research in online [8] and incremental [74] hierarchical clustering to bring the cost per time step down to $O(w)$.

4.2.5 Active Learning System

The active learning system which exploits the frequent patterns we discovered must be domain dependent. Nevertheless, we can classify two broad approaches, depending on the teacher (oracle) available. Teachers may be:

- **Strong Teachers** which are assumed to give correct and unambiguous class labels. Most, but not all, strong teachers are humans. Strong teachers are assumed to have a significant cost.
- **Weak Teachers** which are assumed to provide more tentative labels. Most, but not all weak teachers are assumed to be algorithms, however, they could be input of a crowdsourcing algorithm, or a classification algorithm that makes errors but performs above the default rate.

The ability of our algorithm to maintain frequently occurring time series opens a plethora of possibilities for active learning. Two common frameworks for active learning are Pool-Based sampling and Stream-Based sampling [84]. In Pool-Based sampling we assume there is a pool of unlabeled data available, and we may (at some cost) request a label for some instance, In Stream-Based sampling we are presented unlabeled examples one at a time and the learner must decide whether or not it is worth the cost to request its label. Our framework provides opportunities that can take advantage of both scenarios; we are *both* maintaining a pool of instances in the dendrogram, *and* we also see a continuous stream of unlabeled data.

Because this step is necessarily domain dependent, we will content ourselves here with giving real world examples, and defer creating a more general framework to future work.

Given our dictionary-based model the only question that remains in *when* we should trigger a query to the teacher, and *what action* we should take given the teacher's feedback.

4.2.5.1 When to trigger queries

Different assumptions about the teacher model and its associated costs can lead to different triggering mechanisms [84]. However, most frameworks can reduce to questions of how frequently we should ask questions. A conservative questioner that only asks questions rarely may miss opportunities to learn concepts, whereas an aggressive questioning policy will accumulate large costs and will frequently ask questions about data that are unlikely to represent any concept.

For any given domain, we assume that the teacher will tell us how many queries on average they are willing to answer in a given time period. For example, our cardiologist (c.f. Section 4.3.3) is willing to answer two queries per day from a system recording a healthy adult patient undergoing a routine sleep study, but twenty queries per day from a system monitoring a child in an ICU who has had recent increase in her SOFA score [37].

Let SR be the sampling rate of P, and QR be the mean number of seconds between queries that the teacher is willing to tolerate. We can then calculate the trigger threshold as:

$$triggerthreshold = -probit(1/(SR * QR))$$

We defer a detailed derivation to [5]. This equation assumes the distributions of heights of subtrees (e.g. Figure 4.5.*right*) are approximately Gaussian, a reasonable assumption when $j \ll w$.

4.2.5.2 Learning a concept: Strong teacher case

In Algorithm 4.4 the active learning system begins by comparing the *significance* (c.f. Section 4.2.4) of the top motif to this user supplied *trigger threshold*. If the motif warrants bothering the teacher, the `get_labels` function is invoked. The exact implementation of this is domain dependent, requiring the teacher to examine images, short audio or video snippets, or in one instantiation we discuss below, the bodies of insects, and provide labels for these objects. Once the labels have been obtained, then in Line 5 the dictionary is updated.

We have two tasks when updating the dictionary. First we must create the concept C_i, we can do this by either averaging the objects in the motif or choosing one randomly. Empirically, both perform about the same, which is unsurprising since the variance of the motif must be very low to pass the *trigger threshold*. The second thing we must do is decide on a value for threshold T_i. Here we could leverage off a wealth of recent advances in One-Class Classification [35], however, for simplicity we simply set the threshold T_i to three times the `top` subtree's height. As we shall see, this simple idea works so well that more sophisticated ideas are not warranted, at least the domains we investigated.

Algorithm 4.4 **activeLearningSystem** (*top*, *dict*)

Require: dictionary containing top k most significant concepts and the whole dictionary *dict*.
Ensure: Return dictionary containing concepts *dict*.

```
1: if (significance(top) < trigger threshold)    // The subtree is not worth investigating
2:     dict ← dict; return
3: elseif inStrongTeacherMode
4:     labels ← getLabels(top)
5:     dict ← updateDictionary(dict, top, labels)
6: else
7:     spawnWeakLearnerAgent(top)
8: return dict
```

4.2.5.3 Learning a concept: Weak teacher case

A weak teacher can leverage off side information. For concreteness we will give an illustration that closely matches an experiment we consider in Section 4.3.6, however, we envision a host of possible variants (hence our insistence that this phase be domain dependent). As illustrated in Figure 4.7.*top*, we can measure the X-axis acceleration on the wrist of the subject as he works with various tools. Moreover, RFID tags mounted on the tools can produce binary time series which record which tools are close to the user's hand, although these binary sensors clearly cannot encode any information about whether the tool is being used or carried or cleaned, etc. At some point our active learning algorithm is invoked in weak teacher mode with pattern C_1, which happens (although we do not know this) to correspond to an axe swing.

The weak teacher simply wait for future occurrences of the pattern to be observed, and then, as shown in Figure 4.7.*middle* immediately polls the binary sensors for clues as to C_1's label.

In the example in shown in Figure 4.7.*bottom* after the first detection of C_1 we have one vote for `Axe`, and one for `Cat`, and zero for `Bar`. However, by the third detection of C_1 we have seen three votes for `Axe`, one for Bar and one for Cat. Thus we can compute that the most likely label for C_1 is `Axe`, with a probability of 0.6 = 3 / (3 + 1 +1).

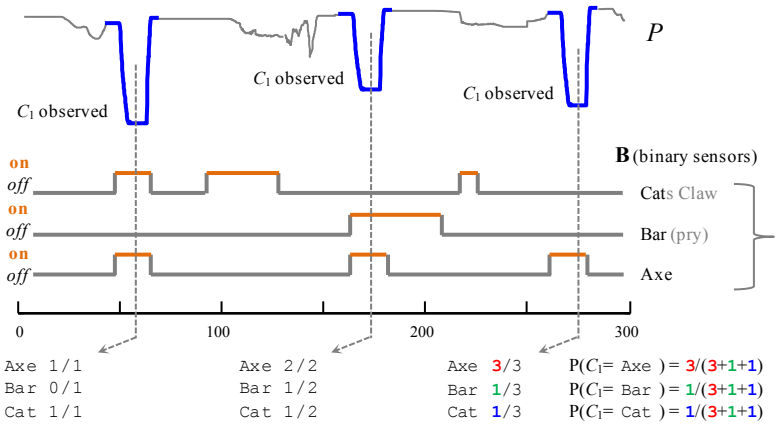

Figure 4.7: An illustration of a weak teacher. *top*) A stream P in which we detect three occurrences of the pattern C_1. *middle*) At the time of detection we poll a set of binary sensors to see which of them are active. *bottom*) we can use the frequency of associations between a pattern and binary "votes" to calculate probabilities for C_1's class label.

This simple weak teaching scheme is the one we use in this work and we empirically evaluate it in Section 4.3.6. However, we recognize that more sophisticated formulations can be developed. For example, our approach assumes that the binary sensors are mostly in the *off* position. A more robust method would look at the prior probability of a sensor's state, and the dependence between sensors. Our point here is simply to provide an existence proof of a system that can learn without human intervention.

Finally, note that the sensors polled do not have to be *natively* binary. They could be normally real-valued, for example, an accelerometer time series can be discretized to binary {*has moved in the last 10-sec*, *has not moved in the last 10-sec*}.

4.3 Experiments

We begin by noting that all code and data use in this thesis, together with addition details and many additional experiments are archived in perpetuity at [5].

While true never-ending learning systems are our ultimate goal, here we content ourselves with experiments that last from minutes to days. Our experiments are designed to demonstrate the vast range of problems we can apply our framework to.

We do not consider the effect of varying w on our results. As noted in Section 4.2.4, once it is set to a reasonable value (typically around 250) its value makes almost no difference, and we can process streams with such values in real-time for all the problems considered below.

Because our system discards subsequences randomly, where possible, we test each dataset 100 times and report the average performance. For each class, we report the number of times the class is learned, as well as the average precision and recall [79]. To compute the average precision and recall, we count in each run the number of true positives, false positives and false negatives *after* the class is first added to the dictionary.

4.3.1 Activity Data

We begin with a short but visually intuitive domain, the activity dataset of [89]. This dataset consists of a 13.3 minute 10-fps video sequence (thresholded to binary by the original authors) of an actor performing one of eight activities. From this data the original authors extracted 721 optical flow time series. We randomly chose just *one* of these time series to act as P, with **S** being the original video.

We set our trigger threshold to 3.5, which is the value that we expect to spawn about three requests for labels on each run, and we assume a label is given after a delay of ten second. Figure 4.8.*left* shows the first query shown to the teacher on the first run.

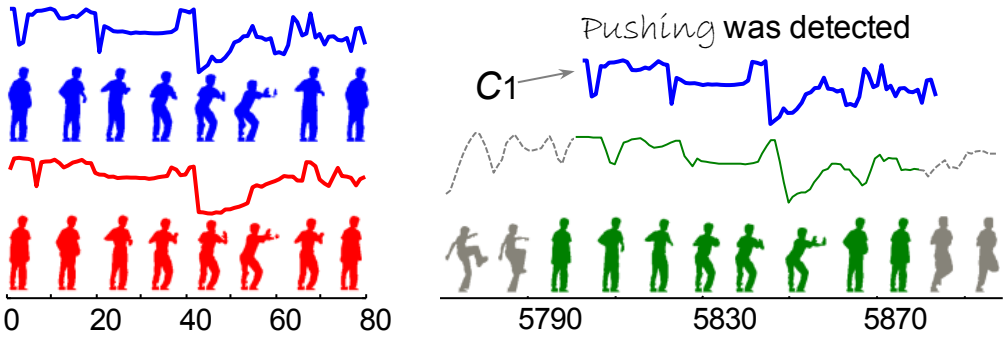

Figure 4.8: *left*) A query shown to the user during a run on the activity dataset, the teacher labeled it Pushing and a new concept C_1 was added to the dictionary. *right*) About 9.6 minutes later the classifier detected a new example of the class.

The teacher labeled this Pushing, and the concept was inserted into the dictionary. About 9.6 minutes later, this classifier correctly claimed to spot a new example of this class as shown in Figure 4.8.*right*. This dataset has the interesting property that the actor starts in a canonical pose and returns to it after completing the scripted action at eight-second intervals. This means that we can permute the data so long as we only "cut and

paste" at multiples of eight seconds. This allows us to test over one hundred runs and smooth our performance estimates.

Averaged over one hundred runs we achieved an impressive 41.8% precision and 87.96% recall on the **`running`** concept. On some other concepts we did not fare so well. For example, we only achieved 19.87% precision and 51.01% recall on the **`smoking`** concept. However, this class has much higher variably in its performance, and recall that we only used a *single* time series of the 721 available for this dataset.

4.3.2 Invasive Species of Flying Insects

Recently it has been shown that it is possible to accurately classify the *species*[11] 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 [15]. Figure 4.9 demonstrates the practicality of this idea.

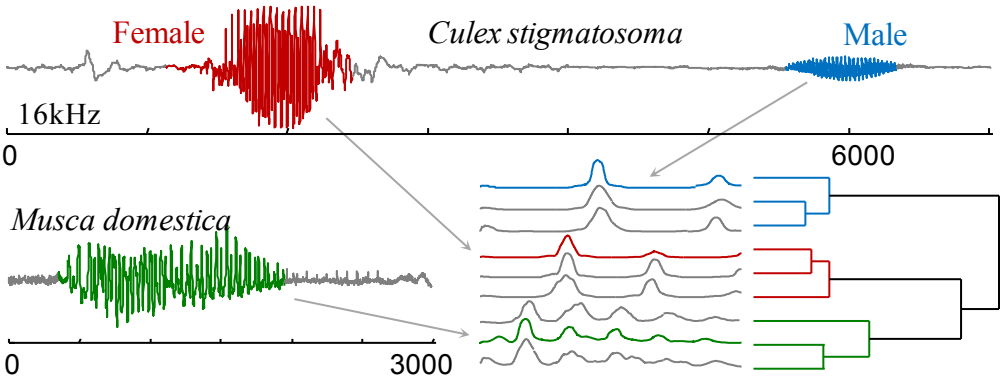

Figure 4.9: *top*) An audio snippet of a female *Cx. stigmatosoma* pursued by a male. *bottom left*) An audio snippet of a common house fly. *bottom right*) If we convert these sound snippets into periodograms we can cluster and classify the insects.

[11] And for some sexually dimorphic species such as mosquitoes, the *sex*.

This allows us to classify *known* species, for example 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; 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, about the size of the period at the end of this sentence.

Our solution is to exploit the fact that some insect traps can *physically* capture the flying insects themselves and recorded-their time of capture [69]. Thus, the **S** data source is 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 till she comes to physically inspect the trap, once a day.

Using insects raised from larvae in our lab we learned two concepts: *Culex stigmatosoma* male (C*stig*♂) and female (C*stig*♀). These concepts are just the periodograms shown in Figure 4.9 with the thresholds that maximized cross-validated accuracy.

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 cannot detect the invasive species, we increase their number for the next day, and try again until we do detect them. After we have detected the invasive species, the next day we release 500 of them with 500 *Cx. stigmatosoma* of each sex and measure 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 4.1 shows the results.

Number of insects before detection		Precision / Recall		
invasive species name		*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

Table 4.1: Our Ability to Detect then Classify Invasive Insects.

Recall that the results for C*stig*♂ and C*stig*♀ test only the representational power of 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.

4.3.3 Long Term Electrocardiogram

We investigated BIDMC Dataset ch07, a 20-hour long ECG recorded from a 48-year old male with severe congestive heart failure [1][41]. This record has 17,998,834 data points containing 92,584 heartbeats. As shown in Table 4.2, the heartbeats have been independently classified into five types.

Name	**Abbreviation**	**Frequency (%)**
Normal	N	97.752
R-on-T Premature Ventricular Contraction	r	1.909
Supraventricular Premature or Ectopic Beat	S	0.209
Premature Ventricular Contraction	V	0.104
Unclassifiable Beat	Q	0.025

Table 4.2: The ground truth frequencies of beats in $BIDMC_{ch}07$.

In Figure 4.10, we can see this data has both intermittent noise and a wandering baseline, we did *not* attempt to remove either.

Figure 4.10: A small snippet (0.0065%) of $BIDMC_{ch}07$ Lead 1.

Let us consider a single test run. After 45 seconds the system asked for a label for the pattern shown in Figure 4.11.*left*. Our teacher, Dr. Criley[12] gave the label Normal(N).

[12] Dr. John Michael Criley, MD, FACC, MACP is Professor Emeritus at the David Geffen School of Medicine at UCLA.

Just two minutes later the system asked for a label for the pattern shown in Figure 4.11.*center*, here Dr. Criley annotated the pattern as R-on-T PVC (r).

These two requests happened so quickly that the attending physician that hooked up the ECG apparatus will be in the same room and able to answer the queries directly. The next request for a label does not occur for another 9.5 hours, and we envision it being sent by email to the teacher. As shown in Figure 4.11.*right* our teacher labeled it PVC (V).

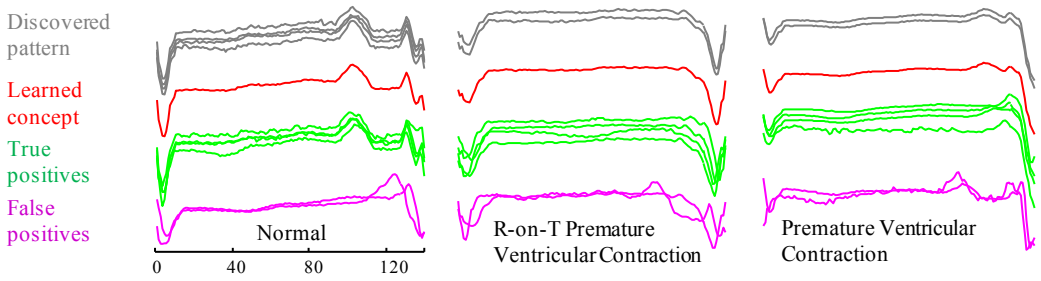

Figure 4.11: *left to right*) Three patterns discovered in our ECG experiment. *top to bottom*) The motif discovered and used to query the teacher. The learned concept. Some examples of true positives. Some examples of false positives.

In this run the class (S) was also learned, but just thirty minutes before the end of the experiment. We did not discover class (Q), however it is *extremely* rare and as hinted at by its name (Unclassifiable Beat) *very* diverse in it appearance.

Because the data has being independently annotated beat-by-beat by an algorithm, we can use this ground truth as a virtual teacher and run our algorithm 100 times to find the average precision and recall, as shown in Table 4.3. We note however that our cardiologist examined some of the "false positives" of our algorithm and declared them to be *true* positives, suggesting that some of the annotations on the original data are

incorrect. In fairness, [1] notes the data was "*prepared using an automated detector and has not been corrected manually*." Thus, we feel the numerical results here are pessimistic.

Class	**Detection Rate**	**Precision**	**Recall**
Normal (N)	100%	0.9978	0.9948
R-on-T PVC (r)	100%	0.9147	0.8080
Supraventricular (S)	100%	0.5028	0.4141
PVC (V)	100%	0.2342	0.6775
Unclassifiable (Q)	0%	-	-

Table 4.3: Results on $BIDMC_{ch}07$.

Beyond the objectively correct cardiac dysrhythmias discovered by our system, we frequently found our algorithm has the ability to surprise us. For example, after eighteen minutes of monitoring BIDMC-chf07-lead 2 [1], the algorithm asked for a label for the extraordinary repeated pattern shown in Figure 4.12.

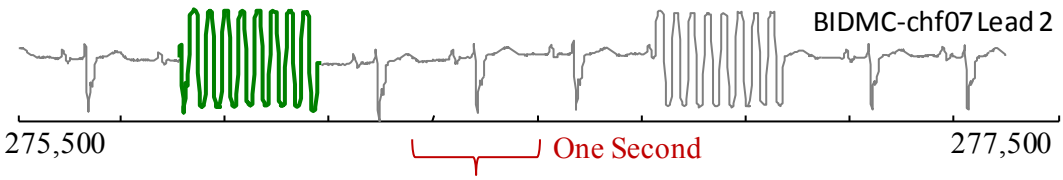

Figure 4.12: A pattern (green/bold) show with surrounding data for context, discovered in lead 2 of $BIDMC_{ch}07$.

The label given by the teacher, Dr. Criley was "Interference from nearby electrical apparatus: probably infusion pump". Having learned this label our algorithm detected fifty nine more occurrences of it in the remaining twenty hours of the trace. A careful retrospective examination of the data suggests that the algorithm had perfect precision/recall on this unexpected class.

4.3.4 Bird Song Classification

Recently a worldwide citizen science project called Bat Detective [2] has being using crowdsourcing to attempt to count bat populations by having volunteers classify sounds as one of {`bat`, `insect`, `mechanical`} (the latter class is a umbrella term for sounds created by human activities). In our efforts to volunteer for this project we noted that the majority of signals the system asked us to classify are wind noise or other low interest signals (see [5] for sample screenshots. We wondered if our framework would allow more useful queries to be sent to the users, thus making more effective use of their time.

We do not have ready access to bat sounds, so we produced a similar system for bird's sounds. To produce a dataset for which we had ground truth we did the following. We recorded an hour at midnight at the UCR botanical gardens on January 12 2012. A careful human annotation of the sound file reveals wind noise, voices in the distance, low volume rumbles from aircraft, etc, but no obvious wildlife calls. Using data from xeno-canto.org we randomly embedded ten examples of short (about 3 seconds) calls of a Tawny Owl in the data. Using the raw audio as **S**, and a single 100Hz Mel-Frequency Cepstral Coefficient (MFCC) as *P*, we ran our algorithm on this data. As Figure 4.13 shows our system can easily recover the patterns.

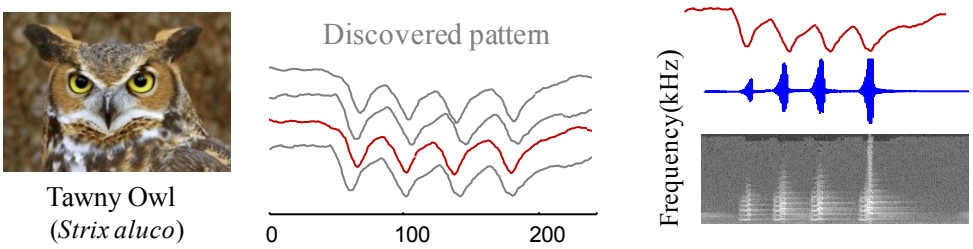

Figure 4.13: The motif discovered in the first run on the bird dataset. *right*) One snippet in three representations, bottom-to-top a spectrogram, an oscillogram and the MFCC we used.

The snippets may be heard at [5], they are easily identifiable as an owl, and however, it is less clear if an ornithological crowdsourcing community could identify them as a Tawny Owl.

4.3.5 Understanding Sapsucking Insect Behavior

Insects in the order *Homoptera* feed on plants by using a feeding tube called a stylet to suck out sap. This behavior is damaging to the plants, and it has been estimated that species in this order cause billions of dollars of damage to crops each year. Given their economic importance, hundreds of researchers study these insects, and increasingly they use a tool called an Electrical Penetration Graph (EPG), which as shown in Figure 4.14, adds the insect to an electrical circuit and measures the minuscule changes in voltage that occur as the insect feeds [56].

While there are now about ten widely agreed upon behaviors that experts can recognize in the EPG signals, little progress has been made in automatic classification in this domain. One reason for this is that the 32,000 species that make up order *Homoptera* are incredibly diverse; for example their size ranges over at least three orders of

magnitude. Thus, for many species an expert could claim of a given behavior "*I know it when I see it*", but they could not expect a template from even a related species to match.

As such, this is perfect application for our framework, and several leading experts on this apparatus agreed to help us by acting as teachers.

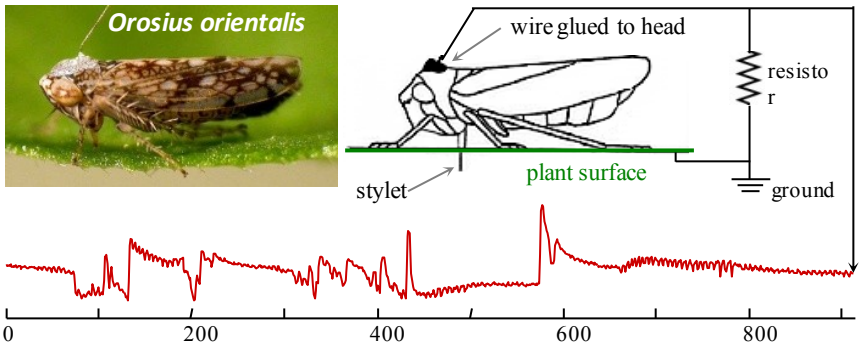

Figure 4.14: *left*) A tethered brown leafhopper. *right*) A schematic diagram of the circuit for recording EPGs. *bottom*) A snippet of data produced during one of our experiments.

Let us consider a typical run on a dataset consisting of a Beet Leafhopper (*Circulifer tenellus*) recorded by Dr. Greg Walker of UCR Entomology Department. Dr. Elaine Backus of the USDA, one of the co-inventors of the EPG apparatus, agreed to act as the teacher. She was *only* given access to the requests from our system; she could not see the whole time series or the insect itself. After 65 seconds the system requested a label for the three patterns shown in Figure 4.15.*top.left*. Dr. Backus labeled the pattern: phloem ingestion with interruption for salivation. After 13.2 minutes the system requested a label for behavior shown in Figure 4.15.*top.right*. Dr Backus labeled this pattern: transition from non-probing to probing. The former learned concept went on to classify

twenty-four examples, and the latter concept classified six. Examples of both can be seen in Figure 4.15.*bottom*.

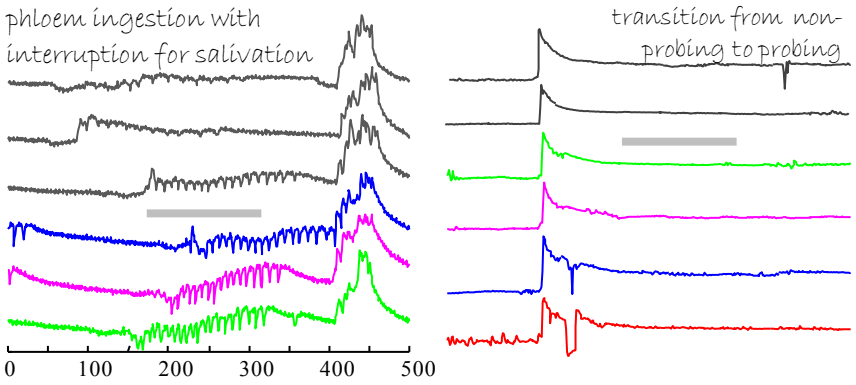

Figure 4.15: *top-row*) The two concepts discovered in the EPG data. *bottom-row*) Examples of classified patterns.

A careful retrospective study of this dataset suggests that we had perfect precision and recall on this run. Other runs on different datasets in this domain had similar success [5].

4.3.6 Weak Teaching Example: Elder Care

The use of sensors placed in the environment and/or on parts of human body has shown great potential in effective and unobtrusive long term monitoring and recognizing the activities of daily living [85][80]. However, labeling accelerometer and sensor data is still a great challenge and requires significant human intervention. In [80] the authors bemoaned the fact that high quality annotation is an order of magnitude slower than real-time, "*A 30-minutes video footage requires about 7-10 hours to be annotated.*" In this example we leverage off our weak teacher framework to explore how well the framework can label the sensor data without any human intervention.

We consider the dataset of [85] in which comes from an activity monitoring and recognition system is created using a 3D accelerometer and RFID tags mounted on household objects. A sensor containing both an RFID tag reader and a 3D accelerometer is mounted on the dominant wrist. Volunteers were asked to perform housekeeping activities in any order of their choosing to the natural distribution of activities in their daily life. Thus, the dataset is multidimensional time series with three real-valued and 38 binary dimensions.

For our experiment, we consider the just the X-axis acceleration sensor. The active learning algorithm is set in weak teacher mode. After 24 seconds the system finds a concept C_1, worth exploring (Figure 4.16.*top.left*). As we can see in Figure 4.16, our algorithm waited for the next occurrence of the pattern (it happens that three occur close together) and it polls the 38 binary RFID-detected sensors to see which are *on*.

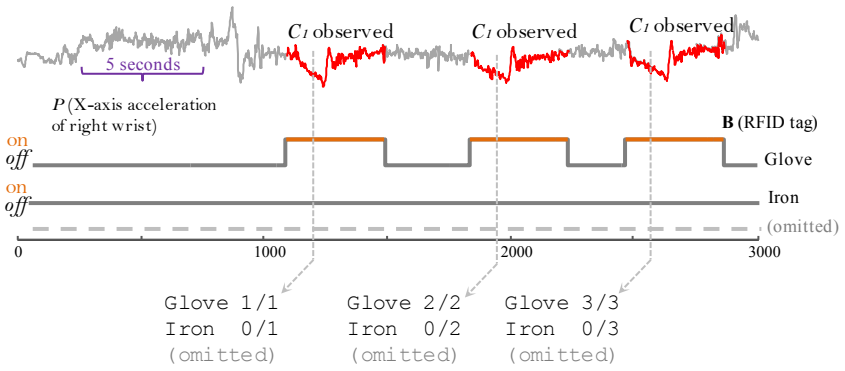

Figure 4.16: *top*) After we have learned the concept C_1 our system monitors for future occurrences of it. Here it sees three examples in a row. *bottom*) By polling the binary RFID sensors when a "hit" for C_1 is detected, we can learn that the concept is associated with 'glove'.

Our algorithm found an additional ten subsequences similar to the template. For six of these subsequences only the RFID tag sensor labeled `glove` was on. Of the remaining

four hits, just the `iron` was on for three times and just `fan` was on once. Thus, we end up with the probabilities shown in Figure 4.17.*right*.

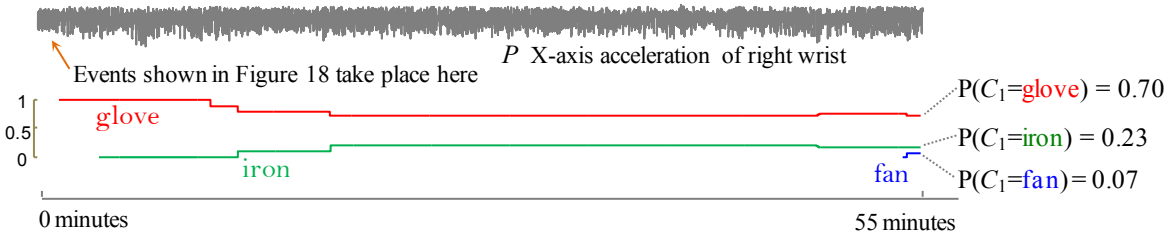

Figure 4.17: *top*) A zoom-out of the time series shown in Figure 4.16. *bottom*) The probability of concept C_1 been associated with various items. Of 38 possibilities only 3 have non zero entries.

In Figure 4.18, we show the relevant subsequences. Here, the true positives are subsequences that voted for `glove`, and the false positives voted for `iron` or `fan`. After a careful check of the original data we discovered that the pattern actually corresponds to *dishwashing*, which is the only behavior for which the participant wore `gloves`.

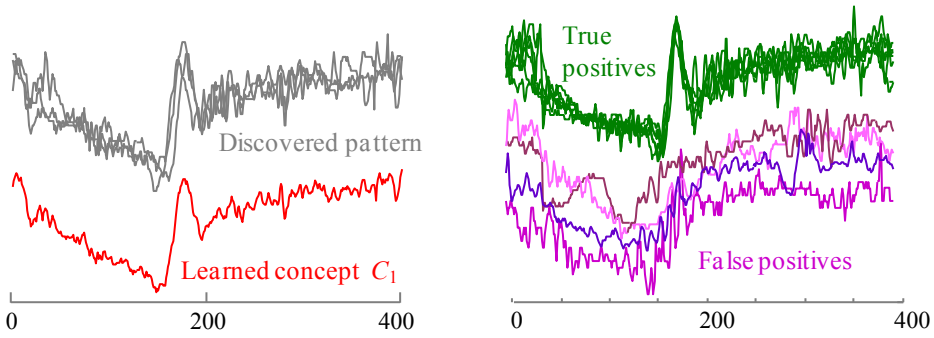

Figure 4.18: *top-left*) The motif discovered in our Elder care sensor experiment and averaged into concept C_1 (*bottom-left*). Examples of true positives and false positives.

4.4 Conclusion

In this chapter, we introduce the first never-ending framework for real value time series streams. We show our system is scalable, able to handle 250Hz with ease (cf. Section 4.3.3) and that it is robust to significant noise (cf. Figure 4.15 and Figure 4.18). Moreover, by applying it to very diverse domains we have shown it is a very general and flexible framework. In future work, we hope to remove the few assumption/parameters we have and apply our ideas to year-plus length streams. We have made all our code and data freely available [5] and hope to see our work built upon and applied to an even richer set of domains.

Chapter 5

Conclusion

Discovery of structures from rich, diverse data is a challenging task, and thanks to rapid improvements in data mining/machine learning community, many proposed robust system *can* effectively solve different domain problems. For example, audio processing is one of the most difficult but highly demanded tasks to be solved. Automatic speech recognition system, for example, has been applied to healthcare, military, telephony areas, and is very convenient for our daily lives. Speech recognition and music analysis have achieved impressive improvements so far, however, most research on monitoring animal sound to measure the health level of the ecosystem still just replies on human speech techniques, which are ill-suited for the task. Therefore, it is essential to build unified frameworks because tuning parameters is not easy for domain experts. Moreover, processing high speed streaming data efficiently is an essential task for many practical applications in various domains.

In this thesis, we first discussed the utilization of audio fingerprint, which is a powerful primitive representation to investigate animal sounds, and allows us to achieve different level species classification, and recognize rare species animals. Moreover, the

algorithm provides interpretable evaluation, therefore, can give domain experts better intuition of their disorganized data collections.

We introduced a scalable and general approach to find repeated patterns in large audio data archives by an anytime framework in Chapter 3, where we demonstrated our algorithm on various domains, such as finding audio motifs in human speech, music, laboratory mice vocalization, wild animal sounds, etc. We demonstrated that our algorithm can find semantically meaningful patterns in human speech, corresponding animal behaviors by checking the with domain experts.

Lastly, we introduced the first, scalable, never-ending learning framework for real valued time series streams, which could be a proxy for high dimensional data, such as audio, image, video, etc. We demonstrated our framework in diverse domains, including data generated from human motion capture/monitoring, ECG, EPG, animal sounds, etc.

Bibliography

[1] A. Goldberger et al. Physionet, accessed Feb-04-2013.

physionet.ph.biu.ac.il/physiobank/database/chfdb/

[2] K. Jones (2012). *www.batdetective.org*

[3] Macaulay Library, Cornell Lab of Ornithology, *www.macaulaylibrary.org/index.do*

[4] J. Roach, Cricket, Katydid Songs Are Best Clues to Species' Identities. National Geographic News, (URL), *news.nationalgeographic.com/news/2006/09/060905-crickets.html*

[5] Supporting webpage contains datasets, source code, slides, and other related resources. *www.cs.ucr.edu/~yhao*

[6] A. Vedaldi. *www.vlfeat.org/~vedaldi/index.html*

[7] Xeno-canto. *www.xeno-canto.org/*

[8] E. Achtert, C. Bohm, H-P. Kriegel, P. Kröger. Online Hierarchical Clustering in a Data Warehouse Environment Data Mining. *ICDM*, pp.10–17, 2005.

[9] S. E. Anderson, A. S. Dave, and D. Margoliash. Template-based automatic recognition of birdsong syllables from continuous recorderings. *Acoustic Society of America Journal*, 100: 1209-19, Aug 1996.

[10] I. Assent, P. Kranen, C. Baldauf, T. Seidl. AnyOut: Anytime Outlier Detection on Streaming Data. *DASFAA* (1), 228-242, 2012.

[11] J.-J. Aucouturier and M. Sandler. Finding repeating patterns in acoustic musical signals: applications for audio thumbnailing. *In AES 22nd Int' Conference*, 2002.

[12] M. E. Ayadi, M. S. Kamel, F. Karray. Survey on Speech emotion recognition: Features, classification schemes, and databases. *Pattern Recognition*. vol 44, 3, 572-587, 2011.

[13] S. Baluja, M. Covell. Waveprint Efficient wavelet-based audio fingerprinting. *Pattern Recognition* 41, 3467-80, 2008.

[14] R. Bardeli. Similarity search in animal sound databases. *IEEE Transactions on Multimedia*, vol. 11, no. 1, pp. 68–76, 2009.

[15] 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.

[16] Y. Beiderman, Y. Azani, Y. Cohen, C. Nisankoren, M. Teicher, V. Mico, J. Garcia, Z. Zalevsky. Cleaning and quality classification of optically recorded voice signals. *Recent Patents on Signal Proc*' 6-11, 2010.

[17] E. Berlin and K. Laerhoven. Detecting leisure activities with dense motif discovery. *Proceedings of the Intl Conference on Uniquitous Computing*. pp. 250-59, 2012.

[18] F. Bianconi, A. Fernandez. Evaluation of the effects of Gabor filter parameters on texture classification. *Pattern Recognition*, 40(12), 3325–35, 2007.

[19] D. T. Blumstein et.al. Acoustic monitoring in terrestrial environments using microphone arrays: applications, technological considerations, and prospectus. *J. Appl Ecol* 48:758–767, 2001.

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

[21] Bourvil. C'etait bien (*le petit bal perdu*). Lyrics: R. Nyel, Music: G. Verlor, *Editions Bagatelle*. 1961.

[22] J. C. Brown, P. Smaragdis. Hidden Markov and Gaussian mixture models for automatic call classification. *J. Acoust. Soc. Am*, (6):221–22, 2009.

[23] N. A. Butinov, Y. Knorozov. Preliminary Report on the Study of the Written Language of Easter Island. *Journal of the Polynesian Society* 66 (1): 5–17, 1957.

[24] B. J. L. Campana, E. J. Keogh. A compression-based distance measure for texture. *Statistical Analysis and Data Mining*, *SDM*, 3(6): 381-398, 2010.

[25] 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.

[26] P. Cano, E. Battle, T. Kalker, J. Haitsma. A review of audio fingerprinting. *The Journal of VLSI Signal Processing*, pp. 271-284, 2005.

[27] 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.

[28] R. Cole, J. Mariani, H. Uszkoreit, G. B. Varile, A. Zaenen, A. Zampolli. Survey of the State of the Art in Human Language Technology. *Cambridge University Press*, 1998.

[29] G. Cormode, M. Hadjieleftheriou. Methods for finding frequent items in data streams. *VLDB* J. 19(1): 3-20, 2010.

[30] T. Dang, N. Bulusu, W. C. Feng, W. Hu. RHA: A Robust Hybrid Architecture for Information Processing in Wireless Sensor Networks, In 6th *ISSNIP*, 2010.

[31] K. P. Dial, E. Greene, and D. J. Irschick. Allometry of behavior. *Trends in Ecology and Evolution* 23:394–401.

[32] C. Dietrich, F. Schwenker, G. Palm. Classification of Time Series Utilizing Temporal and Decision Fusion. *Proceedings of Multiple Classifier Systems (MCS)*, pp 378-87, 2001.

[33] 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.

[34] Dr. Seuss. The Cat in the Hat. ISBN 0-394-80001-X, *Random House*, 1957.

[35] C. Elkan and K. Noto: Learning classifiers from only positive and unlabeled data. *KDD*: pp213-220, 2008.

[36] L. Elliott, W. Hershberger. The Songs of Insects. *Houghton-Mifflin Company*, 2007

[37] F. Ferreira, D. Bota, A. Bross, C. Mélot, J Vincent. Serial evaluation of the SOFA score to predict outcome in critically ill patients. *JAMA* Oct 10, 286(14):1754-8, 2001.

[38] A. Fu, E. Keogh, L. Lau, C. A. Ratanamahatana, R. C.-W. Wong. Scaling and time warping in time series querying. *VLDB J*. 17(4): 899-921, 2008.

[39] P. Fussell. Poetic Meter and Poetic Form. *Random House*. 1965.

[40] C. M. Glaze, T. W. Troyer. Behavioral Measurements of a Temporally Precise Moto Code for Birdsong. *Journal of Neuroscience*, 27(29): 7631-7639, July 18, 2007.

[41] A. Goldberger et al. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation* 101(23):pp 215-20 2000.

[42] Y. Gong. Speech recognition in noisy environments: A survey. *Speech Communication* 16, 261-291, 1995.

[43] J. M. S. Grimsley, J. J. M. Monaghan, J. J. Wenstrup. Development of Social Vocalizations in Mice. *PLoS ONE* 6(3): e17460, 2007.

[44] R. J. Hagerman, et.al. Advances in the Treatment of Fragile X Syndrome. *Pediatrics* Vol. 123 No.1, January, pp 378–390 2009.

[45] J. Haitsma, T. Kalker. A Highly Robust Audio Fingerprinting System. *In Proceedings of International Conference on Music Information Retrieval*, 2002.

[46] N. C. Han, S. V. Muniandy, J. Dayou. Acoustic classification of Australian anurans based on hybrid spectral-entropy approach. *Applied Acoustic*. 72(9): 639-645, 2011.

[47] C. Herley. ARGOS: Automatically Extracting Repeating Objects from Multimedia Streams. *IEEE Transactions on multimedia*, Vol.8, No.1, February 2006.

[48] T. E. Holy, Z. Guo. Ultrasonic songs of male mice, *PLoS Biol* 3:e386, 2005.

[49] J.-L. Hsu, C.-C. Liu, A. L. P .Chen. Discovering nontrivial repeating patterns in music data. *IEEE Trans. Multimedia*, vol. 3, no. 3, pp. 311-25, Sep. 2001.

[50] B. Hu, Y. Chen and E. J. Keogh. Time Series Classification under More Realistic Assumptions. *SDM* 2013.

[51] B. Hu, T. Rakthanmanon, B. J. L. Campana, A. Mueen, E. J. Keogh. Image Mining of Historical Manuscripts to Establish Provenance. pp 804-815. *SDM* 2012.

[52] Y. Jang, HC. Gerhardt. Temperature Effects on the Temporal Properties of Calling Songs in the Crickets: Gryllus fultoni and G.vernalis: Implications for Reproductive Isolation in Sympatric Populations. *Journal of Insect Behavior*, Vol.20, No.1, 2007.

[53] A. Jansen, K. Church, H. Hermansky. Towards Spoken Term Discovery at Scale with Zero Resources. *INTERSPEECH*, 1676-1679, 2010.

[54] A. Jansen, B. V. Durme. Indexing Raw Acoustic Features for Scalable Zero Resource Search. *INTERSPEECH*, 2012.

[55] H. Jiang, T. Lin, H.-J. Zhang. Video segmentation with the assistance of audio content analysis. *In Proc. ICME*, New York, 2000.

[56] S. Jin, Z. Chen, E. Backus, X. Sun, B. Xiao. 2012. Characterization of EPG waveforms for the tea green leafhopper on tea plants and their correlation with stylet activities. *Journal of Insect Physiology*. 58:1235-1244.

[57] D. C. Karnopp. Random Search Techniques for Optimization Problems. *Automatica*, 1963.

[58] Y. Ke, D. Hoiem, R. Sukthankar. Computer Vision for Music Identification. *Proc. Computer Vision and Pattern Recognition, (CVPR)*, pp. 597-604, 2005.

[59] E. Keogh, S. Lonardi, C. A. Ratanamahatana, L. Wei, S. Lee, J. Handley. Compression-based data mining of sequential data, *DMKD*, 14(1): 99-129, 2007.

[60] J. A. Kogan, D. Margoliash. Automated recognition of bird song elements from continuous recordings using dynamic time warping and hidden markov models: a comparative study. *J. Acoust. Soc. Am*. 103(4):2185–219, 1998.

[61] P. Kranen, M. Hassani, T. Seidl: BT*- An Advanced Algorithm for Anytime Classification. *SSDBM*: 298-315, 2012.

[62] M. Li, X. Chen, X. Li, B. Ma, P. Vitanyi. The similarity metric. *Proc'of the 14th Symposium on Discrete Algorithms*, pp: 863 -72, 2003.

[63] D. G. Lowe. Distinctive Image Features from Scale Invariant Key Point. *International Journal of Computer Vision*, vol.60, pp. 91-110, 2004.

[64] J. B. MacQueen. Some Methods for Classification and Analysis of Multivariate Observations. *Proceedings of 5th Berkeley Symposium on Mathematical Statistics and Probability. University of California Press*. pp.281-297. MR0214227. ZbI0214.46201, 2009.

[65] R. Mankin, D. Hagstrum, M. Smith, A. Roda, M. Kairo. Perspective and Promise: a Century of Insect Acoustic Detection and Monitoring. *Amer. Entomol*. 57: 30-44.

[66] M. Marcarini, G. A. Williamson, L. de S. Garcia. Comparison of methods for automated recognition of avian nocturnal flight calls. *ICASSP*. 2029-32, 2008.

[67] D. K. Mellinger, C. W. Clark. Recognizing transient low-frequency whale sounds by spectrogram correlation. *J. Acoust. Soc. Am.*, 107: 6, pp. 3518-29, 2000.

[68] C. Menuet, Y. Cazals, C. Gestreau, P. Borghgraef, L. Gielis, et al. (2011) Age-Related Impairment of Ultrasonic Vocalization in Tau.P301L Mice: Possible Implication for Progressive Language Disorders. *PloS ONE* Jan; 6(10).

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

[70] D. Mitrovic, M. Zeppelzauer, C. Breiteneder. Discrimination and Retrieval of Animal Sounds. *In Proc. of IEEE Multimedia Modelling Conference*, Beijing, China, 339-343, 2006.

[71] A. Mueen, E. J. Keogh, N. Young: Logical-shapelets: an expressive primitive for time series classification. *KDD*: 1154-62, 2011.

[72] A. Mueen, E. J. Keogh, Q. Zhu, S. Cash, M. Brandon Westover. Exact Discovery of Time Series Motifs. *SDM* 2009: 473-484

[73] A. Celis-Murillo, J. L. Deppe, M. F. Allen. Using soundscape recordings to estimate bird species abundance, richness, and composition. *Journal of Field Ornithology*, 80, 64–78, 2009.

[74] S. Nassar, J. Sander, C. Cheng: Incremental and Effective Data Summarization for Dynamic Hierarchical Clustering. *SIGMOD* Conference: 467-478, 2004.

[75] D. J. Nowak, J. E. Pasek, R. A. Sequeira, D. E. Crane, V. C. Mastro. Potential effect of Anoplophora glabripennis on urban trees in the United States. *Journal of Entomology*. 94(1): 116-122, 2001.

[76] J. B. Panksepp, K. A. Jochman, J. U. Kim, J. J. Koy, E. D. Wilson, Q. Chen, C. R. Wilson, G. P. Lahvis. Affiliative behavior, ultrasonic communication and social reward are influenced by genetic variation in adolescent mice, *PLoS ONE* 4:e351, 2007.

[77] S. Pfeiffer, S. Fischer, W. Effelsberg. Automatic Audio Content Analysis. *ACM Multimedia* 1996.

[78] K. Riede, F. Nischk, C. Thiel, F. Schwenker. Automated annotation of Orthoptera songs: first results from analysing the DORSA sound repository, *Journal of Orthoptera Research*, 15(1), 105-113, 2006.

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

[80] D. Roggen et al. Collecting complex activity data sets in highly rich networked sensor environments, *In Proc' 7th IEEE INSS*, pp. 233-240, 2010.

[81] J. C. Ross, Vinutha T. P., P. Rao. Detecting Melodic Motifs from Audio for Hindustani Classical Music. *ISMIR*, 2012.

[82] A. I. Rudnicky, A. G. Hauptmann, K. F. Lee. Survey of Current Speech Technology. *Communications of ACM*, pp 52-57. 1994.

[83] M. L. Scattoni, S. U. Gandhy, L. Ricceri, J. N. Crawley. Unusual Repertoire of Vocalizations in the BTBR T+tf/J Mouse Model of Autism. *PLoS ONE* 3: e3067, 2008.

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

[85] M. Stikic, T. Huynh, K. V. Laerhoven and B. Schiele. ADL Recognition Based on the Combination of RFID and Accelerometer Sensing. PervasiveHealth, pp. 258–263, 2008.

[86] J. Sueur, S. Pavoine, O. Hamerlynck, S. Duvail. Rapid Acoustic Survey for Biodiversity Appraisal. *PloS ONE*. vol 3, 12, 2008.

[87] E. Trentin, M. Gori. A survey of hybrid ANN/HMM models for automatic speech recognition. *Neurocomputing*. vol 37, 1-4, 91-126, 2001.

[88] V. M. Trifa, L. Girod, T. Collier, D. T. Blumstein, C. E. Taylor. Automated Wildlife Monitoring Using Self- Sensor Networks Deployed in Natural Habits. *AROB* 2007.

[89] A. Veeraraghavan, R. Chellappa, A. K. Roy-Chowdhury, The Function Space of an Activity, *Computer Vision and Pattern Recognition*, 2006.

[90] L. Wahle. *Plants and Animals of Long Island Sound*. Sea Grant, CT-SG-90-11. 1990.

[91] M. M. Wells, C. S. Henry. Songs, reproductive isolation, and speciation in cryptic species of insect: a case study using green lacewings. *In Endless Forms: species and speciation*, *Oxford Univ. Press*, NY, 1998.

[92] X. Xi, K.Ueno, E. Keogh, D.J. Lee. Converting non-parametric distance-based classification to anytime algorithms. *Pattern Anal. Appl*. 11(3-4): 321-36, 2008.

[93] L.Ye and E.Keogh. Time series shapelets: a new primitive for data mining. *In Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD*, pages 947-956, 2009.

[94] G. Yu, J.-J. Slotine. Audio classification from time-frequency texture, *IEEE ICASSP*, pp. 1677-80, 2009.

[95] J. Zakaria, S. Rotschafer, A. Mueen, K. Razak, E. Keogh. Mining Massive Archives of Mice Sounds with Symbolized Representations. *SIAM SDM*, 2012. pp 588-599.

[96] D. Zaykovskiy. Survey of the Speech Recognition Techniques for Mobile Devices. *SPECOM*, 25-29. 2006.

[97] M. Zeppelzauer, A. S. Stöger, C. Breiteneder. Acoustic detection of elephant presence in noisy environments. *Proceedings of the 2nd ACM international workshop on Multimedia analysis for ecological data*. Pages 3-8, 2013.

[98] Y. Zhang, S. Rajagopalan, M. Salman. A Practical Approach for Belt Slip Detection in Automotive Electric Power Generation and Storage System. In Aerospace Conference, IEEE pp.1-7, 2010.