UNIVERSITY OF CALIFORNIA
RIVERSIDE

Pattern-Based Data Mining on Diverse Multimedia and Time Series Data

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

Doctor of Philosophy

in

Computer Science

by

Bilson Jake Campana

December 2012

The Dissertation of Bilson Jake Campana is approved:

Committee Chairperson

University of California, Riverside

Acknowledgements

I would like to begin by thanking my great friend and advisor, Dr. Eamonn Keogh. Over the years he has taught me priceless skills for both research and everyday life. He has forever changed the course of my life and helped me achieve a level of success that I would have never thought possible. I owe to him my eternal gratitude. I would also like to thank Dr. Walid Najjar and Dr. Victor Zordan. During my undergraduate studies their demonstrated knowledge and passion for their work inspired me to further my education and extend my goals. I would not have pursued this PhD without their influences.

I also wish to thank the graduate students within our department for all the shared pains of paper rejections and meals we ate in celebration. My time spent at UC Riverside with you all will always be a highlight of my life; I will definitely miss these good times (far between deadlines). Furthermore, I especially would like to thank my long time friends and family for their patience in supporting me during my degree. Without them I would have more gray hairs and less sanity.

I would also like to thank those who have funded my research: the National Science Foundation, Google, and the GAANN program, as well as donors of code and data.

Finally, I thank my Father and late Mother, Isabelo and Alphonsa Campana, my sister, Debbie, my loving wife, Lexiang, and my dog, Duke Kahanamoku II. Your patience, lessons, strength, and unyielding love and encouragement have been and always will be my driving force. Thank you all.

ABSTRACT OF THE DISSERTATION

Pattern-Based Mining of
Diverse Multimedia and Time Series Data

by

Bilson Jake Campana

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, December 2012
Dr. Eamonn J. Keogh, Chairperson

The ubiquity of patterns in data mining and knowledge discovery data sets is a binding characteristic across a diverse, and possibly otherwise unrelated, range of images, audio, video, and time series data. Despite the intra and inter distinctions of the data sets, there is usually the notion of a pattern within each data. These patterns may manifest as macro and micro textures in images, n-grams in text, motifs in time series, etc. Though despite this recurring trait, scientific studies on these data sets are an expansive history of varied methods, with new algorithms continuously presenting novel techniques and/or specialized parameters to adjust to their particular data. Because of the growing algorithmic complexities, efforts with new data then require an in depth review of its voluminous research background in order to optimize the selection of algorithm's sub-functions, feature spaces, and parameters.

Rather than providing data-dependent approaches which exist to cater to the variances in the data, this work leverages on the existence of patterns in many, if not all, data sets by using the data's pattern as its atomic form of representation. By forming our algorithms to operate on the data's patterns, all that is necessary for the application of these pattern-based methods on new, unseen data is an understanding the data's patterns; a term well understood by human intuition and abundantly expressed in the literature in many fields.

We first present a framework which provides an extremely accurate, fast, and parameter-less methods for measuring textural pattern similarity in images. We then demonstrate that this pattern-based method continues to be high performing across a large variety of image data sets from very diverse fields. We then show that it performs equally well in the realm of audio similarity. To further demonstrate the reach of pattern-based approaches, we present a novel method for the discovery of motif rules in time series; a pattern discovery problem where previous research efforts have been shown to deliver meaningless results. We then demonstrate optimizations for time series similarity search, a core subroutine to time series rule discovery and many other time series data mining algorithms.

Table of Contents

List of Figures

List of Tables

1 CHAPTER 1: PATTERN-BASED SIMILARITY METHODS

The field of data mining and knowledge discovery umbrella a large and diverse set of fields which analyze and operate on a plethora of data. These data exist in the form of time series, images, audio, text, and several other formats. Within a single format, the content has even a broader range. Audio may be the sound of song birds singing or a news broadcast. Time series may be monitoring the ever change stock market or weather throughout the world. An image may display microscopic organisms or feature vast landscapes. Yet despite the intra and inter distinctions of the data sets, there is usually the notion of a pattern within each data. The chirps of a song bird exhibit rhythmic tunes; weather follows seasonal patterns; Texture or an image can be described at multiple scales. Although data often exhibits this trait, scientific studies on these data sets are an expansive history of varied methods, with new algorithms continuously presenting novel techniques and/or specialized parameters to adjust to their particular data. Because of the growing algorithmic complexities, efforts with new data then require an in depth review of its voluminous research background. Because of the variance introduced by the new data set or application, novel methods or combinations of algorithms must be examined. All the while, the repeated patterns which define the characteristics of the data are easily overlooked.

Rather than providing data-dependent approaches which exist to cater to the variances in the data, our work leverages on the existence of patterns in many, if not all, data sets by using the data's pattern as its atomic form of representation. By forming our algorithms to operate on the data's patterns, all that is necessary for the application of these pattern-based knowledge discovery and data mining methods on new, unseen data is an understanding the data's patterns; a term well understood by human intuition and abundantly expressed in the literature in many fields. Towards this, we begin by presenting the CK method in section 2; a pattern-based image similarity framework. This framework describes a method which encapsulates well known and optimized video compression methods to measure repeated patterns between images. On top of this framework we then introduce the CK-1 measure. This measure provides an extremely accurate, fast, and parameter-less methods for measuring textural pattern similarity in images. We then demonstrate that this pattern-based similarity continues to be high performing across a large variety of image data sets from very diverse fields. Not only does our method regularly beat out strawman approaches, but it often does so by huge margin. Extending outside image data sets, we then briefly demonstrate that it performs equally well in for audio similarity; covering two distinct formats of data using a single, pattern-based method.

Beyond the realm of image and audio data sets, we apply pattern-based similarity to time series data in Section 3. Past attempts at discovery of relations between time series subsequences has proven to be faulty as they analyzed raw time series subsequences. To further demonstrate the reach of pattern-based approaches, we present a novel method for the discovery in time series which utilizes a motif-based symbolization. By focusing our rule discovery algorithm on found time series motifs, we are able to discover meaningful and actionable rules in the data. Unfortunately, this method is built on top of a costly motif discovery algorithm. We mitigate this scalability problem by introducing a suite of optimizations for subsequence similarity search; a core subroutine to time series rule discovery and many other time series data mining algorithms. This suite of optimizations provides speed up between 10x and 100x on a variety of data sets and allows us to conduct our pattern based optimizations.

2 CHAPTER 2: THE CK-METHOD

Texture analysis is used in classification, clustering, segmentation, and anomaly detection in images culled from domains as diverse as biology, medicine, robotics, biometrics, forensic science, and the study of historical texts. Texture recognition systems can have surprising uses; for example in Malaysia, a leading exporter of hardwoods, texture recognition is used to check against the logging of protected wood species and against attempts to pass off inferior strength species as stronger wood species for strength critical construction applications [1].

In the Content-Based Information Retrieval (CBIR) community, there has been extensive research in algorithms to measure texture similarity; however virtually all existing methods require the careful setting of many domain-specific parameters. For example, the commonly used Gabor filter requires the setting of scales, orientations, and filter mask size parameters [2][3]. As researchers have recently noted, "*Gabor filters show a strong dependence on a certain number of parameters, the values of which may significantly affect the outcome of the classification procedures*" [4].

Of the many problems associated with an abundance of parameters, the most obvious is simply that with many parameters to fit, it is exceptionally difficult to avoid over fitting [5]. An additional problem of parameter-laden algorithms is that they make it exceptionally difficult to reproduce published experimental results and to truly understand the contribution of a proposed algorithm [6].

In this work we propose to extend recent advances in Kolmogorov complexity-based similarity measures [6] [7][8][9] to texture matching problems. These Kolmogorov based methods have been shown to be very useful in intrinsically *discrete* domains such as DNA, natural languages, protein sequences, and symbolic music sequences such as MIDI or Parsons code; however, they are not defined for *real*-valued data, such as textures. We show that by approximating the Kolmogorov complexity with the Campana-Keogh (CK) method of using state-of-the-art video compressors, such as MPEG, we can create an efficient and robust texture similarity measure. To give our ideas a concrete grounding, we will discuss in detail two motivating examples.

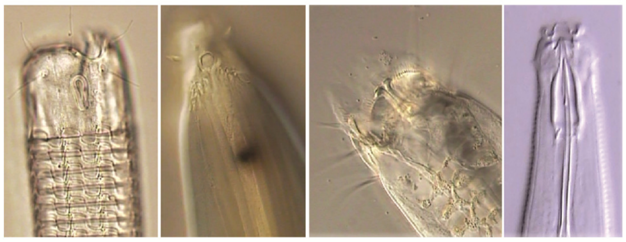
Figure 1: Examples of nematode diversity as seen under magnification.

Nematodes are a diverse phylum of "wormlike" animals, and one of the most diverse of all animal groups. Nematode species are very difficult to distinguish; over 80,000 have been described, however the true number may be closer to 500,000. As shown in Figure 1, nematode bodies are semi-transparent structures which mostly consist of digested foods and fat cells.

Understanding the biodiversity of nematodes is critical for several applications such as pest control, human health, and agriculture. For example, millions of people are infected by nematodes worldwide with a quarter of the world's population infected by a single genus of nematodes, *Ascaris* [10].

Because of their diversity and abundance, finding distinct characteristics of a nematode species for classification is a non-trivial task. Identification by experts requires three to five days to accomplish [11]. While the shape of the head and tail can be a useful feature in some cases, it is not enough to distinguish down to even the genus level. However, as we can see in Figure 1, nematodes are often richly textured, both externally and (given that they are semi-transparent) internally. As we shall show, our simple texture measure based on the CK method is extremely effective in classifying nematodes, without the need for careful parameter tuning or human-guided feature extraction.

Breast cancer results in about 500,000 deaths each year [12]. The survival rate of breast cancer patients greatly depends on an early diagnosis. In the US, survival rates of early diagnosed patients are 98%, where the survival rate of a regionally spread cancer is 84%, and those in a late stage where distant organs are effected have a survival rate of 28% [13]. Figure 2 displays an annotated image from the Mammographic Image Analysis Society mammogram database [14] with a malignant mass inscribed.

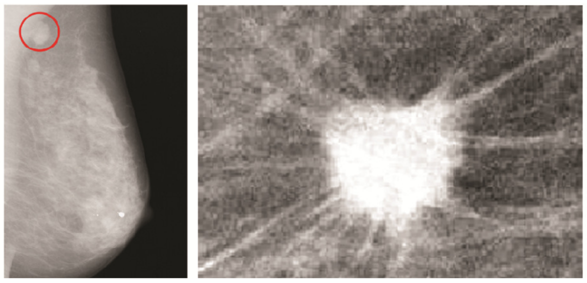
Figure 2: (*left*) A mammogram image with a malignant mass encircled. (*right*) Cancerous lesions tend to invade the surrounding tissue and exhibit a radiating pattern of linear spicules, resulting in unusual textures.

Numerous trials and evaluations have shown that mammography is the single most effective method for early detection of breast cancer and greatly increases chances of survival and treatment options[15][16][17]. Radiologists analyze mammograms for the existence of micro-calcifications, masses, asymmetries, and distortions which are hidden in a noisy texture of breast tissue, glands, and fat. Along with the noisy data, they must analyze large amounts of mammograms yearly [18], with only about 0.5% containing cancerous structures [19]. Because of the large amount of negative mammograms, radiologist may become less acclimated to detecting subtle signs of breast cancer. Computer aided diagnosis (CAD) provides a second look in the mammogram screen process. The radiologist is prompted with regions of interest which can increase classification accuracy and screening efficiency. Because the anomalies exist within highly homogenous fatty tissue and glands, it is a non-trivial task to detect and locate them. Texture analysis in this field allows for a detection method that does not depend on a distinctively shaped growth.

As we shall show in the experimental section, measures based on the CK method allows us to classify and cluster nematodes and other datasets with great accuracy and speed, without the need (indeed, without the ability) to fine tune many parameters. We further show the generality of our ideas with a comprehensive set of experiments.

The rest of this paper is organized as follows. Section 2 contains a discussion of related and background work. In Section 3 we introduce our novel CK method and, the MPEG-1 video compression employing, CK-1 measure. In Section 4 we give details of the most obvious rival methods before we consider the most extensive set of experiments ever attempted for texture measures, in Section 5. In section 6, we provide a speed performance evaluation for the presented methods along with a demonstration for embedding metrics. Finally, in Section 7 we offer conclusions and a discussion of avenues for future research.

2.1 Related and Background Work

In this section we overview past related work in image and texture analysis, Kolmogorov complexity, and compression based measures.

1) A Brief Review of Texture Measures

The measurement of texture similarity has a three-decade history and is still the subject of active research, see [20] and the references therein for an excellent overview. In essence, most methods reduce to some method to *extract* features combined with some measure to *compare* features.

These features can be *global scalars* such as energy, entropy, autocorrelation, standard deviation, etc., *global vectors* such as wavelet coefficients, Fourier coefficients, etc., or *local vectors/sets* such as SIFT descriptors, textons, etc.

The distance measures between the features are also highly variable, and include Euclidean distance, Kullback distance, Dynamic Time (histogram) Warping, and the Earth Movers Distance [21]. Note that if the feature vectors/feature sets can be of different lengths, then we are forced to use an "elastic" distance measure that allows non-linear mappings for comparison of features. Note that such measures invariably have at least quadratic time complexity [21], often with high constant factors.

Beyond computer science led research efforts, we have noted that many real-world practitioners in biological domains simply extract many features, feed them into a neural network, and hope for the best [3][22][23]. Our informal survey suggests that this use of neural networks is often a last resort effort that comes at the end of frustrated attempts to deal with the huge combination of features/measures. As we shall later show, the CK-1 measure typically outperforms these efforts with a technique that is much simpler and orders of magnitude faster.

2) Kolmogorov Complexity Inspired Distance Measures

The CK method is based on recent pragmatic work which exploits the theoretical concepts of Kolmogorov complexity. Kolmogorov complexity is a measure of randomness of strings based on their information content. It was proposed by A.N. Kolmogorov in 1965 to quantify the randomness of strings and other discrete objects in an objective manner.

The Kolmogorov complexity K(x) of a string x is defined as the length of the shortest program capable of producing x on a universal computer — such as a Turing machine. Different programming languages will give rise to distinct values of K(x), but one can prove that the differences are only up to a fixed additive constant. Intuitively, K(x) is the minimal quantity of information required to generate the string x by a program.

In order to define a distance based on the Kolmogorov complexity, the notion of conditional complexity is introduced. The conditional Kolmogorov complexity K(x|y) of x to y is defined as the length of the shortest program that computes x when y is given as an auxiliary input to the program. In [8], a distance is defined by comparing the conditional complexities K(x|y) and K(y|x) to K(xy), the latter of which is the length of the shortest program that outputs y concatenated to x. More precisely, the authors define the distance d_k between two strings x and y as:

2.2.1. $$d_k(x, y) = \frac{K(x \mid y) + K(y \mid x)}{K(xy)}$$

The distance measure is completely parameter-free (it is independent of the computer language used) and has been shown to be optimal [9] in the sense that it subsumes other measures. Unfortunately, the Kolmogorov complexity is incomputable for virtually all strings and thus must be approximated.

It is easy to see that universal compression algorithms give approximations to the Kolmogorov complexity. In fact, K(x) is the best compression that one could possibly achieve for the text string x. Given a data compression algorithm, we define C(x) as the size of the compressed *x* and *C*(*x*|*y*) as the compression size achieved by first training the compressor on *y*, and then compressing *x*. For example, if the compressor is based on a textual substitution method, one could build the dictionary on *y*, and then use that dictionary to compress *x*.

We can approximate the distance d_k by the following distance measure:

2.2.2. $$d_c(x, y) = \frac{C(x \mid y) + C(y \mid x)}{C(xy)}$$

The better the compression algorithm, the better the approximation of d_c is for d_k. In recent years this idea has been applied to domains as diverse as discovering the evolutionary histories of chain letters, spam classification, alignment-free comparison of biological sequences, protein structure classification [24], plagiarism detection [25], music genre classification, and a host of other problems [9].

Unfortunately, we cannot directly leverage on this body of work for two reasons. The first is that these ideas are only defined for *discrete* data, such as DNA strings or natural language. In these domains, a lossless compressor can really take advantage of repeated structure, which is exactly what we want to find to measure similarity. However, with the trivial exceptions such as cartoons/clip art, etc., most interesting images are *real*-valued. This difference is telling because lossless compression of discrete data is well defined and trivial to measure. In contrast, lossless compression of real-value images typically does reduce the sizes of the files greatly, but not in a way that finds repeated structure that is indicative of similarity.

The second reason we cannot directly use these ideas is more pragmatic. Calculating $C(x|y)$ requires a detailed understanding of the compression algorithm C, and actually "hacking" into it. While such work would not be beyond a reasonable attempt, it is not within the scope of effort for us in conducting this research. It would limit the adoption of our ideas, especially among domain experts that are not computer scientists.

To solve these two problems we propose a modification of the d_c (and therefore d_k) distance measure which treats a *lossy* compression algorithm as a complete black box, and which works for large, real-valued image data. In the Section 3 we expound these ideas.

3) Other Kolmogorov-Based Measures

To the best of our knowledge, this is the first work to consider compression-based distance measures for texture matching. A recent work considers a compression-based distance measure for *color* distributions in images [26], a paper by Li[1] and Zhu attempts image classification based on a kernel LZ78-based string kernel [27], Cilibrasi and Vitanyi create a compressor for clustering hand written text [28], and a recent work by Cerra and Datcu use a compression based measure for classifying satellite photographs [7].

However, beyond not explicitly considering texture, one thing all these works have in common is that they linearize the images into strings, and define distance measures based on strings. An obvious problem with converting a two-dimensional image into a one-dimensional string is that all spatial localization is lost. This may make no difference for color; however the very definition of texture is tied up with spatial patterns.

[3] This *Ming Li* [27] should not be confused with the *Ming Li* [8][9][30] who is a pioneer of Kolmogorov inspired distance measures.

A recent paper proposes a compression based measure for similarity retrieval of ornamental letters in historical manuscripts (although compression-based, the authors do not make the connection to Kolmogorov inspired methods) [29]. The distance measure is based on the similarity of the run-length-encoding representations of the data. While the idea is interesting, the measure requires careful alignment of the two objects being compared and is only defined for binary images. Either restriction would prevent us using the measure on 90% of the datasets we consider in this work.

2.2 The CK Method and CK-1 Measure

In this section we give the high-level intuition behind the CK method of utilizing video compression for texture analysis and the CK-1 distance measure which utilizes MPEG-1 video encoding. We then give the concrete algorithmic details and conclude with explicit implementation aspects.

1) Intuition behind our Method

Recall that our basic goal, motivated by the successful use of compression-based distance measures in discrete-valued data mining domains [6][8][9], is to somehow exploit compression for measuring texture similarity in real-valued images. Whatever solution we come up with, we are very hesitant to deeply "hack" into image compression code. This reluctance here is not mere sloth on our part, it is simply the case that difficult to implement ideas are rarely widely adopted. We feel that this is particularly true in this case, because much of our intended audience is biologists, nematologists, arachnologists, entomologists, etc. That is to say, people who may be comfortable using computer tools but are unlikely to have the time or the skills to write complex image compression code.

With this is mind we are motivated to use existing tools if possible. This leads us to consider measuring *image* similarity by exploiting *video* compression. Video is simply a three-dimensional array of images. Two dimensions, horizontal and vertical, serve as spatial image information directions of the moving pictures and the remaining dimension represents what is normally the time domain.

Virtually all video data contains significant amounts of spatial and temporal redundancy. Thus most video representations exploit these redundancies to reduce the file's size. Similarities are encoded by merely registering differences within a frame (intra frame compression), and/or between frames (inter frame compression). Our idea then is to exploit video compression for measuring the similarity of two images, simply by creating a synthetic "video" which is comprised of the two images to be compared. If those two images are indeed similar, the inter frame compression step should be able to exploit that to produce a smaller file size, which we will interpret as significant similarity.

While there are dozens of video formats in existence, we choose MPEG-1 and refer to its use with the CK method as the CK-1 measure. We utilize MPEG-1 encoding because of its widespread availability and the fact that all implementations of it tend to be highly optimized. In the next section we will review the necessary details of MPEG-1 encoding.

2) MPEG-1 Encoding

Because the MPEG-1 specification allows variable application based implementation of spatial redundancy reduction and motion vector calculation for temporal redundancy reduction [31][32], we choose to utilize the MPEG-1 encoder provided by MathWorks in Matlab for its simplicity and availability. We use a consistent set of encoder parameters based on empirically verified intuitions. These empirical tests have illustrated that deviation from the following encoder parameters has either drastically reduced classification accuracy or has only shown negligible improvement for a small subset of the data sets.

For speed and consistency, a logarithmic search algorithm is utilized for the inter frame block matching process. Original images for intra-picture reference frames are used to bypass their encoding step. The resulting full quality reference frame also allows for more detailed texture matching by creating a precise “dictionary” of textures from the original image. Since we are only interested in the compression ratios of the images rather than their visual presentation, large quantization scales for reference (I) and predicted (P) frames are selected to prefer compressibility over image quality. This down samples the images and removes subtle differences between textures that may simply be attributed to noise. Since there are no bidirectional (B) frames in our usage, their quantization factor is ignored.

The default Matlab search radius of 10 pixels is maintained. The bits used to specify block matched motion vectors have been limited to two. This modification is to allow for the possibility of an exhaustive block match search and global references which may be too distant from the query block (requiring more bits to reference than to store the original data), but has no affect on our reported results. The utility of global motion compensation and larger search spaces is further discussed in section 2.71).

3) Video Creation

In our function, *mpegSize*, we use the MPEG-1 encoder to construct a video of two images. This function requires two images which are converted to grayscale for color invariance. Each image is then transformed into a Matlab movie frame. Then, an ordered Matlab movie is constructed with these two frames. This Matlab movie is subsequently passed to the MPEG-1 encoder. For speedup, we modify the encoder to bypass disk writes and simply return the resulting size of the MPEG-1 movie. The first image supplied to *mpegSize* is assigned as an **I** frame and the second becomes a **P** frame. Because the second image is compressed to references of the first, this function is not symmetric.

4) CK-1 Distance Measure

As hinted at in Section 3.1, in order to measure the distance between two images we analyze compression ratios. Our measure is accomplished with a simple equation:

2.2.3. $$d_{CK}(x, y) = \frac{C(x \mid y) + C(y \mid x)}{C(x \mid x) + C(y \mid y)} - 1$$

The ratios in the denominator of 2.2.1 are calculated to measure the overhead required by the encoder, doing so gives us a baseline compression size when using any video encoder.

TABLE I. OUR PROPOSED DISTANCE MEASURE.

function distance = CK1Distance(x, y)	
1	distance = ((mpegSize(x, y) + mpegSize(y, x)) / (mpegSize(x, x) + mpegSize(y, y))) - 1;

As shown in TABLE I, this is executed on two images *x* and *y* by just a single line of Matlab code. Our CK-1 distance measure exhibits both positive definiteness and symmetry.

Positive Definiteness

The CK-1 distance measure exhibits non-negativity. Given the consistency of our *mpegSize* function, the *CK-1* distance of an image to itself will be zero. This property is important because many clustering algorithms rely on it to prove convergence properties.

Symmetry

As stated, our *mpegSize* function is not symmetric. To build a distance measure with symmetry, the bidirectional sum of the distances is taken in the numerator of 2.2.3 and the sum of the lower bounding sizes from "perfect" compressions are in the denominator.

In addition, preprocessing techniques can be applied to the images to introduce several additional invariances (rotation, illumination, color, etc.) to our approach. In our reported experiments, we refrain from utilize such techniques, besides rotation and color invariance, that attempt to tune and produce higher accuracies for our method.

Rotation Invariance

For rotation invariance we fix one image and rotate the other to find the minimum CK-1 distance between them. When an image is rotated not at a 90°, 180°, or 270° angle, the image no longer fits into its original rectangular dimensions and a sampling method must be used. Figure 3 demonstrates examples of image sampling methods used with rotated images.

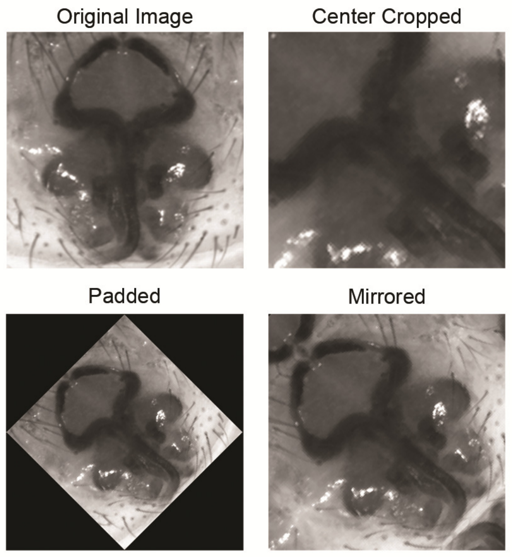

Figure 3: Examples of sampling and padding methods.

In our experiments we utilize three processes: no cropping, cropping to original image dimensions, and center cropping to a minimum bounding rectangle of valid pixels; black pixel padding or mirroring schemes are also used when rotations incur additional image pixels. Though different rotation methods provide better accuracies with different datasets, to avoid over fitting, we only report the accuracy provided by the center cropping method. For further simplification, we only consider ten rotations of the image in reported results; though our measure if fast enough to consider many more rotation degrees As noted in the main text we achieve rotation invariance by holding one image fixed and rotating the other. Since our measure is so fast we can quickly do this 360 times (once per degree) if necessary, however as hinted at in Figure 16, a coarser (and therefore faster search) is possible.

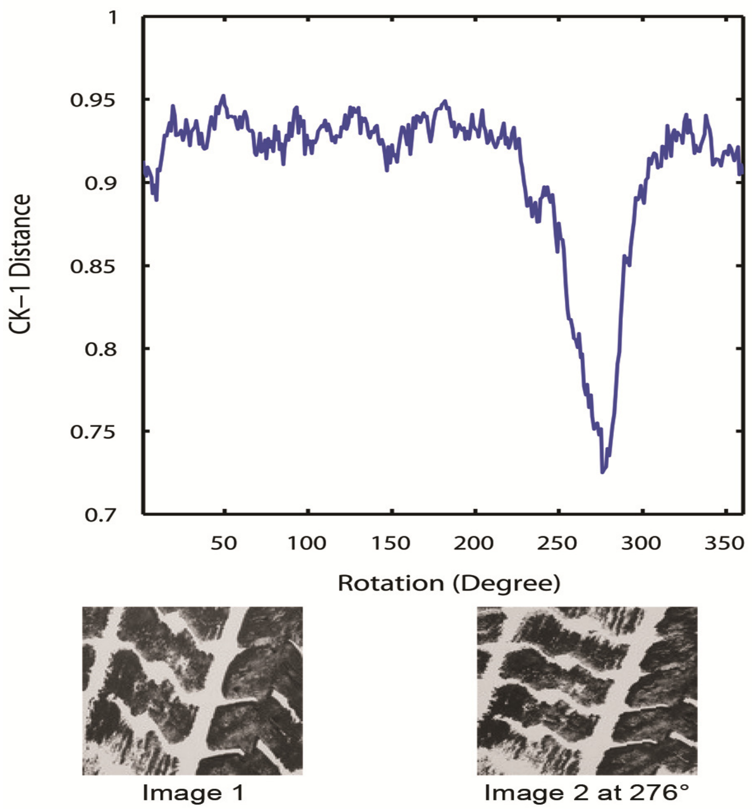

Figure 4: (*Top*) Measured CK-1 distance from image 1 to rotations of image 2. (*Bottom*) Center cropped images of image 1 and, optimally rotated, image 2.

Color Invariance

We remove color information and analyze the textures based on their gray scale intensity values. For datasets where color information is useful, we could combine the CK-1 measure with color features [33].

Illumination Invariance

Illumination between images may vary between photographs of samples, with different cameras, locations, and photographers. To remove the inconsistencies due to lighting we normalize the intensity values of the images. For local illumination invariance due to shadows from edges and surface texture, we can normalize the intensity values across an entire image. We can then normalize between two images for inter-image illumination invariance. For simplicity, our results presented in this paper refrain from exploiting any accuracy improvements provided by these preprocessing techniques.

2.3 Rival Methods

In this section we give concrete details of the most frequently used texture measures, as these will be the baseline to which we compare our ideas.

1) Filter Banks

The use of filter banks for feature extraction of textures has been motivated by their ability to be tuned to many diverse applications [3][22][35]. Their utility has allowed for a wide spread use in computer vision applications with many high-quality results. While there are many possible filter banks, the Gabor filter is by far the most commonly used. An overview of Gabor filters can be found in [2][4][36][37]. To generate our filters, a mother wavelet and generation function as presented in [37] is utilized. Filters of six orientations and four scales are generated, resulting in a filter bank of size $N = 24$ filters. High and low frequency parameters of the filters were set to the specifications found in [37].

Images are convolved with each filter. The standard deviation and mean of each response is then aggregated into a single 48 length vector. The distance between image descriptors can then be found from their Euclidean distance.

2) Textons

In order to fairly compare our method, we take the extra step of extending the previously described filter bank approach by classifying with a dictionary of representative filter responses, *textons*. Textons have been shown to be a great improvement over basic filter bank techniques [38][39]. Following the texton dictionary creation of [39], we represent each pixel of an image by a response vector of its corresponding outputs from each of the 24 filters. Response vectors from all images within a single class are then clustered into 10 groups using kmeans clustering, provided within Matlab, and the centroids of these clusters from each class are added to the texton dictionary. An image can then be represented by its histogram of response vectors binned to the nearest texton in the texton dictionary. The distance between two texton histograms is then found using the chi-squared distance.

2.4 Experimental Evaluation

We begin by stating our experimental philosophy. To ensure that our experiments are not just reproducible, but *easily* reproducible, we have built a website which contains all data and code, together with the raw spreadsheets for the results [40]. In addition, this website contains additional experimental details that are omitted here for brevity.

1) Sanity Check

We begin with simple experiments in domains where human intuition can directly judge the effectiveness of the CK-1 measure. We regard these as subjective demonstrations, rather than objective experiments (which will follow in Section 2)).

We clustered two sets of images, both of which have previously been used to test the utility of color and shape distance measures [33]. The two datasets are: *Heraldic shields* extracted from historical manuscripts from the 14th to 16th century, and *Insects* extracted from various amateur entomologists websites (used with permission). In both cases we selected 12 images which could be objectively grouped into six pairs, Figure 4 shows the results.

Compared to previous work, the results are unexpectedly good. In past work we had clustered (supersets) of these datasets based on color (shields) and color/shape (insects), but ignored the texture because we assumed it would not be very useful [33]. To our surprise, right "out-of-the-box" the compression-based measure works much better than our carefully tuned color/shape measure [33].

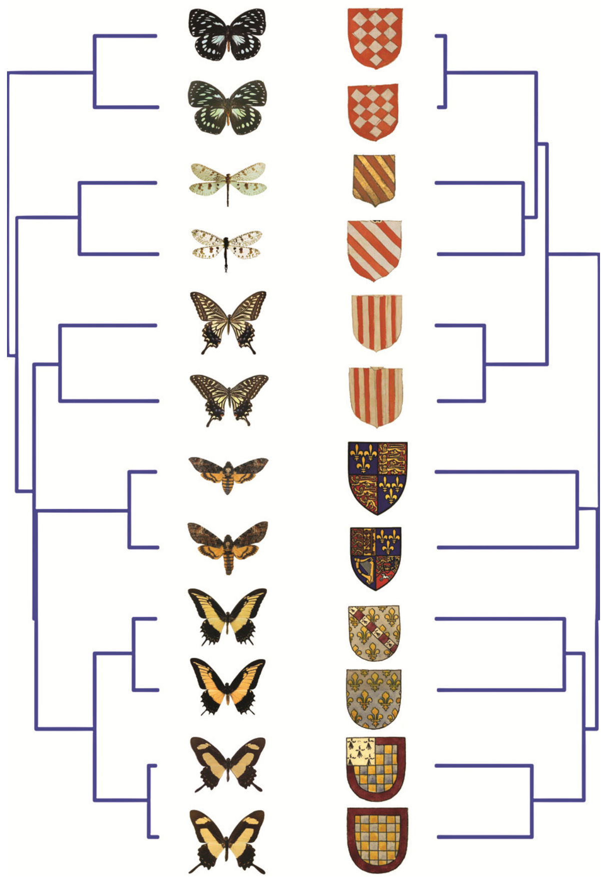

Figure 5: The *Insect* dataset and *Heraldic shields* datasets clustered with the CK-1 distance measure (average linkage clustering). While the images are shown in color for clarity, our distance measure had only access to the grayscale version of the images.

We might well have expected our measure to work very well in the richly textured domain of historical manuscripts, so we sought out two less obviously amenable datasets to test.

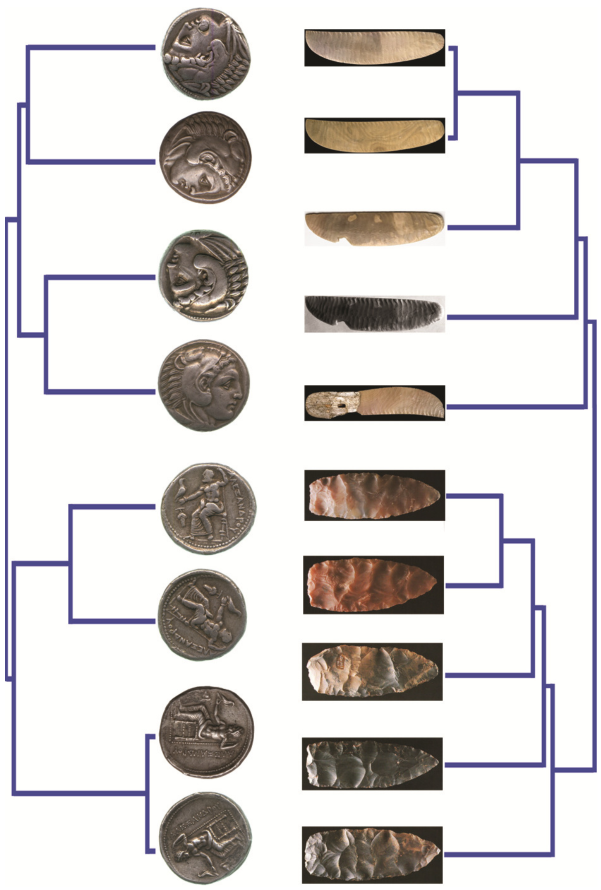

Figure 6: The Coin and the Egyptian Knives/Fenn Bifaces datasets clustered with the CK-1 distance measure (average linkage).

Figure 5 displays the clustering of the Coin and Egyptian Knives/Fenn Bifaces datasets. The coin dataset consists of coins from an image database provided by the Fitzwilliam Museum, Cambridge, UK [34]. All the coins are issued in the name of Alexander the Great who came to power in Macedonia in 336 BCE and died as emperor in 323 BCE. Some of the coins are from much later and were minted in places around the Black Sea, in

Egypt, in modern-day Turkey, Iran, etc. All coins follow the same basic standard: on the obverse side there is the head of Heracles in a lion-skin. The reverse side shows the god Zeus, seated left on a throne. Both obverse and reverse side of each coin is captured. Analyses of these coins are difficult due to their wear, unregistered rotational positioning, non-standardized lighting and photography, and the resulting reflectance and shadowing on the metal. The CK-1 measure clearly clusters the coins into their originating sides by utilizing rotation invariance described in Appendix B and by normalizing the image's grayscale values around its mean.

The grayscale normalization use for the coins is also applied to the Knives/Bifaces dataset to account for the varying lighting that creates shadows on the stone's carved grooves. At the first bifurcation the CK-1 measure correctly divides the dataset into the two known groupings, Egyptian vs. American blades. Whether the measure is of utility at the lower levels of the clustering is the subject of ongoing research.

2) Classification Experiments

In order to demonstrate the generality of our methods we have assembled the largest and most diverse collection of datasets ever considered in a single paper. For more details on these datasets we refer the interested reader to TABLE II, the supporting webpage [40], or the originating papers.

In TABLE II we numerically summarize the datasets. *Image quality* is a subjective measure of how "clean" the images are, for example do they have occlusions on the subject or camera shake.

TABLE II. DATASET DETAILS. DATASETS WITH VARYING IMAGE SIZE ARE LISTED WITH APPROXIMATED SIZES.

Data Set	Number of images	Number of classes	Image Size (Pixels)	Image Quality
Spider Subset	27	3	256x256	High
Full Spider Set	955	14	256x256	High
Tire Tracks	48	3	256x256	High
Nematodes	50	5	1440x1080	High
CAIRO Wood (F)	100	2	768x576	High
CAIRO Wood (S)	100	10	768x576	High
VTT Wood	200	2	~61x61	Medium
Original Moths	774	35	1280x960	Medium
Cropped Moths	774	35	800x800	Medium
Cleaned Moths	774	35	~500x800	High
Brodatz	1,792	112	128x128	High
KTH-TIPS	810	10	200x200	High
Camouflage	80	9	256x256	High
UIUCTex	1000	25	640x480	High
VisTex	334	19	512x512	High
Base Impressions	67	19	1201x900	Medium
Ornamental Letters (P/L)	168	42	~150x150	High
Ornamental Letters (L)	643	19	~150x150	High

Note that in every case we make these datasets publicly available (with the copyright remaining with the original creators were appropriate). The smaller datasets can be downloaded from the support webpage; the entire dataset can be obtained on two free DVDs by emailing the second author. In Figure 6 we show examples from each dataset.

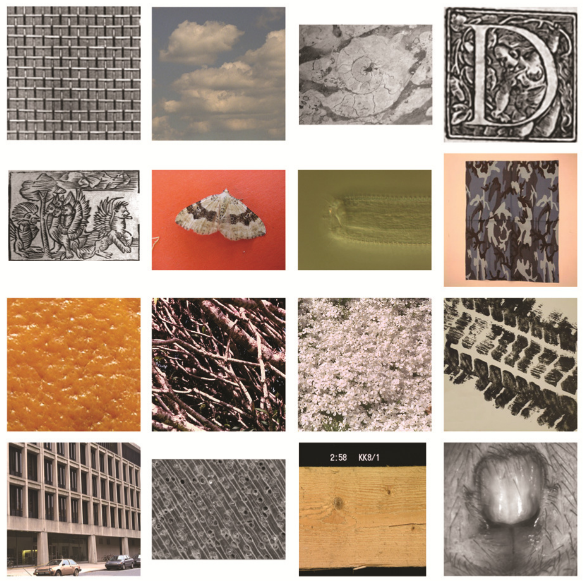

Figure 7: Samples of the datasets considered. A detailed key is omitted here for brevity, see [40] for further details.

Arachnology (Spiders): This dataset consists of images of the Australasian ground spiders of the family *Trochanteriidae*. This is a diverse family - 121 species in fourteen genera, with high variance in inter- and intra-specific variation, thus it represents a very difficult problem for classification. Although some species in this family are relatively common, almost 80 per cent were represented by less than ten individuals (of either sex); more than 50 per cent had fewer than five. Thirteen species had twenty or more individuals. The original images were grey scaled, cropped square, enhanced (for contrast/brightness) and resized by the original authors [3], we did no further pre-processing.

Moths (Macrolepidoptera): This collection [34] consisting of the images of live moth individuals, each moth belonging to one of 35 different species found in the British Isles. It is important to note that unlike most collections, which feature dead moths, carefully posed and photographed in ideal conditions in a lab, this datasets contains images of living moths photographed outdoors in a variety of conditions over a year. We consider three variants of this dataset: the original data, in which the moth occupies about 10% of the image area; center cropped, where an approximate bounding box was placed around the image; and a cleaned version, where the background was deleted with a semi-automatic technique.

Tire Treads: This dataset consists of a collection of tire imprints left on paper. Three well worn tires had paint applied to their treads and were rolled over paper. The tires are painted and rolled 16 times, each in varying directions and with different painted sections of the tire. Discontinuities in the painted tracks resulting from dry or insufficient paint resemble the interruptions in earth tracks caused by a denser arrangement of materials in the ground and uneven weight distribution across the tire.

Nematodes: As noted in the introduction, nematodes are a diverse phylum of "wormlike" animals, with great commercial and medical importance. The department of nematology at UCR, one of the leading institutions of in nematode research, has recently tasked us with creating a distance measure to help them sort through the largest archive of high-quality nematode images in the world [11]. For these experiments we consider a collection of fifty images of five species. Each nematode sample originally exists as a stack of images displaying over 100 focal planes of the organism. We prune the data by only selecting the focal plane image with highest variance in each sample stack (i.e., the most focused image).

Brodatz Textures: This dataset consists of a diverse set of images of man-made and natural textures (grass, straw, cloth, etc.), digitalized from images from a reference photographic album for artists and designers. While not a particularly interesting dataset, it is, by a huge margin, the most studied dataset in texture research. Unfortunately, there are many digitized variants of it available. Our version was obtained mostly from a publicly available online image database [41]. This set was missing slate 14, which we added directly from an original copy of the text [42] held at our campus library. For our experiments, we treat each image as a separate class and divided each image into sixteen non-overlapping, uniform images.

Ornamental Letters: This dataset contains a collection of ornamental letters from the Hand Press period. These letters are stamped from carved wooden blocks, this method of printing provides a "fingerprint" from each uniquely hand crafted block that can be used to track the origins of printed materials and also to analyze the history of the block's usage, wear, transfer between printing houses, and duplication when extensively damaged. Details about the dataset and information retrieval problems can be found in [43]. We present two classification problems using this dataset: letter based classification and print house origin classification.

Base Impression: The base impressions in this dataset are of same origin as the Ornamental Letters dataset. These images were used to signify a concept in the text or represent an important person, such as a king or saint. Their lineages can also be tracked by analyzing their subtle uniqueness. This dataset provides a significant challenge to our measure compared to the ornamental letters. Where the letters are very texture rich, these images may contain significant regions of blank space and are dominated by line based drawings. They may also contain noise in the form of letter bleed from the opposing page and fading from age. We classify these images based on their depicted scene.

CAIRO Wood: This dataset consists of 100 images of ten species of tropical wood provided by the Center for Artificial Intelligence and Robotics [44]. Each species is represented by ten photographs taken at a microscopic level. The images are also evenly split into two families of wood, *Leguminosae* and *Dipterocarpaceae*. The dataset is classified in two approaches: a two-class problem across family designations and a ten-class problem across species classifications. A similar set of this data has been worked on by [45].

Camouflage: This dataset consists of 70 images of nine varieties of modern US military camouflage. The images were created by photographing military t-shirts and fabrics at random orientations.

VVT Wood: This dataset consist of 839 samples of wood lumber used originally for color based inspection and grading for industrial usage [46]. Square tessellations of about 2.5x2.5cm of every image are annotated to be either sound or one of about 40 types of wood defect (dry knot, small knot, bark pocket, core stripe, etc.). The annotated data is parsed and each tessellated region is cropped and given a class label of either sound or defective. For classification tests, we use a subset consisting of 100 images from the two class problem: sound or not sound.

UIUCTEX: The University of Illinois at Urbana-Champaign Texture database [47] features twenty-five texture classes with forty samples each. The data set is composed of images of common textures such as glass, bark, and water. They are taken at varying orientations, illuminations, and subset locations on the sample texture.

VisTex: The MIT Vision Texture data set [48] consists of 167 images from 19 classes. Unlike many other texture datasets, does not hold rigid rules for orientation or lighting. Rather, it provides images from real world conditions such as flowers within a field or the water texture from an inland position.

KTH-TIPS: The KTH-TIPS [49] texture data set exists as an extension of the CURet data set [50] by adding variances in scale and by photographing from multiple samples in a single class. The dataset consists of 810 images from ten classes.

TABLE III. ACCURACY OF THE ONE-NEAREST-NEIGHBOR CLASSIFIER USING THE FOUR MEASURES UNDER CONSIDERATION. NOTE THAT RESULTS MAY BE BIASED TOWARDS THE TEXTON APPROACH.

Data Set	CK-1 (%)	Rotation Invariant CK-1 (%)	Gabor Filters (%)	Texton (%)
Spider Subset	96.3	-	59.6	89.6
Full Spider Set	93.5	-	39.1	74.1
Tire Tracks	79.2	91.7	87.5	93.8
Nematodes	56.0	-	38.0	52.0
CAIRO Wood (F)	83.0	94.0	95.0	95.0
CAIRO Wood (S)	77.0	90.0	93.0	94.0
VTT Wood	81.5	92.0	88.0	89.5
Original Moths	49.1	-	18.3	42.6
Cropped Moths	63.4	-	27.5	48.8
Cleaned Moths	71.0	-	24.0	58.2
Brodatz	52.1	44.8	37.0	52.0
KTH-TIPS	73.7	63.3	58.3	54.8
Camouflage	87.5	-	85.0	92.5
UIUCTex	51.0	43.6	45.3	55.8
VisTex	32.9	26.3	36.5	47.9
Base Impressions	98.5	-	8.96	19.4
Ornamental Letters (P/L)	100	-	12.5	48.2
Ornamental Letters (L)	90.7	-	21.0	45.4

TABLE III presents the best experimental results for these data sets with the CK-1 measure, the rotation invariant CK-1 measure (where appropriate), the Gabor filter bank method, and the texton method.

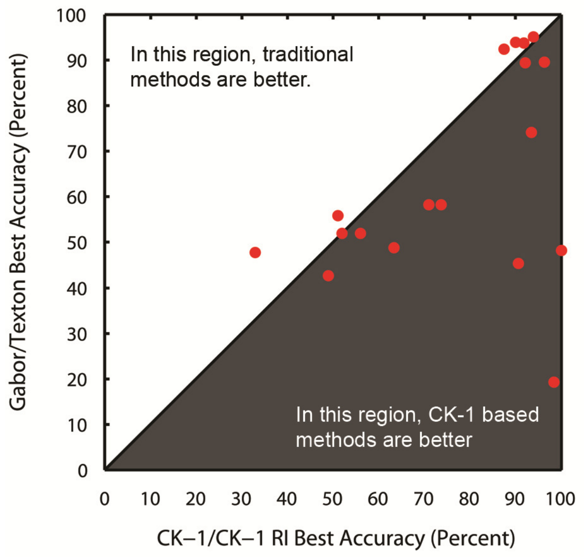

Figure 8: A visual summary of the relative strength effectiveness of our proposed distance measure.

Because the sheer number of results makes it difficult to judge the relative performance of the distance measures, we produced a figure to help visualize the results. For each dataset, we created a variable X = max(CK-1, *RI* CK-1), and a variable Y = max(Gabor Filters, Textons); we used these variables to plot a point for each dataset in Figure 7. Here we can see at a glance that the CK methods are extremely effective (Recall that classifications are biased towards the texton measure due to its learning on the entire dataset).

3) Analysis of Noisy Datasets

An obvious question for any distance measure is how robust is it to noise. While many of the datasets we have considered are noisy, we can best answer that question by systematically adding noise and testing its effect on classification accuracies. Noise may exist in the pixels of images due to low quality or inconsistent equipment, lighting environments, or difficulty in imaging the subject (i.e. live moths). Occlusions may also corrupt the quality of datasets and reduce the amount of measurable data in the image. To test for these effects we apply various types of noise and measure the resulting accuracy on the Cleaned Moths and CAIRO wood (species) datasets. These accuracies are averaged over ten runs with differing amounts of noisy images and percents of image content corruption. Figure 8 displays examples of a moth image with added noise.

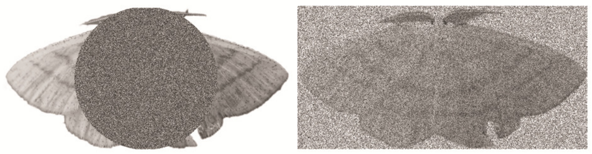

Figure 9: Examples of induced image noise: (*left*) center occlusion and (*right*) random pixel noise.

Center occlusion

In this test a circle of varying radius is drawn in the center of the selected images. This circle's pixels are randomly assigned a grayscale value to prevent the CK-1 measure from achieving perfect block matches in these noisy regions. The center position of the noisy circle is selected due to the high possibility of important information found at this location in many of our datasets. This test allows us to analyze CK-1's ability to make accurate matches with decreasing amounts of usable information. Note that the maximum circle diameter is that of the longest side of the image. Even at 100% size there may still be original data left in the corners of the image.

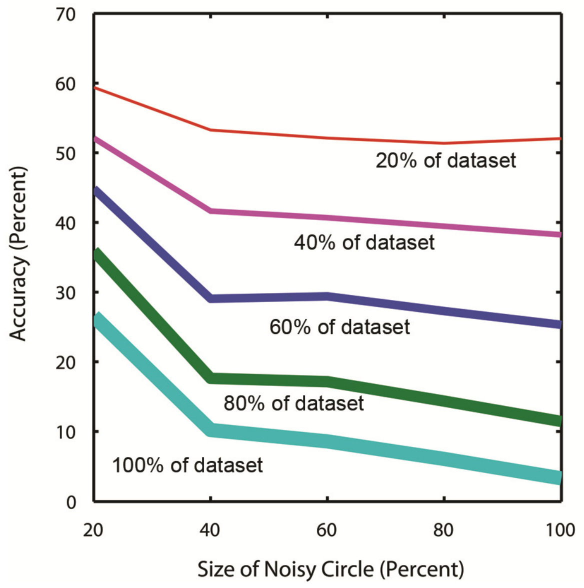

Figure 10: Cleaned Moths accuracies with center occlusion noise.

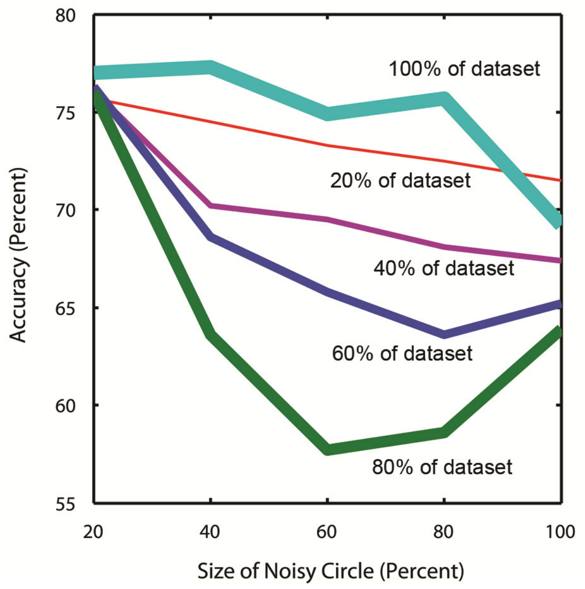

Figure 11: CAIRO dataset with center occlusion noise.

Figure 9 and Figure 10 present the accuracies of center occlusion tests on the Cleaned Moth and CAIRO wood (species) dataset. With the moths, the CK-1 measure is most variant to noise applied to an increasing portion of the dataset. Additional occluded images have a weaker effect with the CAIRO dataset. This is caused by the different nature of the subjects in each of the test sets. The subject of the moth dataset is an object, an insect, where the CAIRO set displays a texture of wood. Considering the moth results, there is a sharp decrease in accuracy as the occlusion's area begins to increase. This is because the object featured in the center of the image is quickly obstructed and only the background and fringe anatomical parts of the subject remain for analysis.

In the CAIRO set, the decrease is more gradual and may sometimes increase due to the characteristics of the wood texture which may have key features throughout the image. Up until about 20% occlusion there is little change in average accuracy, with the tests at 100% dataset corruption scoring equivalent to the dataset without any occlusion. After this threshold there is a sharp decrease as more blocks are left unmatched and begin to bulk the final video size. This demonstrates that CK-1 is invariant to small occlusions or noisy blocks in richly textured images.

As the number of corrupted images increase, more moths are removed from view for analysis and the cross validation performances continually decrease. With the CAIRO dataset, this phenomenon is the same until the dataset corruption reaches 100%. With the entire dataset exhibit the same occlusion noise, the occluded area is equally left out from all images and the texture in the images again becomes homogenous. All images have an equivalent amount of overhead due to the blocks in the occluded area and measurements on the remaining blocks can be weighed equally. With a small amount of occlusion at and below 20%, the corruption of additional images has little effect on the cross validation accuracies. This is another demonstration of CK-1's ability to handle small occlusion in high textured images.

Random pixel noise

To test CK-1's invariance to pixel noise, a given percentage of the pixels in an image are reassigned a random grayscale value.

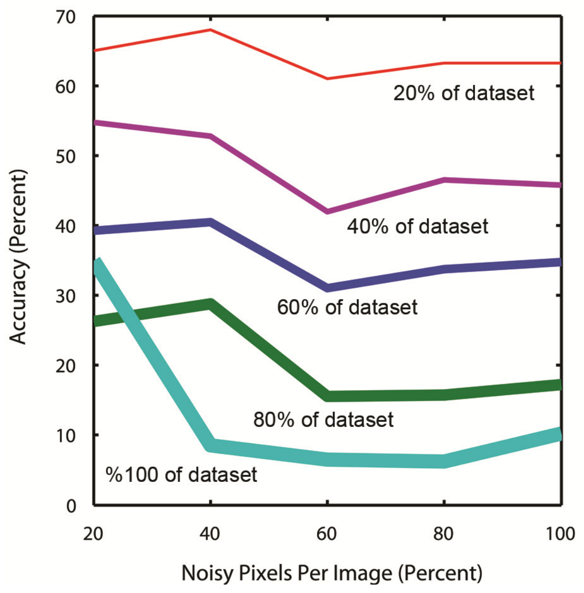

Figure 12: Cleaned Moth accuracies with random pixel noise.

Figure 11 and Figure 12 depict the one nearest neighbor cross validation performance of the CK-1 measure with the existence of random pixel noise on the Cleaned Moths and CAIRO (species) data sets.

With the moth dataset at 20% pixel noise and equal amount of images corrupted, the performance of the CK-1 measure drops almost 20%. Our measure remains roughly invariant to further pixel noise but drops by a half a percent for every additional percent of corrupted images. Though the quantization factors of MPEG-1 give the measure some resilience to independent pixel errors in the image, spreading the pixel corruption throughout the image, rather than localizing the errors into occlusions, can leave fewer unaffected blocks available for an accurate matching.

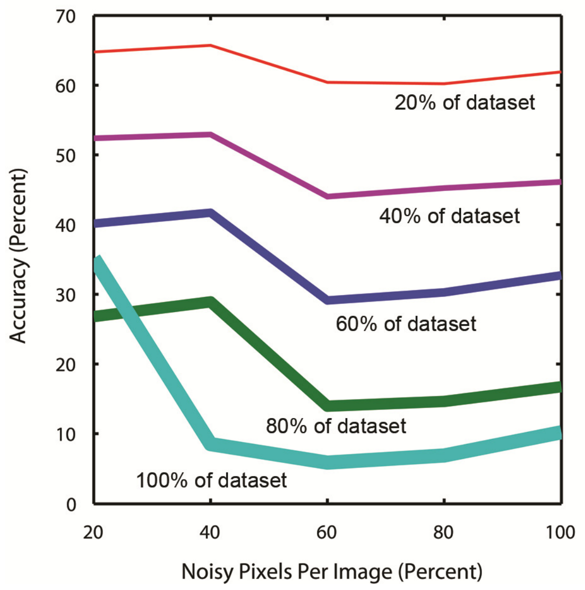

Figure 13: CAIRO dataset with random pixel noise.

Because the pixel noise is applied throughout the entire image, both test sets have similar performances as we vary the amount of dataset and pixel corruption. Just as with occluded blocks, blocks damaged with enough randomized pixels are less likely to be matched and therefore only add overhead for storing the entire block to the final video's size. At 100% dataset corruption and a small amount of pixel corruption, we observe slightly worse results as the amount of overhead from unmatched blocks is roughly equal in each image.

4) Similarity in Audio Space

Not only has the CK-1 method generalized extremely well throughout a diverse range of images, but we have extended this method to the problem of audio similarity [54]. In this distinct field, from visual images, of heard sound, we translate audio into an image space by analyzing its spectrogram using CK-1. With this approach we are successfully able to distinguish the calls of insects from diverse species.

5) An Application to Web Mining

We conclude our experiments with a simple example of a web mining application that can benefit from a robust texture measure. Our experiment *is* somewhat contrived, but demonstrates the robustness of the CK-1 distance to general unseen and unstructured data.

While gathering datasets for the classification experiments in the previous section, we noted we had a folder of moth images simply labeled *munda* (we know now the Genus name is *Orthosia*). Suppose we wished to retrieve more images of these moths from web, we can simply issue a Google image search. We did this on October 4th, 2009 and found that of the twenty-one images returned on the first page, none showed the correct moth. An image of the moth could not be found until the second page and the next image of the moth did not appear until the third page. As shown in Figure 13, the false positives include images of Munda Island and an unrelated insect that has the same specific name.

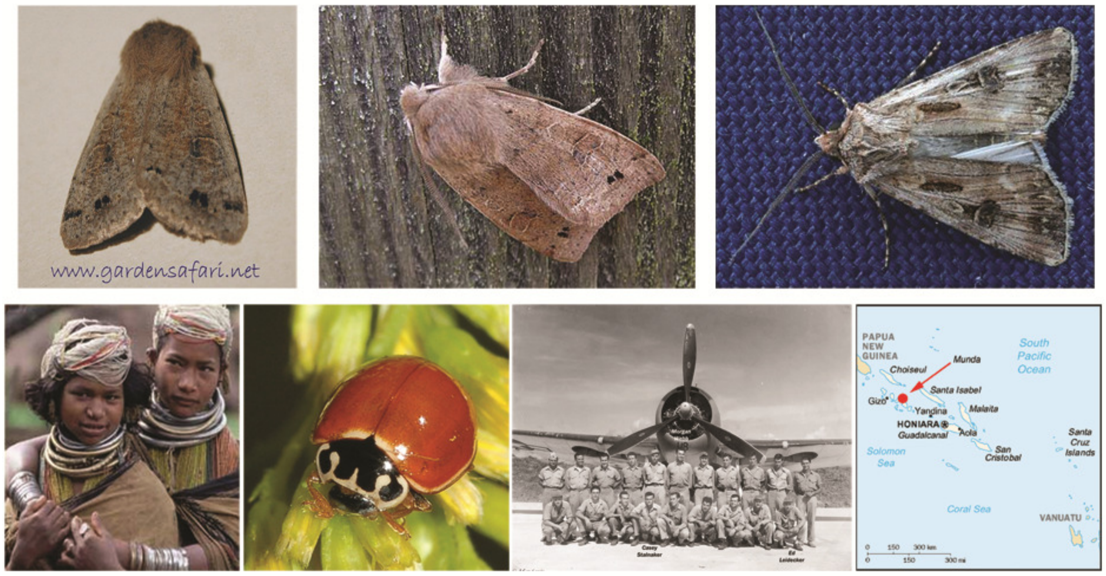

Figure 14: A web query for *munda* did produce some images of the moth, *Orthosia munda,* we expected (*top row*), but it also returned images of (*from bottom left to right*): the Munda tribes of India, an unrelated insect *Cycloneda munda*, a military photo taken at Munda Island, and a map of Munda Island.

For simplicity, let us consider the first four pages, which consist of 84 images, as the entire universe of images. Considering only these pages, there is a precision and recall of zero on the first page. There is an obvious way we could increase the precision of the query in the first page of results. Since we have some images of the moth we are interested in we could issue the text query as before, then reorder the query results based on their distance to a representative of our training data. This training representative is the training image with the lowest mean CK-1 distance to all other training images. We then score each query image based on their CK-1 distances to this training representative. This reordering brought about a recall of 1.0 and a precision of 0.19 on the first page.

2.5 Runtime Performance

The speed of our CK based method can be attributed to the simplicity of the underlying MPEG-1 compression algorithm. Since the reference image is not down sampled, there is no time required for its spatial redundancy reduction. The most time costly process, interframe block matching, is a logarithmic search process. Also, each block in the query image need only be compared to its corresponding neighborhood in the reference image. This greatly limits the running time needed to block match an entire image to O($nlogn$). Because the search can *early abandon* depending on the quality of a found match, this worst case runtime is usually avoided in empirical tests in favor of a fast average case runtime. Early abandoning also speeds up the calculation of the denominator in (1) by allowing for the block matching of the identity compression to be completely skipped. Furthermore, since most uses of MPEG involve large movies in the commercially important entertainment industry, the MPEG compression algorithms are extraordinarily well optimized.

In contrast, Gabor filters must convolve N filters for each image. The time performance of this operation must then also consider the dimension D of the square filters, where $D >> N$. The size of D depends on the scale and frequency parameters used in the filter generation and, in some cases, can be larger than the image itself. Just the Gabor descriptor extraction is therefore an O(n^2) operation.

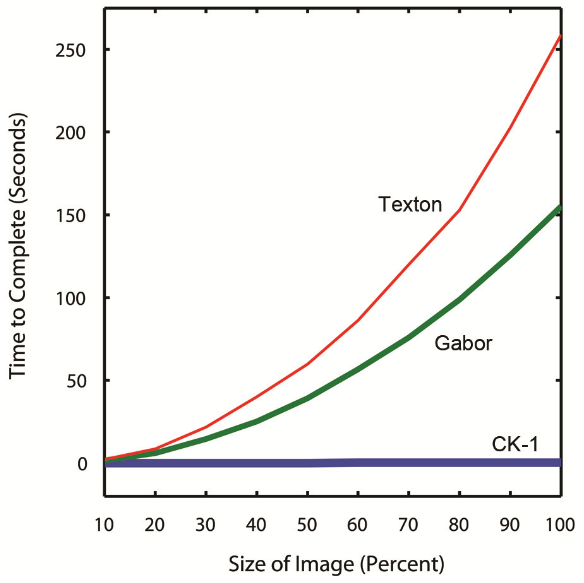

Figure 15: Time comparison on image size with CK-1, Gabor Filter Banks (GFB), and Texton approaches.

Textons add onto the running time of the original Gabor filters approach by requiring clustering within each class. Its runtime is bounded by O(n^2) + n x (*images per class)* x (*number of classes),* where each element to be clustered is of *N* dimensions. Texton calculation speed performance is therefore heavily dependent on its application. Large numbers of classes, large images, and large collections of images can greatly increase the execution time.

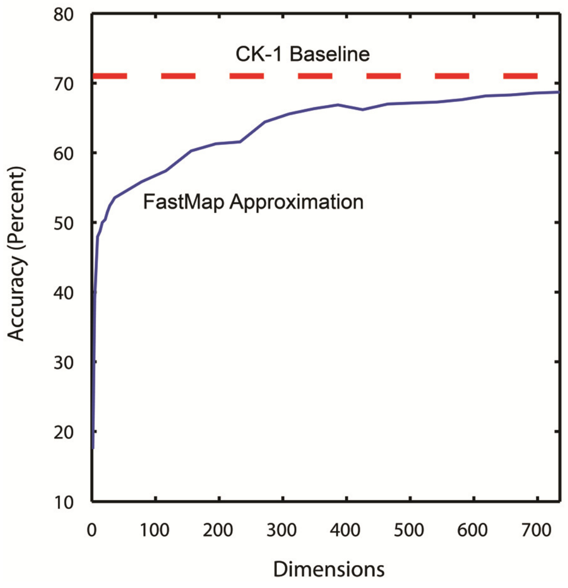

Figure 16: FastMap dimensional reduction on cleaned moth dataset.

As a concrete example: the distance between two images from the VisTex dataset, grass and brick, are compared with each of the three methods. The distances of ten scales of these images are computed and the average execution times over several iterations are plotted in Figure 14.

As we can see, the time taken for the CK-1 measure is negligible relative to the other measures.

1) Embedding Metrics

Though CK-1 is not a metric, it is possible to use it with techniques that are designed for exploiting metric properties. By extending our measure with these processes we can improve the speed and usability of CK-1. We demonstrate this by applying a fast dimensional reduction algorithm, FastMap [51], to our classification experiments.

FastMap can reduce the complexity of our query calculations while approximately preserving the dissimilarities of the original space. The process involves selecting two maximally distant pivot objects for each of the *k* dimensions of the reduced space. Transformation of new objects into this new space only requires comparison to these *2k* objects. For lookup a fast spatial method may be used for query lookups in the reduced space. Because we apply this algorithm to the pairwise distances of images in our datasets, we can dramatically decrease the amount of calculations in search for a good approximation of the object's nearest neighbor.

Figure 15 displays the resulting cross validation accuracies in a reduced FastMap space. With the FastMap accuracies approaching our baseline accuracy, we can trade performance accuracy for performance speed to fit any application.

2.6 Acknowledgements for the CK Method

This portion of work was funded by NSF 0808770. We would like to thank the many donors of data, especially of Anna Watson and Michael Mayo (moths), Melissa Yoder and Paul De Ley (nematodes), Kimberly Russell (spiders), Marzuki Khalid (CAIRO Wood), Frédéric Morain-Nicolier (historical manuscripts), Reinhold Huber-Mork (numismatics), and Tara Ramply (knives/arrowheads).

2.7 CK Conclussions and Future Work

In general the results in the previous section speak for themselves. For the most part, we have avoided comparisons to published results that consider the same datasets since different experimental conditions make direct comparisons difficult. However in some cases tentative comparisons can be instructive.

In the Spider Subset problem we got an accuracy of 96.3%, the original authors obtained accuracy in "*the range of 90–96%*" [3]. Note that this range of accuracy was obtained at the end of a four-year project devoted to just this problem, and their algorithm required occasional human intervention, "*it was important to review the log files of this process to pick out any potentially contaminating images and remove them from the training sets*" [3].

Of the variants of the Moth dataset, we obtained a best accuracy of 71.0%. Using two variants of the Nearest Neighbor algorithm (as we did), the original authors obtained 65.7 and 71.6% respectively [34]. However it is important to note that we used *only* texture features, whereas the original work had access to both color and texture features. It is clear that color is very useful in discriminating at least some of the classes. For example *Ourapteryx sambucaria* is yellow, whereas *Campaea margaritata* gets it common name, the Light Emerald moth, from its distinctive green hue, and *Cabera pusaria* is aptly known as the Common White Wave.

It is important to note that in spite of the generally excellent performance of the CK-1 measure in diverse domains, we are not claiming it is the best measure possible for all problems. For specialized application areas, better measures, which incorporate domain specific constraints and features, *may* do better. However for exploratory data mining, our CK-1 measure, built on our CK method, offers a powerful yet simple baseline measure.

1) Future Work

In this work we have not focused on the speed or indexability of the CK-1 measure, other than a tentative experiment to show that Fastmap embedding may be useful. One reason for this is that we wanted to forcefully demonstrate its *utility* first. In addition, we feel that optimizing speed may be irrelevant in many domains. Theo Pavlidis, one of the founders of CBIR recently remarked, "*In a medical application it may take well over an hour to produce an image, so waiting another hour to find matches in a database is not particularly onerous*" [52]. Such remarks apply to many of our domains; the moth dataset took almost a year to collect and the nematode dataset took four years to collect [11][34].

Nevertheless, as we have shown in Figure 14, the CK-1 measure is orders of magnitudes faster than some obvious rivals. We have shown a method to further increase the speed of our measure by allowing indexing with dimensionality reduction by embedded metrics. Using methods such as FastMap, we can achieve faster queries to fit various performance requirements, albeit at the cost of some reduction of accuracy.

In our analysis on the effects of noise in the form of occlusion and corrupted pixels, we have shown that CK-1 is surprisingly robust to noise. With 100% of the CAIRO dataset exhibiting occlusions, the cross validation accuracies outperform other tests including the original occlusion-free dataset.

There are several possibilities we plan to pursue. One possibility is to modify the *measure* so that it becomes a *metric*. This would allow us to avail of a wealth of techniques that exploit the triangular inequality to index data.

Even then there may be data mining applications for which we need to further improve efficiency. For example, within the next two years we expect to have terabytes of nematode images [11]. Further improvements in speed may come from exploiting several known ideas in image/video processing. For example multi-resolution analysis for scale invariance could improve our method's performances in many domains. More advanced compression algorithms could be explored to be used with the CK method for possible performance increases in speed and accuracy. Modifying the block matching search algorithm to allow for global motion vectors could allow for higher accuracies or faster search procedures and batch processing of multiple images. Possible options include the creation of a block matching algorithm specifically for the application of texture analysis, or to explore the global compensation techniques implemented in newer compression methods such as MPEG-4 and H.264 [53].

3 Chapter 3: Rule discovery in Time Series

The ability to make accurate predictions about future events is at the heart of much of science, and so it is not surprising that prediction/forecasting has been a topic of great interest in the data mining community for the last decade. Most of the work in the literature has dealt with *discrete* objects, such as keystrokes (i.e. predictive text), database queries [67], medical interventions [81], web clicks, etc. [80]. However, prediction may also have great utility in *real-valued* time series. For concreteness we briefly consider two examples:

- Researchers in robotic interaction have long noted the importance of short-term prediction of human initiated forces to allow a robot to plan its interaction with a human. For example, a recent paper notes the critical *"importance of the prediction of motion velocity and the anticipation of future perceived forces* [to allow the] *robot to <u>anticipate</u> the partner's intentions and adapt its motion.*" [61]
- Doppler radar technology introduced in the last two decades has increased the mean lead time for tornado warnings from 5.3 to 9.5 minutes, saving countless lives [54]. But progress seems to have stalled recently, with 26% of tornados within the US occurring with no warning. McGovern et al. argue that further improvements will come not from new sensors, but from yet-to-be-invented algorithms that "*examine existing data for <u>predictive</u> rules.*" [69]

Most of the current work has attempted to predict the future based on the current *value* of a stream [68]. However, for many problems the actual values are irrelevant, but the *shape* of the current pattern may foretell the future. For clarity we call the former *forecasting* and the latter, which is the subject of this paper, *rule-based prediction* (although the literature is inconsistent on this convention). There is an additional critical distinction between forecasting and rule-based prediction. Time series forecasting is typically *always-on*; it predicts values at every time step. In contrast, rule-based prediction *monitors* the incoming data at every time step, but only occasionally makes a prediction about an imminent occurrence of a pattern.

While *forecasting* is mature enough to have its own conferences and commercial software (SAS, IBM Cognos, etc.), the handful of research efforts to consider time series rule-based prediction have met with limited success. In particular, it is widely accepted that these efforts allow the discovery of spurious rules [62]. We believe that the reason why rule discovery in real-valued time series has failed thus far is that most efforts have more or less blindly applied the ideas of *symbolic* stream rule discovery to *real-valued* rule discovery. In this work, we argue that the classic ideas of support/confidence are not directly transferable to rule discovery in real-valued time series. Instead, we formulate a rule representation and search strategy that evaluates candidate rules based on how well they can compress the data. We also demonstrate the (lack of) effect of using our simple uniform quantization method. Beyond our novel definitions/representations, we further show that our ideas are amenable to novel, admissible search algorithms that allow us to quickly find high quality rules, even in very large datasets.

3.1 Background and Related Work

In a sequence of papers culminating in [72], Park and Chu investigate a rule finding mechanism for time series. However, the algorithm is only evaluated for speed and then only on random walk data. No evidence was presented that the algorithm could actually find meaningful rules in time series.

Like Park and Chu, Wu and colleagues also use a piecewise linear representation to support rule discovery in time series. They tested their algorithm on real (financial) data, reporting approximately 68% "*correctness of trend prediction*" [82]. However, the authors graciously tested their algorithm on data provided by others, and when they ran their algorithms on random walk data they again achieved approximately 68% correctness of trend prediction [83]. This strongly suggests their original results did not differ from random guessing.

The most referenced time series rule-finding method in the literature is [57], which quantizes the data with K-means clustering of the entire training dataset, and then hands the (now) symbolic data over to a classic association rule discovery algorithm. The success of a rule is measured with a score called the J-measure. The method was used in more than a dozen papers before it was shown that the J-measure gave the same significance to rules found in completely random data, as to rules found in real data [62].

3.2 The Intuition Behind Rule Discovery

It may be instructive to first consider the analogue problem of rule discovery in symbolic strings. Let us consider the familiar poem, "The Raven", by Edgar Allan Poe. It begins:

Once upon a midnight dreary, while I pondered weak and weary...

What are the possible rules we might discover in this text? One possible rule that occurs to someone familiar with the poem is that the word "*door*" often follows the word "*chamber,*" a rule we can denote as:

chamber → door

We refer to the left side of the rule as the *antecedent* and the right side as the *consequent*. This rule is based on our observation that we see the phrase "*chamber door*" ten times in the text. We note that this is not a perfect rule; the word "*chamber*" appears once without been followed by "*door*" ("*Back into the chamber turning...*"). Furthermore, it is important to note that the rule does not make the claim that all, or even many, occurrences of "*door*" are preceded by "*chamber*". In fact, there are four additional examples of the word "*door*" in the text.

The differences between rule discovery in text and the task at hand are telling; with time series data we do not have unambiguous segmentation of the stream, thus we are facing data that is more like this:

onceuponamidnightdrearywhileIponderedweakandweary....

Given such a text, there are (language agnostic) algorithms that can segment the string into the original words [56], with varying degrees of accuracy. However, segmenting a *real-valued* time series into meaningful episodes is *much* more difficult. Furthermore, the problem is further complicated by the fact that, in most cases, the time series does not consist solely of discretely concatenated events. Rather, the events may be interspersed with meaningless filler symbols. For example, if we examine a motion capture of a sign language version of this poem there will be locations that do not correspond to discrete signs, but rather to transitions between signs. This will produce something rather like this:

oncexauponwamidnightmtdrearydwhileulpponderediweakoandajweary...

Finally, time series are inherently real-valued and as such, tests for equality are meaningless. This would be equivalent to our text string having some misspellings, like this:

qncexauponwamidmightmtdreerydwgileulpponderediweekoandajweauy...

The problem is now significantly more difficult than the original statement. We must generalize the antecedent to allow flexibility, perhaps by triggering the occurrence of a pattern that is within a certain threshold *t* distance under some suitable distance measure:

$$dist(\text{“}\textbf{\textit{chamber}}\text{”}, \text{substring}) \leq t \rightarrow \textbf{\textit{door}}$$

However, we are not done generalizing the rule model. The existence of misspellings in our data means that we may wish to accept similar consequents such as *poor* or *dooor* as successful predictions. Furthermore, we originally assumed that the consequent immediately followed the antecedent. However there may be some additional symbols between words. Thus we need to define a parameter, *maxlag*, which is the maximum number of characters between the end of the antecedent and the beginning of the consequent. For example: if *maxlag* is set to two, then any of the below would be considered successful predictions:

*...**chamberdoor**..., ...**chamber**z**door**..., ...**chamber**xy**door**...*

but the following:

*...**chamber**xzuv**door**...*

is not a successful prediction because the lag between the antecedent and consequent is too long.

There are two reasons for having a *maxlag* parameter. The first is to allow for meaningful, falsifiable predictions. The prediction that "*this consequent will eventually occur*" is paradoxically both unfalsifiable and almost certainly true (if we wait long enough). The second reason is more pragmatic, a bounded value on the *maxlag* will make the search over the rule space more tractable. We can now show our final rule format:

$$dist(\textbf{chamber}, substr_{i,j}) \leq t_1 \rightarrow dist(\textbf{door}, substr_{m,n}) \leq t_2,\ m-(i+j-1) \leq maxlag.$$

This can be read as follows: "If we see a substring that is within distance t_1 of the word *chamber*, then we fire the rule and expect to see a similar substring to word *door*, within a learned distance t_2 in the next *maxlag* time steps." In the next section we generalize these ideas to real-valued time series.

1) Moving to Real-Valued Data.

We are now ready to begin to "port" our ideas to the true real-valued time series that are of interest. We will start with an example for which we know the ground truth and for which the reader has already developed some intuition. It is important to note that we are not using external knowledge to help our algorithm, only to validate and explain it. We took an audio recording of the first four verses of "The Raven", and converted it to Mel-Frequency Cepstrum Coefficients (MFCC) space, keeping just the first coefficient; Figure 17 shows the data.

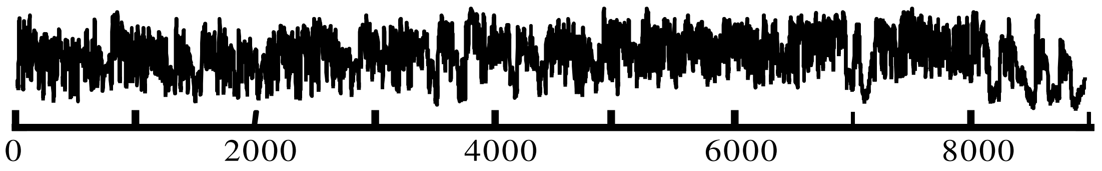

Figure 17: The first four verses of "The Raven", converted into 100Hz MFCC space. Just the first coefficient is kept.

As one might expect, it is difficult to make sense of such data. Using just the first 2,000 data points, which corresponds to the first verse of the poem, we found the pair of non-overlapping subsequences of length 100 (one second length in the original data) that had the minimum distance to each other. Such a pair of subsequences is referred to as a *time series motif* in the literature [69][70]. Figure 18 shows the motif pair.

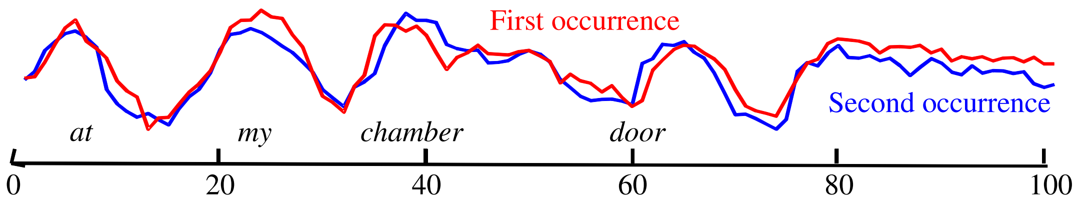

Figure 18: The motif pair discovered in the first 2,000 data points (20 seconds) of "The Raven". The shape corresponds to the utterance "...at my chamber door".

The occurrence of such a highly conserved motif suggests *one* possible method for specifying rules. We could simply split the motif pattern in two, let the average of the left side be the antecedent, and then let the average of right side be the consequent. We then need to set the *maxlag* and the threshold t_1 parameters. For the momen62t, let us simply set the former to zero and the later to the mean distance between the antecedent motif prefixes plus one standard deviation. Figure 19 shows the rule.

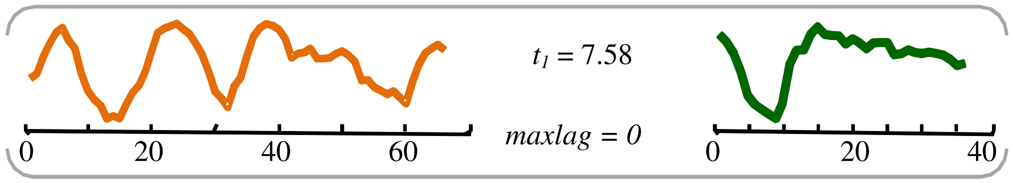

Figure 19: A rule learned from the first 2,000 data points of the data shown in Figure 17. If the antecedent pattern (left) is matched to a subsequence in a stream that is within Euclidean distance of 7.58 to it, we predict the immediate occurrence of the consequent pattern (right).

We can immediately test this rule by running it on the remainder of the data shown in Figure 17. The rule fires exactly three times; in Figure 20 we show the first rule invocation.

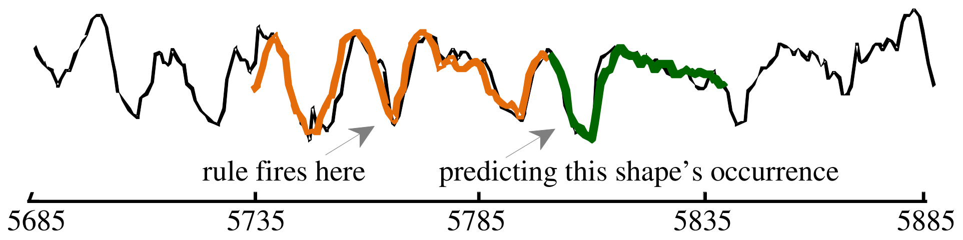

Figure 20: The rule shown in Figure 19 is first invoked at location 5,735. The resulting prediction appears accurate.

In all three cases, not only is the rule's prediction *visually* accurate, as shown in Figure 20, but if we check the location in the original audio we find that in every case it maps to an utterance of "***door.***"

In this simple example, hard-coding the *maxlag* to zero is intuitive; however, we can easily imagine examples that need the flexibility of a larger *maxlag* constraint. Consider Figure 21 which shows some accelerometer data collected from a device worn by a student at USC as he went about daily activities [73] .

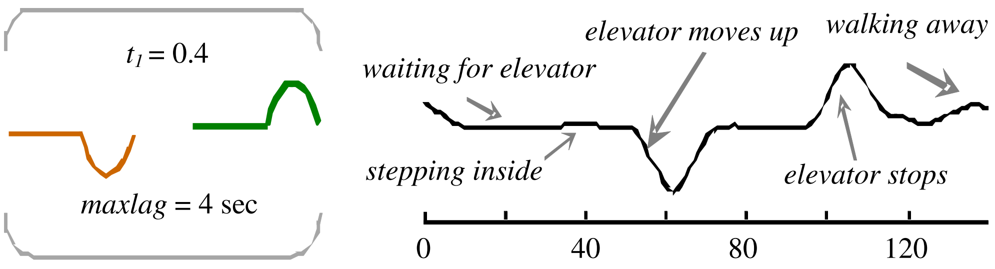

Figure 21: left) A rule for a z-axis accelerometer dataset encodes the fact that the initial acceleration "bump" of going up in an elevator must be eventually be matched by the elevator stopping at a floor. right) A subset of the data from which this rule was learned [73].

This example shows a very easy rule to spot. The semicircular bump created by an elevator accelerating must eventually be matched by a bump in the opposite direction when the elevator brakes at a floor (the rule for elevators going *down* is essentially the same, but with the consequent and antecedent swapped). The time lag between these two events depends on the number of floors serviced by the elevator.

3.3 The Rule Framework

We are now in a position to present the formal definitions necessary to rigorously define our rule framework. We begin with the definition of the data type of interest:

DEFINITION 1. A Time Series is a sequence $T=(t_1,t_2,\ldots,t_n)$ which is an ordered set of n real valued numbers.

Our rule discovery framework examines short sections of the time series, which are called *subsequences*:

DEFINITION 2. A subsequence of length n of a time series $T = (t_1,t_2,\ldots,t_m)$ is a shorter time series $T_{i,n} = (t_i,t_{i+1},\ldots,t_{i+n-1})$ for $1 \leq i \leq m - n + 1$.

When monitoring a time series we continuously extract the subsequence of the last n numbers, a *sliding window*:

DEFINITION 3. A sliding window (W), of length n, is the most recent n values of T.

Recall that we are only interested in the *shape* of the subsequences, not their amplitude or offset. We therefore z-normalize all subsequences [58][62]. The time taken to z-normalize a subsequence is linear in its length; however, we can z-normalize the sliding windows in amortized constant time by incrementally maintaining statistics [77].

We need to define a distance measure between two subsequences. While there are dozens of measures in the literature, recent empirical evidence suggests that Euclidean distance is very difficult to beat [58]. Furthermore, Euclidean distance is parameter-free, fast to compute, and is amiable to various data mining “tricks” such as indexing and early abandoning computation [70].

We empirically considered other distance measures for rule finding, including Dynamic Time Warping and Longest Common Subsequence based measures including Swale, Spade and EPR [58]. However none improved the accuracy of the rules (a finding consistent with [58]) and all required at least two orders of magnitude more time.

The Euclidean distance *D*, between two subsequences *A* and *B* of the same length *n* is given by:

$$D(A,B) = \sqrt[2]{\sum_{i=1}^{n} (A_i - B_i)^2}.$$

A time series *antecedent* is a subsequence used to trigger a rule if it is similar to the current sliding window:

DEFINITION 4. Assume we are monitoring a time series stream by continuously extracting the sliding window. Given a positive constant *t* (the threshold), and an antecedent time series R_a, a binary flag `fired` is set to **TRUE** if $D(R_a, W) < t$.

Note that in order for a candidate antecedent to be even considered as a rule precursor, it must occur at least twice; we obviously cannot generalize from single exemplars. This is essentially the definition of a *time series motif* [70]. In Section 71, we will exploit this fact in order to reduce our search space of antecedents (and later, *consequents*).

In principle, the threshold, *maxlag*, and antecedent could be hand chosen by a domain expert. However, as we will see below, it is possible to automatically find them.

Note that the shortest possible antecedent is of a length of three, because we are assuming we will z-normalize it, and there are only two trivial z-normalized time series of a length of two. In practice, the antecedent is likely to be significantly longer than this in order to reduce the possibly of spurious rules.

As an antecedent is a precursor to an event, a predicted subsequence shape which follows a triggered antecedent within a specified time (the *maxlag*) is called the antecedent's *consequent*:

DEFINITION 5. A consequent R_c is a time series subsequence that is predicted to follow the detection of an antecedent within a *maxlag* number of time steps.

The *maxlag* parameter encodes the fact that for a time series subsequence to be a meaningful consequent in a rule, it must occur within some acceptable time after the rule's antecedent has been detected. Without such a constraint on time, a consequent's occurrence may be coincidental.

DEFINITION 6. The *maxlag* is the maximum number of time steps allowed between a detected antecedent and its consequent. In particular, if t_k is the last value in W the moment the rule is triggered, then the consequent R_c, must be derived from a subsequence of T, $T_{i,n}$, such that $0 \leq i - k - 1 \leq maxlag$.

Given an *antecedent*, its *consequent*, the maximum expected *maxlag* delay between the two, and the *threshold* used to trigger a subsequence match, we have all the necessary components to specify a single *time series rule*:

DEFINITION 7. A time series rule R is a 4-tuple of $\{R_a, R_c, maxlag, t\}$.

To help with our rule search algorithm presented in Section 3.5, we explicitly define the consequent search space:

DEFINITION 8. The consequent search space (CSS), for a consequent $\mathrm{R_c}$ of an assumed length n, is the subsequence following an antecedent where a rule predicts the presence of a subsequence match for $\mathrm{R_c}$. Specifically, $\mathrm{T_{css}} = \mathrm{T}_{k+1,maxlag+n}$, where k is the final index of the detected antecedent.

Having formally defined time series rules and all supporting notation, we have just two more tasks to address before we are ready to test our ideas. We need to formalize a scoring function to tell us how good a candidate rule is. This is a *critical* step if we are to find meaningful rules [62]. Furthermore, since the space of all possible rules is infinite, we need to find an efficient search strategy. Our proposed scoring function is based on the MDL cost, which is only defined for discrete data. In the next section, we show that real-valued time series can be discretized with essentially no loss of information, thus allowing the application of MDL. Then in Section 3.5, we introduce our time series rule finding algorithm.

3.4 Data Discretization

Because of our intention to use MDL to measure the relative merits of candidate time series rules, we must transform our real-valued time series into a discretized space [65][74]. After careful empirical consideration of the many quantization options, we quantize the time series' real values into uniformly distributed bins. For each test data set, we utilize a sliding window of length n and z-normalize all possible subsequences while recording the minimum and maximum values. After attaining the global minimum value, *min*, and global maximum value, *max,* we then specify bin boundaries that are evenly distributed between *min* and *max*. The resulting bin width is then:

$$\frac{max - min}{desired\ cardinality}$$

For example, consider the five time point series subsequence shown in TABLE IV. Note that the five distinct, real values in the original data map on to fewer distinct values in the low cardinality representation.

TABLE IV. *LEFT*) A 5 DIMENSIONAL TIME SERIES *T*; NOTE THAT IT IS Z-NORMALIZED. *RIGHT*) 6 BIT AND 64 CARDINALITY REPRESENTATION OF *T*.

Original Data	***min = -0.915837463044344 max = 1.059696557703197***	***Low Cardinality Representation***
1.059696557703197		63
1.058743239030050		63
-0.289671788196926		20
-0.914384004268160		0
-0.915837463044344		0

As we can see in Figure 22, even if we reduce a 64-bit time series to a mere 6-bit representation, there is no *visual* difference. Beyond this visual and intuitive demonstration, we can show the (lack of) effect of discretization on time series with classification experiments, since the rule triggering step is essentially a classification problem. We conducted empirical tests on data from the UCR Time Series Archive [79].

For each dataset, we ran leave-one-out one-nearest-neighbor classification tests using uniform quantization with varying cardinalities. TABLE V provides a snapshot of the results. It is demonstrated that a real-valued time series can be drastically reduced with our discretization without significantly affecting the intrinsic information available. There is little loss incurred from a large reduction of the data's original real-valued space. In fact, because cardinality reduction of the original data can reduce the effects of noise and outliers, we can sometimes see some tiny (but not statistically significant) improvements in accuracy.

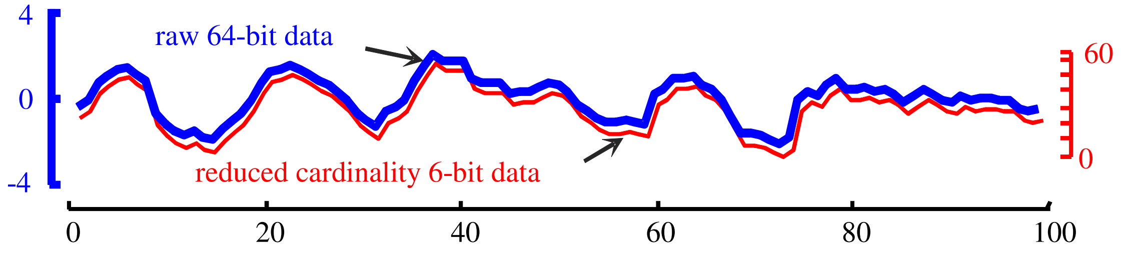

Figure 22: A snippet of the MFCC version of "The Raven," shown in original 64-bit representation (bold/blue) and in a 6-bit reduced cardinality (fine/red). The two versions were slightly shifted in the y-axis for clarity.

These two observations in Figure 22 and TABLE V allow us to use MDL with little fear that we are throwing away valuable information.

TABLE V. ONE-NEAREST-NEIGHBOR LEAVE-ONE-OUT ACCURACY RESULTS ON UCR DATASETS FOR VARIOUS CARDINALITIES.

Dataset	*64-bit (raw) Cardinality is 2^{64}*	*16-bit Cardinality is 65536*	*6-bit Cardinality is 64*
50words	63.1%	63.1%	63.3%
CBF	85.2%	85.2%	85.2%
Beef	66.7%	66.7%	66.7%
ECG	88.0%	88.0%	88.0%
FaceAll	71.4%	69.6%	69.6%
FaceFour	78.4%	78.4%	76.2%
Fish	78.3%	78.3%	77.7%
Lightning2	75.4%	75.4%	77.1%
Lightning7	57.5%	57.5%	58.9%
OSULeaf	52.1%	52.1%	52.1%

Which value of cardinality should we use? Empirically, if the value is anywhere in the range of 65,536 to 64, it makes no difference; we therefore use the smaller cardinality of 64 in all experiments.

3.5 Rule Discovery Algorithm

We are finally in a position to introduce our rule finding algorithm. In essence, it has two parts -- a scoring function and a search method which repeatedly invokes this scoring function, searching for high quality rules.

Our MDL scoring function is given three inputs:

- A training time series dataset.

- A candidate antecedent.
- An expected *maxlag* value (optional, defaults to *infinite*).

The function then returns four things:

- A suggested value for threshold *t*.
- A consequent.
- A learned *maxlag*.
- A score which reflects the quality of the resulting rule.

The score is a measure of how much bits are saved if we could compress the data using the rule, substituting real data with our prediction. As the scoring function is at the heart of our ideas, we will detail the intuition behind it next and formalize it in Section 2).

1) Intuition behind Rule Scoring with MDL Cost.

The intuition behind our scoring function is that if we make a good prediction, the consequent shape we predict will be similar to a subsequence that occurs within *maxlag* steps. We could quantify this similarity with Euclidean distance (which is essentially the *mean squared prediction error* used in forecasting) [68]. However, there is a significant shortcoming of the Euclidean distance for this task; it does not allow us to compare the quality of consequents with different lengths.

To make this clearer, let us return to our expository *text* example. Suppose we have to evaluate the following candidate rule:

$$dist(\text{“}\textbf{chamber}\text{”}, \text{substring}) \leq t \rightarrow \textbf{door},$$

Which when fired makes a prediction of length four. When encountering this string:

*...bustabovehis**chamber****door**withsuchnameasnevermore…*

It achieves a hamming distance (a good analogue of Euclidean distance) of 0. Contrast this result with the following rule: *dist*(“***chamber***”, substring) $\leq t \rightarrow$ ***doorwithlikename***, which when fired makes a prediction of length sixteen. When encountering the same string:

*...bustabovehis**chamber****doorwith~~such~~name**asnevermore…*

It achieves a hamming distance of four. Which of these is better? The former is an *exact but short* prediction; the latter is an *approximate but longer* (and arguably more informative) prediction. Unfortunately, simply normalizing for length does not work here. While it is not commonly understood, the Euclidean distance between two subsequences of length n can actually *decrease* when we expand to length n+1 because of the (re)normalization of the data using a larger denominator. So not only is the effect of length not linear, it is not even monotonic.

The solution to this problem, and the reason for the earlier digression into discretization of time series, is MDL [1]. In fact, for several decades MDL has been used to solve very similar problems in intrinsically discrete domains such as text, DNA, MIDI, etc. [59][66][78]. However, this application to time series rule finding is novel.

The intuition behind our use of MDL is to consider a candidate subsequence as a *hypothesis H* about a future event. This hypothesis has some cost: the number of bits it takes to store it. We denote this cost as *DL*, or the Description Length. Since we are storing the subsequences as plain text, we have $DL(H)$ = length(H) × $\log_2$(cardinality). For example, in Figure 23 we show a version of the consequent previously shown in Figure 19 (reduced to a cardinality of just eight for the sake of visual clarity). This time series is of length 36, and has a 3-bit (8 value) cardinality, so its cost is 36 × 3 = 108 bits.

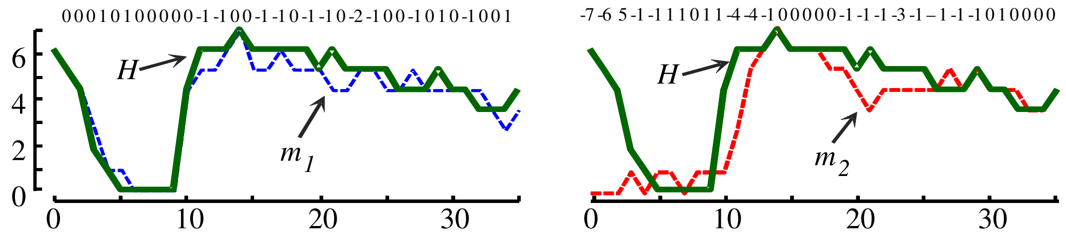

Figure 23: A candidate consequent H (bold) can be considered a reference model and used to encode other time series (fine/dashed) using their delta vectors (top).

We want to evaluate the quality of a candidate consequent by asking how well the prediction matched the future. We do this by asking, "Given our consequent, what is the *cost* to encode the error of the predicted match *m*?" We denote this as $DL(m \mid H)$, that is, the description length of a matching subsequence *m*, *given* our hypothesized consequent *H*.

We can measure this encoding cost by simply subtracting the consequent from the matching time series and encoding the difference vector efficiently. For the candidate match m_1 shown in Figure 23.*left*, the delta vector consists only of four unique small values (-2, -1, 0, 1), and its encoding cost using a Huffman encoding, with a zero-mean Gaussian distribution, is 85 bits. In contrast, in the candidate match m_2 shown in Figure 23.*right* the delta vector has eight larger unique values and its encoding cost is 128 bits.

This indicates that it is *not* as good a match to the hypothesized consequent as m_1. Thus the score of a candidate subsequence, *m*, with a given match, *H*, is as follows:

$$DL(m,H) = DL(m) - DL(m|H)$$

This equation allows us to measure the relative predictive power of subsequences, *independent of their length*.

Note that in order to find rules in a training set, we must have at least two firings. This means that to evaluate the hypotheses we must see how well it encoded a set, *M*, of at least two matches.

$$Score(M,H) = -DL(H) + \sum_{m \in M} DL(m,H)$$

where the set *M* consists of all subsequences to be compressed with the consequent *H*. For concreteness, suppose that the two examples in Figure 23 are both discovered when using a particular antecedent *H*, thus $M = \{m_1, m2\}$. The overall score of the *candidate consequent* using 0 is then:

$$Score(M,H) = -DL(H) + [DL(m_1) - DL(m_1|H)] + [DL(m_2) - DL(m_2|H)]$$

This can be read as, "*The **score** for matches, M, using H is the **cost** of H plus the bits saved from compressing all matches.*" With this model we can now directly measure the quality of a *consequent*. Similarly, we can use equation 0 to judge the quality of a rule's *antecedent* by its ability to compress its own matched subsequences. Summing the antecedent and consequent score of a rule yields an overall score for that rule. In the next section we expound how we can use this primitive to measure the quality of a potential *rule*.

2) Formalizing Rule Scoring with MDL.

The code presented in this section is optimized for simplicity of explanation; some redundancy exists in exchange for clarity. Carefully commented code is freely available at [85]. We now present both detailed pseudo-code and cross-annotated figures to illustrate our rule scoring algorithm. The rule scoring algorithm consists of the following steps:

1) Find non-overlapping subsequences, A, of T within t of R_a.

2) Find motifs which exist after the discovered subsequences.
3) Select the best scoring motif as a consequent subsequence.

Provided a candidate antecedent, threshold t, and a *maxlag*, our rule scoring method, outlined in Algorithm 1, returns the highest quality *consequent* subsequence as measured using equation 0. In line 1, the *CSS* are extracted from the time series for the given antecedent/threshold. For example, in Figure 24 two subsequences $T_{61,66}$ and $T_{398,66}$ are within t distance of R_a.

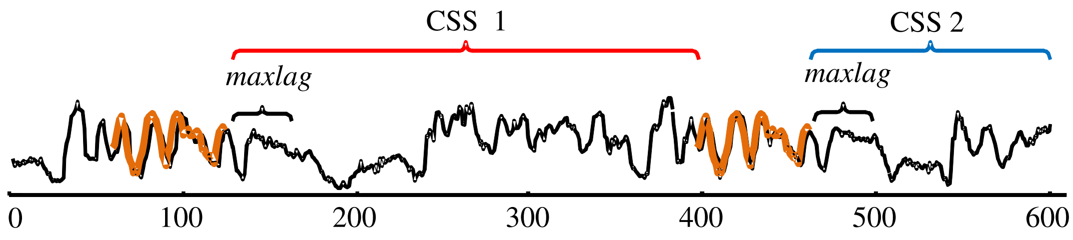

Figure 24: Antecedent matches (bold) are found at $\boldsymbol{T_{61,66}}$ and $\boldsymbol{T_{398,66}}$. Their corresponding consequents must then begin within maxlag time and must end within their consequent search subsequence.

Our candidate rule predicts that a consequent subsequence should exist after each of the antecedent matches. The consequents corresponding to each match must *begin* within the range [127,127+*maxlag*-1] and [464,464+*maxlag*-1], respectively.

```
Algorithm 1: {bsfScore, consequent, maxlag} =
  ScoreAntecedent(timeSeries, antecedent, t, expMaxlag)

{cssTS,bounds} = GetCSSTimeSeries(timeSeries, antecedent, t);
maxConsequentLength = min(MAX_CONEQUENT_LENGTH,
                          differences(bounds));
bsfScore = -∞;
for i = 3 to maxConsequentLength
  motifs = MKmotifs(cssTS, i, bounds, expMaxlag);
  for j = 1 : length(motifs)
    consequent = CreateCandidate(motifs[j]);
    score = GetCost(consequent);
    lag = -∞;
    for k = 1 to |bounds|
      lowerB = bounds[k-1] + 1;
      upperB = bounds[k];
      [index lag cssScore] =
        MinDistSubseq(cssTS[lowerB to upperB], consequent);
      score += cssScore;
      lag = max(index,lag);
```

```
16      if (score > bsfScore)
17        bsfScore = score;
18        bsfConsequent = consequent;
19        maxLag = lag;
```

We assume that antecedents and consequents cannot overlap; the maximum consequent length is then upper bounded by either a user-given expected maximum length, the presence of an antecedent match, or the end of the time series. These bounds define the *consequent search subsequence* (*CSS*) for each antecedent match. Although these bounds do reduce the size of the search space, we still must conduct pairwise comparisons of all subsequences between all CSS. This may prove costly if no *maxlag* is provided for the consequent and if the antecedent matches are very distant from each other. To mitigate this, we note that any high-quality consequent, just like an antecedent, must also be an approximately repeated pattern (a time series motif [70]); thus we search for consequent motifs in line 5. We can therefore reduce the search space from all subsequences between CSS to only motifs with at least one occurrence in two CSS. As shown in Figure 25, we extract each CSS and then conduct a motif search which respects the *maxlag* and the original CSS boundaries by ignoring subsequences which cross them (line 5).

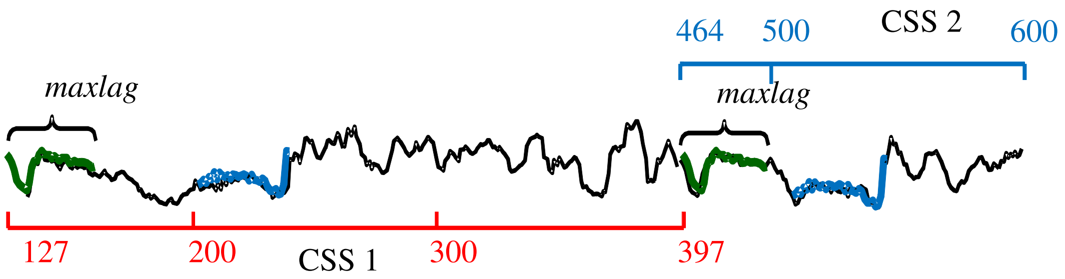

Figure 25: Concatenated CSS time series from Figure 24. The overlaid green and blue subsequences are pairs of candidate consequent motifs. Because the blue consequent does not begin within both maxlag time frames, it cannot be considered for this rule.

In this example, two motif pairs (overlaid by green and blue) exist. Because the subsequences in the blue motif pair do not begin within the *maxlag* time frames, it cannot be considered for the rule's consequent (and wouldn't be detected by the specialized motif search). In order to score the viable motif pairs (such as the green pair in Figure 25), a *candidate consequent* is created by averaging the motif pair (line 7).

Next, in each *CSS* the closest matching subsequence to the candidate is found and its score contribution is calculated (line 10). We then compute equation 0 to obtain the candidate consequent's score. Assuming that the green consequent was the best scoring motif, Figure 26 shows the final rule's antecedent and discovered consequent.

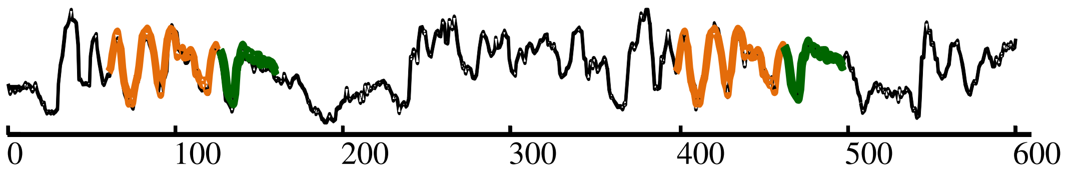

Figure 26: Provided an antecedent (orange) and threshold, which allows two antecedent matches in our training time series, we discover the consequent (green) within the maxlag from all antecedent matches.

Now that we have defined a method of scoring a candidate rule triplet antecedent/threshold/expected *maxlag*, we are in a position to describe a method for searching for likely antecedents to score.

3) Rule Searching

It is clear that the search space of possible antecedents is extremely large; how can we make the search tractable? The most important observation we leverage off is that for any rule its antecedent must be an approximately *repeated* subsequence within *T* (i.e. a *time series motif* [70]). We can perform a simple experiment to give an intuition as to how we may be able to exploit this fact. We took every subsequence of length 100 (one second) of the MFCC version of "The Raven" and recorded its distance to its nearest neighbor. The distribution of these distances is shown in Figure 27 with a few annotated examples. Note that one occurrence of the phrase "*...chamber door...*" has a very small distance to its nearest neighbor, which is naturally just another occurrence of the phrase. Similarly, repeated phrases such as "*...the raven....*", "*…on the floor…*", etc., also have small distances to their nearest neighbors.

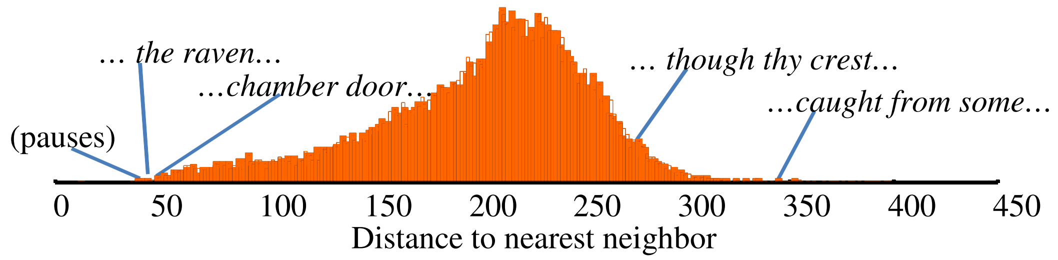

Figure 27: Distribution of distance to nearest neighbor for one second snippets of the audio (in MFCC space) of "The Raven."

In contrast, phrases featuring hapax legomena[1] such as "*caught*" or "*crest*" have a huge distance to their nearest neighbor. Moreover, consider words which are not hapax legomenon, such as "*soul*" (which appears seven times). Because we are considering one-second clips, the word "*soul*" is preceded and followed by other words to make a short phrase, and these *phrases* are typically unique. If we are attempting to find rules in the text space, unique words or phrases do not need to be considered since we clearly cannot generalize from a single example. Moreover, Zipf's law tells us that about half the words in an English text are hapax legomena [64], and an even larger proportion of *phrases* must be unique. This observation is for text; however, many authors have noted this property holds for all kinds of multimedia data [55] and, as Figure 28 hints, it is true for most time series.

This observation suggests that we only need to evaluate candidate antecedents that map to the far left of the distribution. Even if we had to actually build the entire distribution, this would still be a useful result. However, by exploiting recent results in time series motif discovery [70], we can find the left most objects in the distribution very efficiently, in just O($n$log(n)) time.

Using these observations we reduce the space of candidate antecedents by limiting our search to discovered motifs, just as we did for consequents. The rule finding algorithm shown in Algorithm 2 searches a time series for the highest scoring rule. The first step in rule finding of time series is a motif search for possible antecedents of varying lengths

[1] A *hapax legomena* is a word that appears only once in a body of text.

(line 4). Again, we use the MK algorithm which is the state-of-the-art motif discovery algorithm [70]. The test on line 6 removes motifs that cannot be antecedents (e.g. a motif pair is so close together that there is no room for a consequent between them). On line 8, we construct a candidate antecedent from the detected motif pair by averaging their values. Referring back to our example time series, Figure 28 displays a few candidate antecedents.

While the threshold is a real-valued number with an infinite range of values, we only need to check a small number of different values. In particular, we need to check the value that is *just* large enough to make the rule fire twice, then that is *just* enough to make the rule fire three times, etc. The number of checks is limited by the fact that we rapidly run out of data as the rule fires many times. This process is called at line 8.

```
Algorithm 2: rule = FindRule(timeSeries, expMaxlag)

1  maxAntLength = min(|timeSeries| / 2, MAX_ANTECEDENT_LENGTH);
2  bestRule.Score = -∞;
3  for i = 3 to maxAntecedentLength
4    motifs = MKmotifs(timeSeries, i);
5    for j = 1 to length(motifs)
6      if (!IsMotifValid(motifs[j]))
7        continue;
8      {antecedent, thresholds} =
         CreateCandidate(motifs[j], timeSeries);
```

```
    for k = 2 to length(thresholds)
      {score, consequent, maxLag} =
        ScoreAntecedent(timeSeries, antecedent,
                        thresholds(k), expMaxlag);
      if (bestRule.Score < score)
        bestRule =
          NewRule(score, antecedent, thresholds(k),
                  consequent, maxLag);
```

In Figure 29, we show the sorted locations of all non-overlapping matches to our familiar example antecedent. Notice that using this threshold selection technique, we have pruned trivial matches and all subsequences that are too short to contain an antecedent.

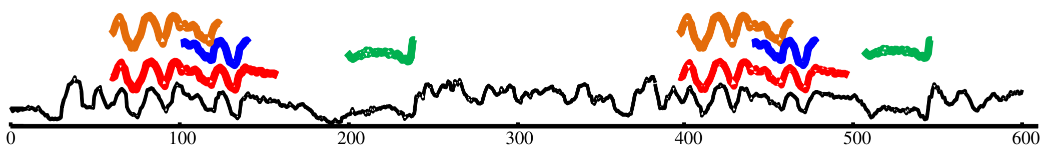

Figure 28: Motif locations extracted from a training time series. Colored/bolded subsequences are antecedent candidates calculated by averaging motif pairs and are shown over the original motif locations.

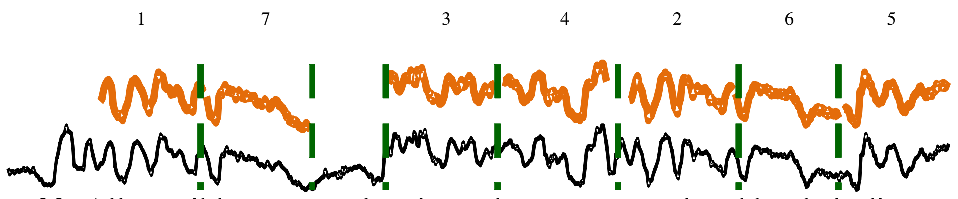

Figure 29: All possible non-overlapping subsequences ordered by their distance to a candidate antecedent. Subsequences ranked at four and higher will be pruned due to the small consequent length they impose.

In our example it can be seen that there are only seven possible thresholds to test with our given antecedent. We can also see that the subsequences at rank four and higher are not viable since the consequent length of a particular antecedent/threshold pair is upper bounded by the smallest available *CSS* length.

Thus these thresholds can be quickly dismissed without expensive calculations. Furthermore, if the user has provided a minimum consequent length, we may be able to prune more thresholds (or the entire antecedent) if the minimum *CSS* length between the remaining matches are below the allowed length. Using the rule scoring method discussed in the previous subsection, we can then score and rank each viable antecedent/threshold pair.

4) Pruning and Early Abandoning.

Our rule finding framework was defined to both produce meaningful results *and* to allow exploitable areas for search space pruning and early abandoning. As described, the first step of rule finding is to search for antecedent motifs of all possible lengths. For each discovered antecedent motif we then consider each acceptable threshold. For any antecedent motif, we can use equation 0 to calculate its contribution, $\boldsymbol{S_{ant}}$, to the rule score. To achieve this, we must consider all possible thresholds for each antecedent. In Figure 29 we have shown that the threshold space is limited. We can further prune thresholds by calculating their optimal score contribution and then abandoning when the upper bounding score drops below a *best so far*. Using equation 0, the upper bounding best score, $\boldsymbol{S_{best}}$, of a consequent candidate of length $\boldsymbol{l}$, $\boldsymbol{c_l}$, within any one *CSS* is:

$$S_{best} = DL(c_l) - l \times \mathrm{cost}_{\mathrm{min}}$$

where $\text{cost}_{\min}$ is the minimum bits needed to store a differenced value. By defining the description length of any subsequence as simply:

$$DL(c_l) = l \times \log_2(cardinality)□$$

we are unbiased to the shape of the subsequence. Using this definition with equation 0, we can measure the description cost of a consequent before actually conducting a motif search for it.

Using equation 0 with equation 0, we can compute an upper bound rule score, S_{max}, of a rule with only a given consequent length, antecedent, and a threshold which matches n subsequences:

(3.1) $$S_{max} = S_{ant} - DL(c_l) + S_{best} \times n□$$

Equation (3.1) calculates the highest possible rule score for an antecedent where a consequent motif of length l is found that matches perfectly to a subsequence in every *CSS*. If this score is less than the *best-so-far* score, we can forgo all consequent motif searches of length l as well as their necessary subsequence comparisons. If we cannot prune this {antecedent, threshold, consequent length} triplet, we must proceed with searching for motifs of length l which fit the current *CSS* constraints.

Once the consequent motif search is completed, we then must score each discovered candidate consequent. Again, equation (3.1) provides an upper bound for any consequent motif with a given antecedent and threshold. With a small modification we can iteratively update our upper bounding score as we search for the best *CSS* matches for a candidate consequent. As we find the best subsequence within each CSS, we will update the upper bound for the current consequent candidate and check to see if abandoning is possible. For *CSS* i of *n,* there exists a subsequence which has maximum score, $score_i$, using equation 0. If we have discovered the best scores from k amount of CSS, we then can calculate the new rule’s upper score bound as:

(3.2) $$S_k = S_{ant} - DL(c_l) + \sum_{i=1}^{k}[score_i] + S_{best} \times (n-k)$$

In this equation, the first half (up to and including the summation) are exact rule score calculations while the remaining quotient is the upper bounding contribution of the uncalculated CSS matches. Note that at $k = 0$, equation (3.2) equals the upper bound equation (3.1) and when $k = n$, equation (3.2) is the rule’s exact score.

We can further prune one level deeper in our search. As we search for the best subsequence match within a *CSS*, we must calculate the distances between every time point of a candidate consequent and candidate subsequence. Because the cost, c_i, supplied from our Huffman encoding for a distance, d_i, is a scalar function on Euclidean distance, we can conduct early abandoning techniques, similar to the Euclidean distance early abandoning conducted in [70], and prune distance calculations.

3.6 Experimental Evaluation

To ensure that our experiments are *easily* reproducible, we have built a website which contains all the data and commented code, together with raw spreadsheets for the results

[85]. In addition this website contains many additional experiments that are omitted here for brevity.

We provide two sources of evaluation of *quality*. In some cases, as in "The Raven" example, we show the rules are meaningful by considering annotation available by external labels of some kind. In the more general case, we use the Euclidean distance between our predicted consequent and the *F* matching locations, a value we denote as F_{error} (this is essentially the root-mean-squared error). Because this number is difficult to interpret by itself, and motivated by results of previous methods on random data, we do the following: on the same testing set, using the same consequent, using the same *maxlag*, we fire the rule *randomly F* times and measure the Euclidean distance between our predicted consequent and the *F* random locations. We denote this value as R_{error} (which is averaged over 1,000 random runs). Our reported measure of quality then is just

$$Q =, \frac{F_{error}}{R_{error}}.$$

Q values close to one suggest our rules are no better than random guessing and values less than one indicate that we are finding true structure in the data. Each of the following datasets has been sampled at 100 Hz.

1) Finding Rules in Bird Songs.

Most animals communicate using hardwired, innately determined sounds. Humans and songbirds, in contrast, are among the few animals that learn their communication skills. As songbirds, in particular zebra finches (*Taeniopygia guttata*), are easy to study in the laboratory, they are often used as model organisms to investigate the neural bases of learning, memory, sensorimotor integration, and even models of cultural transmission [63][71][76].

For at least a decade, experts have been examining bird songs, usually by manually inspecting the visually intuitive spectrogram plots, looking for structure and grammar [76]. Could such grammars manifest as time series rules? To test this, we converted zebra finch songs donated by the authors of [76] into MFCC space (as with "The Raven", c.f. Figure 17) and searched for rules. We begin by considering an entire call small enough, four seconds, to be completely reproduced in Figure 30.

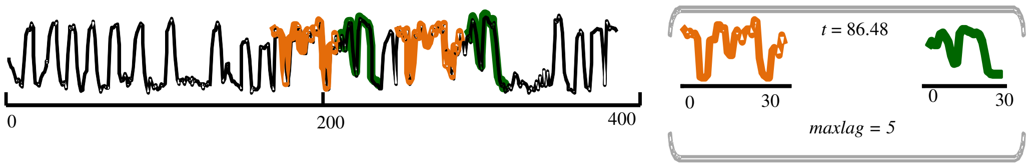

Figure 30: left) A short snippet of bird song in 1st coefficient MCFF space. right) The first ranked rule learned from it.

We then tested this top ranked rule on the full recording of the same finch; the results are shown in Figure 31.

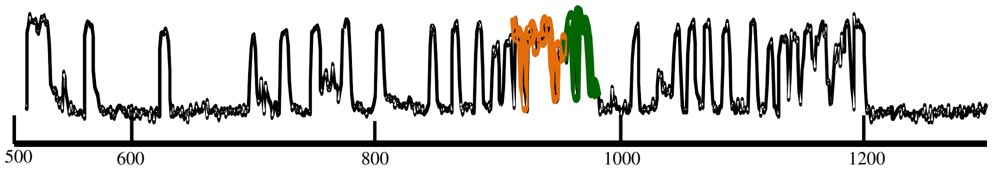

Figure 31: The rule shown in Figure 30.right was invoked on this section of song by the same bird.

The rule firing here is visually intuitive, and also appears correct if viewed as a spectrogram plot. The quality of this rule was $Q = 0.213$; however, that was based on a single firing. In Figure 32, we show the firing of the learned rule on a *different* bird's call which fires six times in its complete twelve-second period. Its quality was $Q = 0.237$, suggesting a very robust predictive rule that generalizes to other data.

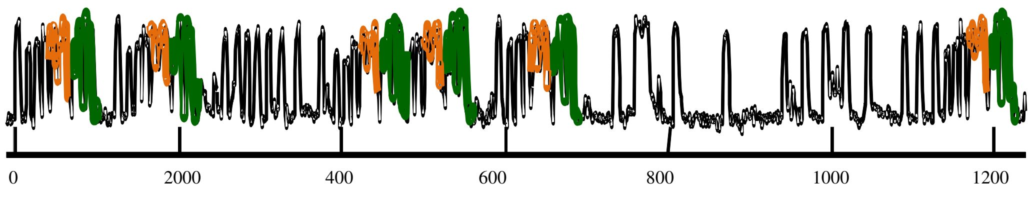

Figure 32: The rule learned in Figure 30 on an independent dataset was fired six times in this twelve second song.

Do rules change over time? Do rules learned from a parent generalize to their offspring? These questions are currently under investigation.

2) Finding Rules in NASA Telemetry Data.

The NASA valve data set consists of 36 events of interleaved erroneous and nominal solenoid voltage measurements recorded off of Marrotta series MPV-41 valves as they are tested in a laboratory [60]. We conducted rule finding on the entire time series. Figure 33.*right* shows the top ranked rule learned from the training segment in Figure 33.*left,* where the first peak is that of a failed solenoid.

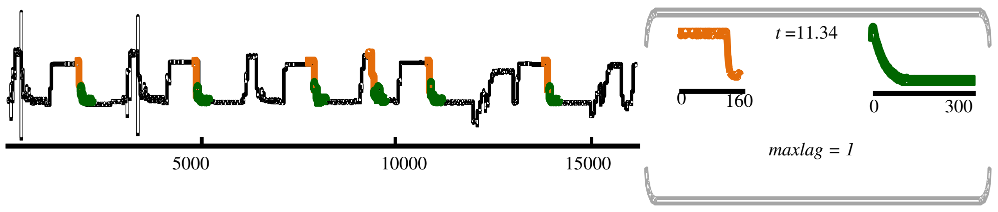

Figure 33: *left*) A short training segment of the NASA data. *right*) The first ranked rule learned which characterizes a nominal discharge.

This rule appears to describe a normal solenoid discharge event: a rapid decrease in the current is immediately followed by a slight ramp and gradual, complete discharge. Because of the variety of malfunction events, in contrast to the homogeneity of normal solenoid readings in this data set, this rule learned from successful tests achieves a higher MDL score as well as an exceptional $Q = 0.053$. Note that this rule is also learned from a single failure instance at the fourth marked peak. This may indicate that the valve assembly miss-cycled and that the solenoid itself still experienced a nominal discharge. Apart from this outlier, the entire set of 18 normal solenoid trials was detected with this rule.

3) Finding Rules in Human Speech.

We note that the rule in our running example on "The Raven" was actually a *learned* rule. A similar rule trained on the first verses is shown in Figure 34, and two invocations of it on independent test data (the remaining 17 verses) are shown in Figure 35.

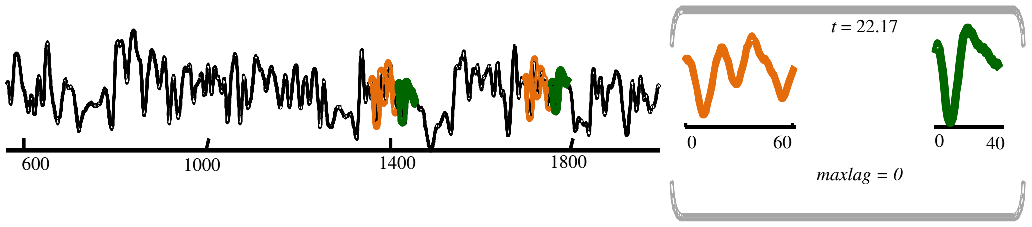

Figure 34: left) The first verse of "The Raven" converted into MFCC space with the learned rule shown. right) The learned rule which corresponds to "my chamber" and "door".

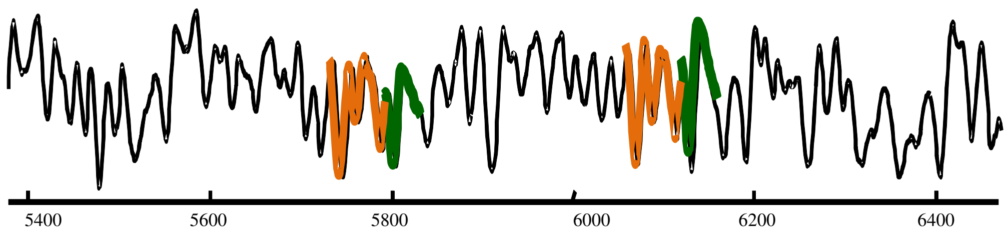

Figure 35: The rule learned in Figure 34 is applied to the rest of the data set. The displayed rule firings correspond to the utterances "my chamber door" in verse three.

This rule antecedent corresponds to "... *my chamber*..." This phrase does appear six more times in the text and our learned rule fires *twice* and in the correct places located in the third verse, achieving $Q = 0.199$. The firing of the rule only twice suggests our simple threshold value selection was too conservative. In fact, if we manually increase it by a few percent we *do* correctly detect all of the other occurrences.

Clearly learning the threshold from just two examples is difficult, tentative experiments suggest that we could use *transfer learning* to mitigate this issue [84], an idea we gloss over due to space limitations.

4) Randomized Rule Data Set.

We believe that the results in the previous section provide forceful and intuitive evidence of the utility of our ideas. However the number of case studies presented is necessarily limited by space considerations (*additional* examples are at [85]). To summarize our performance on thousands of labeled data sets, we must resort to generating synthetic time series into which we embed synthetic rules to be discovered.

To create such data sets, we construct a random walk time series of length l_{gen}, which is in the range of [75 : 250]. A split point is randomly chosen, and then the left partition is treated as our rule antecedent while the remaining section is the consequent. An example of a generated rule is shown in Figure 36.*right*. A *maxlag* is also randomly chosen from the range [0 : l_{gen} x 0.1]. We then create a second random walk time series, of a fixed length of 10,000, and with an added caveat; at every time point calculation the generated rule antecedent may be inserted (rather than a length of random walk points). If an antecedent is injected, there is a likely probability of 95% that the consequent will be inserted after at most the selected *maxlag*. Rule antecedents and consequents are appended to the current random walk with some added Gaussian noise, with amplitude equal to 5% of the standard deviation of the data.

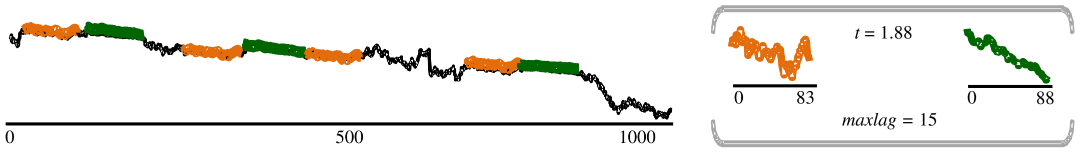

Figure 36: right) A randomly generated rule. left) A small example random walk with occurances of the rule. Note that the third antecedent injection does not have a corresponding consequent.

With our synthetic data we can perform our rule finding method and directly measure our detection capabilities with the classic measures of *precision* and *recall*. We define a detected region as *precisely* detected if the tentatively discovered rule and the ground truth region have at least 90% overlap each. A ground truth region can only be associated to one suggested region.

Because we can produce arbitrary amounts of such data, we averaged our findings over 1,000 randomly generated rules and with 1,000 injected random walk time series for each rule. On average, our method has a precision of 0.85 with a recall of 0.86. Note that unlike some of the real-world examples in the previous section, in most cases the synthetic rules are so subtle as to evade even careful human inspection.

Our recall rate reflects our conservative threshold setting. As previously demonstrated with "The Raven" data set, slightly increasing the threshold manually allows us to recover the additional rule regions. This suggests that the setting of the threshold warrants further research.

5) Runtime Analysis.

As our intent is to introduce a novel framework for *meaningful* rule discovery in time series, we have not optimized our initial implementation for *speed*. We begin our search over all possible antecedent lengths; this operation is linear with respect to the length of the test time series and each search costs $\boldsymbol{O(nlog(n))}$. For each motif, we then conduct a second motif search for candidate consequents in the *CSS* ranges; this is done for each possible threshold of a candidate antecedent, running at $\boldsymbol{O(nlog(n) \times \frac{n}{m})}$ where *m* is the length of the candidate antecedent. For each candidate consequent we find its closest subsequence in each CSS, but we extract these distances cached from motif search. This leads to a worse case runtime of $\boldsymbol{O(n^4\, log^2(n))}$.

As described in Section 3.54), on average our rule search space is greatly reduced by several things. First, our definition of antecedents and consequents as *motifs* immensely prunes the number of subsequences we must test. Extending to a one-time batch lookup of motifs for both antecedents and consequents could be performed, yielding a worse case runtime of $O(n^2 + n^2\, log(n))$, with some extra overhead for reconsidering subsequences previously excluded as trivial matches. Second, motif lengths of both antecedents and consequents can be upper bounded. As an antecedent must occur *at least* twice, its length can be no more than $\frac{n}{2} - 2c$, where c is an expected consequent length. Consequents must also fit within their CSS.

Furthermore, as demonstrated in Figure 29, the threshold possibilities for antecedents are finite and, because antecedent matches may appear spatially close, can be significantly pruned. These exploits occur without any user rule preferences. If we further exploit user constraints, such as "*The antecedent and consequent should be about the same length*" (suggested by ornithologists) or "*The maxlag should be close to zero*" (suggested by linguists for the poetry example), this knowledge further greatly improves the algorithm's runtime. Empirically, our pruned algorithm has demonstrated a considerable reduction (over 73.2% pruned at just the threshold search level) of the search space. We have performed additional analysis at [85].

Beyond our optimizations using early abandoning and pruning methods, we can improve the efficiency of the underlying critical sub-function of motif search. By applying a suite of abandoning and online calculations techniques to the problem of subsequence similarity, a core component of motif search, we can achieve speed up between 10x and 100x for our time consuming motif search [75].

Finally, we note that if a domain expert has spent several years to collect data, as is the case in both the bird song and NASA examples, they will probably not baulk at waiting a few minutes or even a few hours for an algorithm that can produce useful, actionable knowledge. Thus we feel that time complexity is not a major bottleneck to the adoption and development of our ideas.

3.7 Conclusions

We now finished introducing a parameter-lite technique for finding rules in time series. Our rule representation is expressive enough to allow rules with different length antecedents/consequents/lags, but at the same time does not require extensive human intervention or tweaking. We have shown our framework can find intuitive, high-quality rules in diverse domains and that it can scale to large datasets by using state-of-the-art motif discovery algorithms to drastically reduced the space of possible antecedents and consequents. We have made all code/data freely available to the community [85], so that others can confirm, extend, and ultimately use our ideas. Future work includes further addressing scalability issues and dealing with the concept drift encountered in streams.

4 CHAPTER 4: CONCLUSIONS AND FUTURE WORK ON PATTERN-BASED SIMILARITY

Throughout this work we have utilized the existence of patterns in data sets to enable for an extremely generalizable similarity measure for both images and audio and created a novel method for rule discovery in time series which operated on a motif base symbolization of the data. In our work, we have leveraged varying definitions of patterns. In images, we matched repeating textural patterns; for audio, we analyzed the repetitions of the auditory image spectrograms; in time series, we extracted motifs. These contributions indicate that for future classification problems on novel data, if we can provide a definition for a pattern in our data set then we can create similarity methods on the CK method, or discover relations between these patterns as we did for rule discovery in time series. As such, future works include an expansive analysis on the generality of CK based measures on further audio data sets and combining the analysis of image and time series data by measuring similarity in video. Another avenue of future research is the analysis of rules in heterogeneous data formats, such as the relation between the imagery and audio from a video, by symbolizing the data by their base patterns (as we did for rule discovery in time series).

References

[1] M. Khalid, *Design of an Intelligent Wood Species Recognition System*, Technical Report of Center for Artificial Intelligence and Robotics (CAIRO), Malaysia, 2008.

[2] R. Porter, N. Canagarajah, *Robust Rotation-Invariant Texture Classification: Wavelet, Gabor Filter and GMRF Based Schemes*, IEE Proc. - Vision, Image and Signal Processing, 144:180-188, 1997.

[3] K. Russell, H. Do, J. Huff, N. Platnick, *Introducing SPIDA-web: wavelets, neural networks and Internet accessibility in an image-based automated identification*

system, In N. MacLeod (ed), Automated Object Identification in Systematics: Theory, Approaches, and Applications. Springer Verlag, 2007.

[4] F. Bianconi, A. Fernandez, *Evaluation of the effects of Gabor filter parameters on texture classification*. Pattern Recognition 40(12), 3325–3335 (2007).

[5] P.R. Cohen, D. Jensen, *Overfitting explained*, In Prelim. Papers Sixth Intl. Workshop on Artificial Intelligence and Statistics, pages 115–122, January 1997.

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

[7] D. Cerra, M. Datcu, *A Model Conditioned Data Compression Based Similarity Measure*, DCC, 2008.

[8] M. Li, J. H. Badger, X. Chen, S. Kwong, P. Kearney, H. Zhang, *An information-based sequence distance and its application to whole mitochondrial genome phylogeny*, Bioinformatics 17: 149-154, 2001.

[9] M. Li, X. Chen, X. Li, B. Ma, P. Vitanyi, *The similarity metric*, Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, Pages: 863 – 872, 2003.

[10] J. Bethony, S. Brooker, M. Albonico, S. Geiger, A. Loukas, D. Diement, P. J. Hotez, *Soil-transmitted helminth infections: ascariasis, trichuriasis, and hookworm*. The Lancet, 2006.

[11] P. De Ley, Assistant Professor and Assistant Nematologist at UCR, Personal communication, Feb 2009.

[12] M. Garcia, A. Jemal, E. M. Ward, M. M. Center, Y. Hao, R. L. Siegel, M.J. Thun, *Global Cancer Facts & Figures 2007*, Atlanta, GA: American Cancer Society, 2007.

[13] A. Jemal, R. Siegel, E. Ward, et al, *Cancer Statistics*, CA Cancer J Clin., 56:106-130, 2006.

[14] J. Suckling, *The Mammographic Image Analysis Society Digital Mammogram Database*, Exerpta Medica, International Congress Series 1069, pp375-378, 1994.

[15] S.W. Duffy, L. Tabar, H. H. Chen, et al, The impact of organized mammography service screening on breast carcinoma mortality in seven Swedish counties, Cancer, 95(3):458-469, Aug 1 2002.

[16] L. L. Humphrey, M. Helfand, B.K. Chan, S.H. Woolf, *Breast cancer screening: a summary of the evidence for the U.S. Preventive Services Task Force*, Ann Intern Med., 137 (5 Part 1):347-360, Sep 3 2002.

[17] L. Tabar, M. F. Yen, B. Vitak, H. H. Chen, R. A. Smith, S. W. Duffy, Mammography service screening and mortality in breast cancer patients: 20-year follow-up before and after introduction of screening, Lancet, 1405-1410, Apr 26 2003.

[18] American Cancer Society, *Breast Cancer Facts & Figures 2007-2008*, American Cancer Society, Inc.

[19] M. Giger, *Current Issues in CAD Mammography. Digital Mammography*, Proc. 3rd Int. Workshop of Digital Mammography, 1996.

[20] M. Mirmehdi, X. Xie, J. Suri, (eds.), *Handbook of Texture Analysis*, Imperial College Press., December 2008.

[21] Y. Rubner, C. Tomasi, L. Guibas, *The Earth Mover's Distance as a Metric for Image Retrieval*, International Journal of Computer Vision, v.40 n.2, p.99-121, Nov. 2000.

[22] I. Kavdır, *Discrimination of sunflower, weed and soil by artificial neural networks*, Computers and Electronics in Agriculture 44, 153–160, 2004.

[23] P. Li, J. R. Flenley, *Pollen texture identification using neural networks*, Grana, 38:59-64, 1999.

[24] N. Krasnogor, D. A. Pelta, *Measuring the similarity of protein structures by means of the universal similarity metric*, Bioinformatics, 20, : 1015–1021, 2004.

[25] A. Bratko, G. Cormack, B. Filipic, T. Lynam, B. Zupan, *Spam Filtering Using Statistical Data Compression Models*, Journal of Machine Learning Research 7, Dec. 2006.

[26] A. Macedonas , D. Besiris , G. Economou , S. Fotopoulos, *Dictionary based color image retrieval*, Journal of Visual Communication and Image Representation, v.19 n.7, p.464-470, October, 2008.

[27] M. Li, Y. Zhu, Image Classification Via LZ78 Based String Kernel: A Comparative Study, PAKDD, 704-712, 2006.

[28] R. Cilibrasi, P. Vitányi, "*Clustering by Compression,* IEEE Transactions on Information Theory 51, 1523-1545, 2005.

[29] M. Delalandre, J. Ogier, J. Llad´os, *A fast cbir system of old ornamental letter*, In Workshop on Graphics Recognition (GREC), vol. 5046 of Lecture Note in Computer Science (LNCS), pages 135–144, 2008.

[30] M. Li, P. Vitanyi, An Introduction to Kolmogorov Complexity and Its Applications. Second Edition, Springer Verlag, 1997.

[31] Coding of moving pictures and associated audio. *Committee Draft of Standard ISO 11172: ISO/MPEG 90/176*, Dec. 1990.

[32] D. Le Gall, *Mpeg: a video compression standard for multimedia application*, Commun. ACM, vol. 34, no. 4, pp. 46-58, April 1991.

[33] X. Wang, L. Ye, E. J. Keogh, C. R. Shelton, *Annotating historical archives of images*, JCDL, 341-350, 2008.

[34] M. Kampel. And S. Zambanini. *Coin Data Acquisition for Image Recognition*, accepted at 36th Conference on Computer Applications and Quantitative Methods in Archaeology (CAA'08), Budapest, Hungary.

[35] M. Mayo, A. Watson, *Automatic species identification of live moths*. In Ellis et. al, editor, Proc. of the 26th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, 195-202, 2006.

[36] B. S. Manjunath, W. Y. Ma, *Texture Features for Browsing and Retrieval of Image Data*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 18, no. 8, pp. 837-842, 1996.

[37] P. Wu, B. S. Manjunath, S. Newsam, H. D. Shin, *A texture descriptor for browsing and similarity retrieval*, Signal Processing: Image Communication, Volume 16, Issues 1-2, Pages 33-43, September 2000.

[38] T. Leung, J. Malik, *Representing and Recognizing the Visual Appearance of Materials using Three-dimensional Textons*, Int. J. Comput. Vision 43, 1, 29-44. Jun. 2001.

[39] M. Varma, A. Zisserman, *A Statistical Approach to Texture Classification from Single Images*, Int. J. Comput. Vision 62, 1-2, 61-81, Apr. 2005.

[40] B. Campana, Website for this paper http://www.cs.ucr.edu/~bcampana/texture.html.

[41] T. Randen, *Brodatz Textures Image Database*, http://www.ux.uis.no/~tranden/brodatz.html.

[42] P. Brodatz, Textures: *A Photographic Album for Artists and Designers*, New York: Dover, 1966.

[43] E. Baudrier, S. Busson, S. Corsini, M. Delalandre, J. Landre, F. Morain-Nicolier, "Retrieval of the Ornaments from the Hand-Press Period: An Overview," Document

Analysis and Recognition, International Conference on, pp. 496-500, 2009 10th International Conference on Document Analysis and Recognition, 2009

[44] Center for Artificial Intelligence and Robotics, http://www.cairo-aisb.com/.

[45] J. Tou, P. Lau, Y. Tay. *Computer Vision-based Wood Recognition System*, Proceedings of International Workshop on Advanced Image Technology (IWAIT 2007), pp. 197-202, 2007.

[46] O. Silven, M. Niskanen, H. Kauppinen, *Wood inspection with non-supervised clustering*, COST action E10 Workshop - Wood properties for industrial use, 18-22, Espoo, Finland, June 2000.

[47] S. Lazebnik, C. Schmid, J. Ponce, *A Sparse Texture Representation Using Local Affine Regions*, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27, no. 8, pp. 1265-1278, August 2005.

[48] MIT Vision and Modeling Group, http://vismod.media.mit.edu/vismod/.

[49] E. Hayman, B. Caputo, M. Fritz, J. O. Eklundh, *On the significance of real-world conditions for material classification*, In: Proc. European Conf. on Computer Vision, No. 3, Springer-Verlag, pp. 253-266, 2004.

[50] K. J. Dana, B. van Ginneken, S. K. Nayar, J. J. Koenderink, *Reflectance and texture of real-world surfaces*, ACM Trans. Graph. 18, 1, 1999.

[51] C. Faloutsos, K. Lin, FastMap: A Fast Algorithm for Indexing, Data-Mining, and Visualization of Traditional and Multimedia Datasets, SIGMOD '95.

[52] T. Pavlidis, *Limitations of CBIR* (Keynote Talk). International Conference on Pattern Recognition, 8-11, Tampa, Florida, Dec. 2006.

[53] T. Wiegand, G. J. Sullivan, G. Bjontegaard, A. Luthra, *Overview of the h.264/avc video coding standard*. Circuits and Systems for Video Technology, IEEE Trans 13(7):560–576, 2003.

[54] Y. Hao, B. Campana, E. Keogh. Monitoring and Mining Insect Sounds in Visual Space. SDM 2012.

[55] A. Barron, J. Rissanen and B. Yu, The minimum description length principle in coding and modeling, IEEE Trans. Information Theory, vol. 44, no. 6, pp. 2743-2760, 1998.

[54] J. Brotzge and S. Erickson, Tornadoes without NWS warning, Weather Forecasting, 25, 159-172. 2010.

[55] P. Chen, et al.. A language-based approach to indexing heterogeneous multimedia lifelog, ICMI 2009.

[56] P. Cohen and N. Adams, An algorithm for segmenting categorical time series into meaningful episodes, ICAIDA 2001, p.198-207.

[57] G. Das, K. Lin, H. Mannila, G. Renganathan, P. Smyth, Rule discovery from time series, KDD 1998: 16-22.

[58] H. Ding, G. Trajcevski, P. Scheuermann, X. Wang, and E.J Keogh, Querying and mining of time series data: experimental comparison of representations and distance measures, PVLDB 1(2): 1542-52, 2008.

[59] S. C. Evans et al, MicroRNA target detection and analysis for genes related to breast cancer using MDLcompress, EURASIP J. Bioinform. Syst. Biol., 1–16, 2007.

[60] B. Ferrell and S. Santuro, NASA shuttle valve data, http://cs.fit.edu/~pkc/nasa/data/. 2005.

[61] E. Gribovskaya, A. Kheddar, and A. Billard, Motion learning and Adaptive Impedance for Robot Control during Physical Interaction with Humans. Preprint version on Gribovskaya's webpage.

[62] E. Keogh and J. Lin, Clustering of time-series subsequences is meaningless: implications for previous and future research, Knowl. Inf. Syst. 8(2), 2005, 154-177.

[63] S. Kojima and A. Doupe, Social performance reveals unexpected vocal competency in young songbirds, Proceedings of the National Academy of Science, Vol 108 pp 1687-1692, 2011.

[64] A. Kornai, Mathematical linguistics, Springer, 2008.

[65] P. D. Grunwald and I.J. Myung. Advances in minimum description length theory and applications, MIT Press, 2003.

[66] I. Jonyer, L. B. Holder, and D. J. Cook, Attribute-value selection based on minimum description length, IC-AI, 2004.

[67] G. Li, S. Ji, C. Li, and J. Feng, Efficient type-ahead search on relational data: a TASTIER approach, SIGMOD 2009: 695-706.

[68] S. Makridakis, S. Wheelwright, and R.J. Hyndman, Forecasting: methods and applications, New York: John Wiley & Sons, 1998.

[69] A. McGovern, D. H. Rosendahl, R. A. Brown, and K. K. Droegemeier, Identifying predictive multi-dimensional time series motifs: an application to severe weather prediction, DMKD 2010.

[70] A. Mueen, E. Keogh, Q. Zhu, S. Cash, and B. Westover, Exact discovery of time series motif, SDM, 2009.

[71] M.L. Phan, C. L. Pytte, and D. S. Vicario, Early auditory experience generates long-lasting auditory memories that may subserve vocal learning in songbirds, Proc Natl Acad Sci USA, 103:1088-93, 2006.

[72] S. Park and S. W. Chu. Discovering and matching elastic rules from sequence databases, Fundam. Inform. 47, 75-90, 2001.

[73] A. Parnandi, et al. Coarse in-building localization with smartphones, Mobiecase 2009.

[74] P. D. E. Pednault, Some experiments in applying inductive inference principles to surface reconstruction, IJCAI 1989: 1603-1609.

[75] T. Rakthanmanon, B. Campana, A. Mueen, G. Batista, B. Westover, Q. Zhu, J. Zakaria, E. Keogh (2012). Searching and Mining Trillions of Time Series Subsequences under Dynamic Time Warping. SIGKDD 2012.

[76] S. Saar and P. P. Mitra. A technique for characterizing the development of rhythms in bird song, PLoS ONE 3, e1461.

[77] D. Sart, A. Mueen, W. A. Najjar, E.J. Keogh, and V. Niennattrakul. Accelerating dynamic time warping subsequence search with GPUs and FPGAs, ICDM 2010.

[78] Y. Tanaka et al. Discovery of time-series motif from multi-dimensional data based on MDL principle, Machine Learning, 58(2), 2005.

[79] UCR time series and classification and clustering. http://www.cs.ucr.edu/~eamonn/time_series_data/

[80] J. T. Wang, B. A. Shapiro, and D. E. Shasha, Pattern discovery in biomolecular data: tools, Oxford University Press, 1999.

[81] S. Weiss, N. Indurkhya, and C. Apte, Predictive rule discovery from electronic health records, ACM IHI, 2010.

[82] H. Wu, B. Salzberg, and D. Zhang, Online event-driven subsequence matching over financial data streams, SIGMOD 2004: 23-34.

[83] H. Wu, personal email communication, 2005.

[84] Z. Zhu, X. Zhu, Y. Guo, and X. Xue, Transfer incremental learning for pattern classification, CIKM 2010: 1709-1712, 2010.

[85] Project URL: http://www.cs.ucr.edu/~bcampana/rulefinding.php