Important Note

These notes are provisional

Update notes will be available at the conference, both electronically and (some) paper copies.

Fair Use Agreement

This agreement covers the use of all slides, please read carefully.

- You may freely use these slides for teaching, if
  * Your name and email address appears on the first slide (if you are using all or most of the slides), or on each slide (if you are just taking a few slides)
  * You send me an email telling me the class number/university in advance.
- You may freely use these slides for a conference presentation, if
  * Your name appears on each slide you use.
- You may not use these slides for tutorials, or in a published work (tech report/conference paper/thesis/journal etc), if you wish to do this, email me first, it is highly likely I will grant you permission.

(c) Eamonn Keogh, eamonn@cs.ucr.edu

Outline of Tutorial I

- Introduction, Motivation
- The ubiquity of time series and shape data
- Examples of problems in time series and shape data mining
- The utility of distance measurements
- Properties of distance measures
  - Euclidean distance
  - Dynamic time warping
  - Longest common subsequence
  - Why no other distance measures?
- Preprocessing the data
- Invariance to distortions
- Spatial Access Methods and the curse of dimensionality
- Generic dimensionality reduction
  - Discrete Fourier Transform
  - Discrete Wavelet Transform
  - Singular Value Decomposition
  - Adaptive Piecewise Constant Approximation
  - Piecewise Linear Approximation
  - Piecewise Aggregate Approximation
- Why Symbolic Approximation is different
  - Why SAX is the best symbolic approximation

Come, we shall learn of the mining of time series

Machine Learning in Time Series Databases (and Everything Is a Time Series!)
AAAI Tutorial 2011
Eamonn Keogh, UCR
eamonn@cs.ucr.edu

Very Briefly
Outline of Tutorial II

In both shape and time series, we consider:

- Novelty detection (finding unusual shapes or subsequences)
- Motif discovery (finding repeated shapes or subsequences)
- Clustering
- Classification
- Indexing
- Visualizing massive datasets
- Open problems to solve
- Summary, Conclusions

The Ubiquity of Shape

- Butterflies, fish, petroglyphs, arrowheads, fruit fly wings, lizards, nematodes, yeast cells, faces, historical manuscripts...

The Ubiquity of Time Series

- Motion capture, meteorology, finance, handwriting, medicine, web logs, music...

Don't Shoot!

Examples of problems in time series and shape data mining

- In the next few slides we will see examples of the kind of problems we would like to be able to solve, then later we will see the necessary tools to solve them.
All our Experiments are Reproducible!

People that do irreproducible experiments should be boiled alive

Agreed! All experiments in this tutorial are reproducible

We can take two different families of butterflies, *Limenitidinae* and *Danainae*, and find the most similar shape between them.

Example 1: Join

Given two data collections, link items occurring in each

- *Limenitidinae*
- *Danainae*
- *Catagaul peguina*
- *Chrysalis*  
- *Danaus affinis*
- *Euploea camaralzeman*
- *Limenitis archippus*
- *Limenitis reducta*
- *Tellervo zoilus*
- *Aterica galene*
- *Catuna crithea*
- *Placidina euryanassa*
- *Greta morgane*
- *Cyrestri*

We can take two different families of butterflies, *Limenitidinae* and *Danainae*, and find the most similar shape between them.

Why would the two most similar shapes also have similar colors and patterns? That can’t be a coincidence. This is an example of Müllerian mimicry.

*Inferno -- Canto XXIII*  29

Not Batesian mimicry as commonly believed

...so similar in coloration that I will put them both to one*
Example 2: Annotation

Given an object of interest, automatically obtain additional information about it.

Friedrich Bertuch’s Bilderbuch für Kinder (Weimar, 1798–1830)

Bilderbuch is a children’s encyclopedia of natural history, published in 237 parts over nearly 40 years in Germany.

Suppose we encountered this page and wanted to know more about the insect. The back of the page says “Stockinsekt” which we might be able to parse to “Stick Insect”, but what kind? How large is it? Where do they live?

Suppose we issue a query to Google search for “Stick Insect” and further filter the results by shape similarity.

Example 3: Query by Content

Petroglyphs

- They appear worldwide
- Over a million in America alone
- Surprisingly little known about them

who sketched out the shapes there?*

... they would strike the subtlest minds with awe*  

*Paraphrased — Cato XI 6
Two Kinds of Shape Matching

Key Ideas:
- Convert shape to pseudo time series or feature vector. Use time series distance measures or vector distance measures to measure similarity.
- We only consider this approach in this tutorial.
- It works well for the butterflies, fish, patalopyle, armadillos, fruit fly, wings, lizards, nematodes, yeast cells, faces, historical manuscripts etc discussed at the beginning of this tutorial.

We can convert shapes into a 1D signal. Thus we can remove information about scale and offset.

Rotation we must deal with in our algorithms...

For virtually all shape matching problems, rotation is the problem.

If I asked you to group these reptile skulls, rotation would not confuse you.

There are two ways to be rotation invariant:
1) Landmarking: Find the one “true” rotation
2) Rotation invariant features

Example(s) of Human Motion
- Join
- Annotation
- Query-by-Content
- Classification
- Classification
- Anomaly Detection
- Motif Discovery

Two Kinds of Shape Matching

"rigid"

"flexible"

- Some shapes are already “graph-like”
- NEEDED for articulated shapes
- THE shape to graph transformation is very tricky!

We do not further discuss these ideas; see “Shape graphs” work of Shotton, Kita and Kita and the work of Lenclo and others.

Shape Representations

There are many other 1D representations of shape, and the algorithms shown in this tutorial can work with any of them.
Landmarking

- **Generic Landmarking:** Find the major axis of the shape and use that as the canonical alignment.
- **Domain Specific Landmarking:** Find some fixed point in your domain, e.g., the nose on a face, the stem of a flower.

The only problem with landmarking is that it does not work.

**Domain Specific Landmarking**

- Landmarking includes leaf stems, noses, the tip of areoles.

Rotation invariant features

**Possibilities include:**
- Ratios of perimeter to area, fractal measures, elongatedness, circularity, min/max/mean curvature, entropy, perimeter of convex hull, aspect ratio and histograms.

The problem with rotation invariant features is that in throwing away rotation information, you must inexorably throw away useful information.

**The strategy of testing all possible rotations is very very slow**

People have suggested various tricks for speedup, like only testing 1 in 5 of the rotations.

However, there now exists a simple **exact** ultrafast, indexable way to do this.*
The need for rotation invariance shows up in real time series, as in these Star Light Curves.

I saw above a million burning lamps, A Sun kindled every one of them, as our sun lights the stars we glimpse on high.

*The Paradiso -- Canto XXIII 28-30

Shape Distance Measures

Speak to me of the useful distance measures

There are but three…

Euclidean Distance
Dynamic Time Warping
Longest Common Subsequence

Euclidean Distance works well for matching many kinds of shapes.

Dynamic Time Warping is useful for natural shapes, which often exhibit intraclass variability.

Is man an ape or an angel?

Lowland Gorilla
Gorilla gorilla gorilla

Mountain Gorilla
Gorilla gorilla beringei

Marbled Howler Monkey
Alouatta palliata

Red Howler Monkey
Alouatta sara

Mantled Howler Monkey
Alouatta palliata

Red Howler Monkey
Alouatta seniculus seniculus

Euclidean Distance

Mountain Gorilla
Gorilla gorilla beringei

DTW Alignment

Lowland Gorilla
Gorilla gorilla gorilla

Is man an ape or an angel?

Mountain Gorilla
Gorilla gorilla beringei
Matching skulls is an important problem.

Euclidean Distance Metric

Given two time series $Q = q_1 \ldots q_n$ and $C = c_1 \ldots c_n$, the Euclidean distance between them is defined as:

$$D(Q, C) = \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2}$$

I notice that you Z-normalized the time series first.

The next slide shows a useful optimization…

Early Abandon Euclidean Distance

During the computation, if current sum of the squared differences between each pair of corresponding data points exceeds $r^2$, we can safely abandon the calculation.

I see, because incremental value is always a lower bound to the final value, once it is greater than the best-so-far, we may as well abandon.

Abandon all hope ye who enter here.

Dynamic Time Warping I

This is how the DTW alignment is found.

This recursive function gives us the minimum cost path:

$$\gamma(i, j) = d(q_i, c_j) + \min \{ \gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1) \}$$

This is how the DTW alignment is found.

I notice that you Z-normalized the time series first.

The next slide shows a useful optimization…

Early Abandon Euclidean Distance

During the computation, if current sum of the squared differences between each pair of corresponding data points exceeds $r^2$, we can safely abandon the calculation.

I see, because incremental value is always a lower bound to the final value, once it is greater than the best-so-far, we may as well abandon.

Abandon all hope ye who enter here.

Dynamic Time Warping I

This is how the DTW alignment is found.

This recursive function gives us the minimum cost path:

$$\gamma(i, j) = d(q_i, c_j) + \min \{ \gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1) \}$$

This is how the DTW alignment is found.

I notice that you Z-normalized the time series first.

The next slide shows a useful optimization…

Early Abandon Euclidean Distance

During the computation, if current sum of the squared differences between each pair of corresponding data points exceeds $r^2$, we can safely abandon the calculation.

I see, because incremental value is always a lower bound to the final value, once it is greater than the best-so-far, we may as well abandon.

Abandon all hope ye who enter here.

Dynamic Time Warping I

This is how the DTW alignment is found.

This recursive function gives us the minimum cost path:

$$\gamma(i, j) = d(q_i, c_j) + \min \{ \gamma(i-1, j-1), \gamma(i-1, j), \gamma(i, j-1) \}$$

This is how the DTW alignment is found.
Dynamic Time Warping II

There is an important trick to improve accuracy and speed...

This "constrained warping", together with a lower bounding trick called LB_Keogh can make DTW thousands of times faster! But don't take my word for it...

"LB_Keogh is fast, because it cleverly exploits global constraints..."

See the below for more information about constrained warping:

• Ratanamahatana and Keogh. (2004). Everything you know about Dynamic Time Warping is Wrong.

Tests on many diverse datasets

...and I recognized the face

Leaf of mine, in whom I found pleasure

...as a fish dives through water

...the shape of that cold animal which stings and latches people with its tail

Name                       Cluster Instances Touchdown Error (%) DTW Error (%) inser. Other Techniques
Face                        16        2240        3.839           3.170 (3)       17.82 Siderite
Swedish Leaves              15        1125        13.33           10.84 (2)       19.96 (1)
Chicken                     16        446         19.96           19.96 (1)       20.5 Dynamic strings
MixedBag                    16        160         4.375           4.375 (1)       26.0 Morphological Curvature Scale Spaces
OSU Leaves                  16        842         33.71           15.61 (2)       36.0 Curve-Fitting Spline
Diatoms                     17        781         27.53           27.53 (1)       50.60 Fractal Dimension
Plane                       7          210          0.95            0.0 (1)         0.55 Fractal Dimension
Fish                        7          350         11.43           9.71 (1)        36.0 Curve-Fitting Spline

Note that DTW is sometimes worth the little extra effort.
Unlike the primates, reptiles require warping...

Dynamic Time Warping

Unlike the primates, reptiles require warping...

Flat-tailed Horned Lizard
Phrynosoma cornutum

Texas Horned Lizard
Phrynosoma corneum

Data Mining is Constrained by Disk I/O

For example, suppose you have one gig of main memory and want to do K-means clustering...

Clustering ¼ gig of data, 100 sec
Clustering ½ gig of data, 200 sec
Clustering 1 gig of data, 400 sec
Clustering 1.1 gigs of data, 20 hours

The Generic Data Mining Algorithm

• Create an approximation of the data, which will fit in main memory, yet retains the essential features of interest
• Approximately solve the problem at hand in main memory
• Make (hopefully very few) accesses to the original data on disk to confirm the solution obtained in Step 2, or to modify the solution so it agrees with the solution we would have obtained on the original data

But which approximation should we use?

OK, let us take stock of what we have seen so far

• There are interesting problems in shape/time series mining (motifs, anomalies, clustering, classification, query-by-content, visualization, joins).
• Very simple transformations let us treat shapes as time series.
• Very simple distance measures (Euclidean, DTW) work very well.

We are finally ready to see how symbolic representations, in particular SAX, allow us to solve these problems
Some approximations of time series...

..note that all except SYM are real valued...

The Generic Data Mining Algorithm (revisited)

- Create an approximation of the data, which will fit in main memory, yet retains the essential features of interest
- Approximately solve the problem at hand in main memory
- Make (hopefully very few) accesses to the original data on disk to confirm the solution obtained in Step 2, or to modify the solution so it agrees with the solution we would have obtained on the original data

What is Lower Bounding?

- Lower bounding means the estimated distance in the reduced space is always less than or equal to the distance in the original space.

\[
D_{LB}(Q', S') \leq D(Q, S)
\]

This only works if the approximation allows lower bounding
Lower Bounding functions are known for wavelets, Fourier, SVD, piecewise polynomials, Chebyshev Polynomials and clipped data. While there are more than 200 different symbolic or discrete ways to approximate time series, none except SAX allows lower bounding.

Why do we care so much about *symbolic representations*?

Symbolic Representations Allow:
- Hashing
- Suffix Trees
- Markov Models
- Stealing ideas from text processing/bioinformatics community
- etc

There is one symbolic representation of time series, that allows…

- Lower bounding of Euclidean distance
- Lower bounding of the DTW distance
- Dimensionality Reduction
- Numerosity Reduction

That representation is **SAX**

*Symbolic Aggregate ApproXimation*
How do we obtain SAX?

First convert the time series to PAA representation, then convert the PAA to symbols.

It takes linear time.

baabccbc

Note we made two parameter choices

The word size, in this case 8.

The alphabet size (cardinality), in this case 3.

Visual Comparison

A raw time series of length 128 is transformed into the word “ffffffeeddcbbaabcedcbaaaacdddec.”

We can use more symbols to represent the time series since each symbol requires fewer bits than real-numbers (float, double).

SAX Lower Bound to Euclidean Distance Metric

\[ D(Q, C) = \sqrt{\sum_{i=1}^{n} (q_i - c_i)^2} \]

Recall the Euclidean distance?

Yes, here is the function that lower bounds it for SAX, it is called MINDIST

\[ \hat{D} = \text{MINDIST}(Q, C) = \sqrt{\sum (\text{dist}(\hat{q}, \hat{c}))} \]

(\text{dist}) can be implemented using a table lookup.
Data mining problems are I/O bound. The generic data mining algorithm mitigates the problem, if you can obey the lower bounding requirement. There is one approximation of time series that is symbolic and lower bounding, SAX. Being discrete instead of real valued gives SAX some advantages (which we have yet to see).

We are finally ready to see the utility of SAX.

Let us consider the utility of SAX for visualizing time series. We start with an apparent digression, visualizing DNA…. The DNA of two species... Are they similar?
OK. Given any DNA string I can make a colored bitmap, so what?

Note *Elephas maximus* is the Indian Elephant, *Loxodonta africana* is the African elephant.

Pan troglodytes is the chimpanzee.

We call these bitmaps **Intelligent Icons**

Two Questions

- Can we do something similar for time series?
- Would it be useful?
Can we make bitmaps for time series?

Yes, with SAX!

While they are all example of EEGs, example_a.dat is from a normal trace, whereas the others contain examples of spike-wave discharges.

We can further enhance the time series bitmaps by arranging the thumbnails by “cluster”, instead of arranging by date, size, name etc.

We can achieve this with MDS.

A well known dataset Kalpakis_ECG, allegedly contains only ECGS.

If we view them as time series bitmaps, a handful stand out…
Some of the data are not heartbeats! They are the action potential of a normal pacemaker cell.

Bitmaps can be used for anomaly detection.

We can test how much useful information is retained in the bitmaps by using only the bitmaps for clustering.

Think of the implications of this, how unusual this is. Unusual section of the time series which coincidences with the PVC.
Time Series Motif Discovery
(finding repeated patterns)

Are there any repeated patterns, of about this length in the above time series?

Why Find Motifs? I

Finding motifs in motion capture allows efficient editing of special effects, and can be used to allow more natural interactions with video games...

- Tanaka, Y. & Uehara, K.
- Araki, Arita and Taniguchi
- Celly, B. & Zordan, V. B.

To see the full video go to...
www.cs.ucr.edu/~eamonn/SIGKDD07/UniformScaling.html
Or search YouTube for "Time series motifs"

... …...

Finding motifs in motion capture allows efficient editing of special effects, and can be used to allow more natural interactions with video games...

Why Find Motifs? II

- Mining association rules in time series requires the discovery of motifs. These are referred to as primitive shapes and frequent patterns.
- Several time series classification algorithms work by constructing typical prototypes of each class. These prototypes may be considered motifs.
- Many time series anomaly/interestingness detection algorithms essentially consist of modeling normal behavior with a set of typical shapes (which we see as motifs), and detecting future patterns that are dissimilar to all typical shapes.
- In robotics, Oates et al., have introduced a method to allow an autonomous agent to generalize from a set of qualitatively different experiences gleaned from sensors. We see these “experiences” as motifs. See also Murakami Yoshikazu, Doki & Okuma and Maja J Mataric
- In medical data mining, Caraca-Valente and Lopez-Chavarrias have introduced a method for characterizing a physiotherapy patient’s recovery based of the discovery of similar patterns. Once again, we see these “similar patterns” as motifs.
Definition 1. Match: Given a positive real number \( R \) (called range) and a time series \( T \) containing a subsequence \( C \) beginning at position \( p \) and a subsequence \( M \) beginning at \( q \), if \( D(C, M) \leq R \), then \( M \) is called a matching subsequence of \( C \).

Definition 2. Trivial Match: Given a time series \( T \), containing a subsequence \( C \) beginning at position \( p \) and a matching subsequence \( M \) beginning at \( q \), we say that \( M \) is a trivial match to \( C \) if either \( p = q \) or there does not exist a subsequence \( M' \) beginning at \( q' \) such that \( D(C, M') > R \), and either \( q < q' < p \) or \( p < q' < q \).

Definition 3. \( K \)-Motif \((n, R)\): Given a time series \( T \), a subsequence length \( n \) and a range \( R \), the most significant motif in \( T \) (hereafter called the 1-Motif \((n, R)\)) is the subsequence \( C_1 \) that has the highest count of non-trivial matches (ties are broken by choosing the motif whose matches have the lower variance). The \( K \)th most significant motif in \( T \) (hereafter called the \( K \)-Motif \((n, R)\) ) is the subsequence \( C_K \) that has the highest count of non-trivial matches, and satisfies \( D(C_{j^*}, C_j) > 2R \), for all \( 1 \leq j < K \).

OK, we can define motifs, but how do we find them?

The obvious brute force search algorithm is just too slow…

The most reference algorithm is based on a hot idea from bioinformatics, random projection* and the fact that SAX allows use to lower bound discrete representations of time series.

A mask \{2,4\} was randomly chosen, so the values in columns \{2,4\} were used to project matrix into buckets.

Once again, collisions are recorded by incrementing the appropriate location in the collision matrix.

We can now use the information in the collision matrix as a heuristic to hunt for likely motifs.

We can use lower bounding to discover at what point that hunt is fruitless…

This is a good example of the Generic Data Mining Algorithm…

A Simple Experiment

Let us imbed two motifs into a random walk time series, and see if we can recover them.

Planted Motifs
Shape Motifs I

We can find shape motifs with only minor modifications:

- When converting shape to SAX, try all rotations to fit best fit.
- Place every circular shift of SAX word in the projection matrix.

Shape Motifs II

Image Discords

What is the most unusual shape in this collection?

This one!

Image Discords

Shape Discord: Given a collection of shapes S, the shape D is the discord of S if D has the largest distance to its nearest match. That is, if shape C in S, the nearest match M_c of C and the nearest match M_j of D, Dist(D, M_j) > Dist(C, M_c).
This one is even more subtle... Here is a subset of a large collection of petroglyphs.

Only one image shows an arrow stuck into the sheep.

Finding Image Discords

Function [dist, loc] = Discord_Search(S)
best_so_far_dist = 0
best_so_far_loc = NaN
for p = 1 to size(S)
    nearest_neighbor_dist = infinity
    for q = 1 to size(S)
        if p != q
            if RD(C_p, C_q) < nearest_neighbor_dist
                nearest_neighbor_dist = RD(C_p, C_q)
            end
        end
    end
    if nearest_neighbor_dist > best_so_far_dist
        best_so_far_dist = nearest_neighbor_dist
        best_so_far_loc = p
    end
end
return [best_so_far_dist, best_so_far_loc]

The code says, find the smallest (non-diagonal) value in each column, the largest of these is the discord.
Function \[ \text{Heuristic\_Search}(S, \text{Outer}, \text{Inner}) \]

best\_so\_far\_dist = 0

best\_so\_far\_loc = NaN

for each index \( p \) given by heuristic \text{Outer} // begin outer loop

nearest\_neighbor\_dist = infinity

for each index \( q \) given by heuristic \text{Inner} // begin inner loop

if \( p \neq q \)

if \( RD(C_p, C_q) < \text{best\_so\_far\_dist} \)

break // break out of inner loop

end

if \( RD(C_p, C_q) < \text{nearest\_neighbor\_dist} \)

nearest\_neighbor\_dist = RD(C_p, C_q)

end

end // end inner loop

if nearest\_neighbor\_dist > best\_so\_far\_dist

best\_so\_far\_dist = nearest\_neighbor\_dist

best\_so\_far\_loc = \( p \)

end

end // end outer loop

return [best\_so\_far\_dist, best\_so\_far\_loc]

Finding Discords, Fast

The Magic Heuristics

• In the outer loop, visit the columns in order of the Discord score
• In the inner loop, visit the row cells in order of nearest neighbor first

Observations

• Visiting the columns in approximately order of the Discord score is still very helpful
• For the inner loop, we don’t really need visit the rows in order of nearest neighbor first, so long as we find a “near enough” neighbor early on

Approximately Magic Heuristics

The code now says... If while searching a given column, you find a distance less than nearest\_neighbor\_dist then that column cannot have the discord.

The code also uses heuristics to order the search...

The Magic Heuristics would reduce the time complexity from \( O(n^2) \) algorithm to just \( O(n) \)!
How Fast is Approximately Magic?

On a problem dataset of arrowheads

- If we only see 200 arrowheads, we do an extra 21.8% more work than the Magic algorithm
- For larger arrowhead datasets we get even closer to Magic algorithm
- In other words, we are doing $O(n)$ work, not $O(n^2)$ work.
- Empirically we see similar results for other datasets, but in pathological datasets, we can still be forced to do $O(n^2)$ work.

Which is the “odd man out” in this collection of Red Passion Flower Butterflies?

One of them is not a Red Passion Flower Butterfly. A fact that can be discovered by finding the shape discord.

Nematode Discords

A B C D E F

G

Through 25,000 species have been classified it is estimated that this number might be upwards of 500,000 if all were known. Wikipedia

Fungus Images

Some spores produced by a rust (fungus) known as Gymnosporangium, which is a parasite of apple and pear trees. Note that one spore has sprouted an “appendage” known as a germ tube, and is thus singled out as the discord.

A subset of 32,028 images of Drosophila wings.
A time series showing a patient's respiration (measured by thorax extension), as they wake up. A medical expert, Dr. J. Rittweger, manually segmented the data. The 1st discord is a very obvious deep breath taken as the patient opened their eyes. The 2nd discord is much more subtle and impossible to see at this scale. A zoom-in suggests that Dr. J. Rittweger noticed a few shallow breaths that indicated the transition of sleeping stages.

Institute for Physiology. Free University of Berlin. Data shows respiration (thorax extension), sampling rate 10 Hz.

A cardiologist noted subtle anomalies in this dataset. Let us see if the discord algorithm can find them.

The discord is subtle, lets zoom in to see why it is a discord.

Discords in Space Shuttle Marotta Valve Series

Example One

Example Two

This discord is subtle, lets zoom in to see why it is a discord.

Open Problems

- Let us finish with a brief discussion of some open problems worthy of study.
Assessing the Significance of Motifs/Discords

The motif and discord algorithms always return some answer, but is the result interesting, or something we should have expected by chance?

In a large string database, like this \texttt{ABBANBCJSMBAVSMABG}, would it be more interesting to find…

A motif pair \{\texttt{ABBA}, \texttt{ABBA}\}

A motif pair \{\texttt{ABBAACCC}, \texttt{ABBCCCC}\}

(i.e. shorter but perfect or longer with some misspellings)
It makes sense that the bursts for “LeTour”, “Tour de France” and “Lance Armstrong” are all related. But what caused the extra interest in Lance Armstrong in August/September 2000?

Example by M. Vlachos

---

We are done!
We have seen that SAX is a very useful tool for solving problems in shape and time series data mining. I will be happy to answer any questions...

What are the disadvantages of using SAX
There are NaN

Thanks to my students
Eamonn Keogh: UCR
eamonn@cs.ucr.edu
Li Wei (Google)
Jessica Lin
George Mason University
Dragomir Yankov
Chulalongkorn University
Xiaopeng Xi (Yahoo)
Chotirat (Ann) Ratanasukabandhu Chulalongkorn University
Appendix A

• Converting a long time series to a time series bitmap (Intelligent Icon)

Just create random walk of length 40 for testing.
Convert to SAX, with a sliding window of length 16, a word size of 8 and a cardinality of 4

I have converted to "DNA" for visual clarity. Obviously we don't really need to do this.

Count the frequency of all pair of basepairs.
Below I have just done AA and AC
Assign the results to a matrix z

z =
We need to normalize the matrix \( z \), below is one way to do it such that the min value is 0 and the max values is 1. (matlab code)

There may be better ways to normalize...

\[
\text{>> } z=(z-\text{min}(\text{min}(z)));
\]

\[
\text{>> } z=(z/\text{max}(\text{max}(z)))
\]

\[
z = \\
\begin{array}{cccc}
1.0000 & 0.9369 & 0.8618 & 0.9696 \\
0.2282 & 0.7982 & 0.4575 & 0.7725 \\
0.6315 & 0.4701 & 0.6407 & 0.1693 \\
0.5018 & 0.0000 & 0.8302 & 0.4156 \\
\end{array}
\]

We need to normalize the matrix \( z \), below is one way to do it such that the min value is 0 and the max values is 1. (matlab code)

There may be better ways to normalize...

\[
\text{>> } z=(z-\text{min}(\text{min}(z)));
\]

\[
\text{>> } z=(z/\text{max}(\text{max}(z)))
\]

\[
z = \\
\begin{array}{cccc}
1.0000 & 0.9369 & 0.8618 & 0.9696 \\
0.2282 & 0.7982 & 0.4575 & 0.7725 \\
0.6315 & 0.4701 & 0.6407 & 0.1693 \\
0.5018 & 0.0000 & 0.8302 & 0.4156 \\
\end{array}
\]

Map to some colormap, I have done % of the work below...

\[
\begin{array}{cccc}
1.0000 & 0.9369 & 0.8618 & 0.9696 \\
0.2282 & 0.7982 & 0.4575 & 0.7725 \\
0.6315 & 0.4701 & 0.6407 & 0.1693 \\
0.5018 & 0.0000 & 0.8302 & 0.4156 \\
\end{array}
\]

Hints I

\[
\text{ans} = \\
\begin{array}{ccccccc}
G & T & C & T & A & T & C \\
G & C & T & C & A & T & C \\
T & C & T & A & A & G & C \\
A & T & C & A & C & C & T \\
C & C & A & A & T & C & T \\
\end{array}
\]

When counting patterns, don’t count patterns that span two lines.

For example, don’t count the underlined A’s as an occurrence of AA

Hints II

\[
\text{ans} = \\
\begin{array}{ccccccc}
G & T & C & T & A & A & T \\
G & T & C & T & A & A & T \\
G & C & T & C & A & C & T \\
T & C & T & A & A & G & C \\
A & T & C & A & C & C & T \\
C & C & A & A & T & C & T \\
\end{array}
\]

Note that here lines 1 and 2 are the same. This can happen a lot, especially with smooth time series and/or a high compression ratio.

The SAX code has an extra parameter that removes these redundant lines. It seems like this makes the Intelligent Icons work better, and it does make the code run a little faster.
Hints III

For Intelligent Icon the cardinality must be 4
But what is the best sliding window length?
What is the best a word size?

At the moment there is no answer to this other than
playing with the data (or CV if you have labeled data)
The good news is that once you find good settings for
your domain (say ECGs) then the settings should work
for all ECGs.

Heuristics:
The sliding window length should be about twice the
length of the natural scale at which the data is
interesting. For example, about two heartbeats for
cardiology, or for power demand, about two days.
The smoother the data, the smaller you can make the
word size.

Appendix: DTW

• There are some critical facts about the size of the warping window \( r \).
  - \( r \) can vary from 0% (the special case of Euclidean distance) to 100% (the special case of full
    DTW).
  - Without lower bounding, the time taken is approximately linear in \( r \), so \( r < 5\% \) is about twice as
    fast as \( r = 10\% \).
  - With lower bounding, the time taken is highly non-linear in \( r \), so \( r < 5\% \) is perhaps 10 to 100
    times as fast as \( r = 10\% \).
  - In general (empirically measured over 35 datasets) the following is true.
    - If you start with \( r = 0 \) and you make it larger, the accuracy improves, then gets worse (see the
two examples for FACE and GUN in this tutorial, but it is true for other datasets).
    - The best accuracy tends to be at a relatively small value for \( r \) (usually just 2 to 5%)
    - For any dataset, the best value for \( r \) depends on the size of the training set. For example for CBF
      with just 20 instances, you might need \( r = 8\% \), but with 200 instances you only need 1 or 2%, and
      with 2,000 instances, you need \( r = 0\% \) (the Euclidean distance).
    - How do you find the best choice for \( r \)? Use cross validation to test for the best value.

See [a] and [b]

Classification Using Numerosity Reduction. ICML.
Workshop on Mining Temporal and Sequential Data, in conjunction with the Tenth ACM SIGKDD International
Conference on Knowledge Discovery and Data Mining (KDD-2004), August 22-25, 2004 - Seattle, WA