Smart Analytics for Big Time-series Data

Yasushi Sakurai (Kumamoto University)
Yasuko Matsubara (Kumamoto University)
Christos Faloutsos (Carnegie Mellon University)

Roadmap

– Motivation
– **Similarity search, pattern discovery and summarization** — **Part 1**
– Non-linear modeling and forecasting — **Part 2**
– Extension of time-series data: tensor analysis — **Part 3**

Part 1

Similarity search, pattern discovery and summarization

Yasushi Sakurai (Kumamoto University)
Yasuko Matsubara (Kumamoto University)
Christos Faloutsos (Carnegie Mellon University)

Part 1 - Roadmap

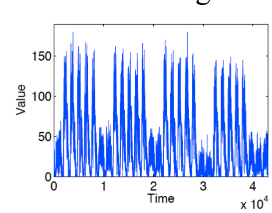
- Motivation
- Similarity Search and Indexing
- Feature extraction
- Linear forecasting
- Streaming pattern discovery
- Automatic mining

Motivation - Applications

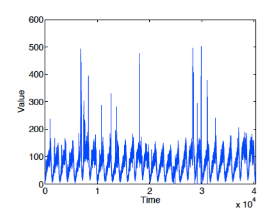
- Web online activities
  - Web access logs
  - Search volume
  - Online reviews
- IoT device data
- Medical, healthcare data

Motivation - Applications

- Web access logs



Web clicks (business news site)

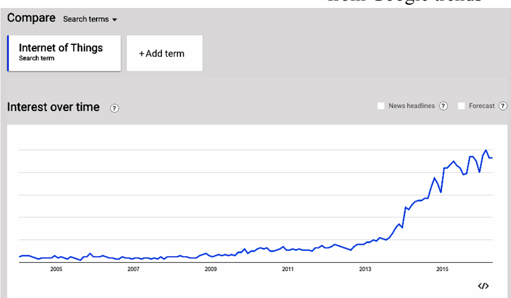


Ondemand TV (access count of users)

Kumamoto U | CMU CS

Motivation - Applications

- Web search volume from Google trends

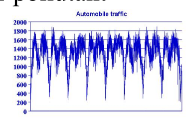


Kumamoto U | CMU CS

Motivation - Applications

- IoT (Internet of Things) device data
  - Civil/automobile infrastructure
  - Bridge vibrations [Oppenheim+02]
  - Road conditions / traffic monitoring
  - Environmental data (air/water pollutant monitoring)


Tokyo Gate Bridge

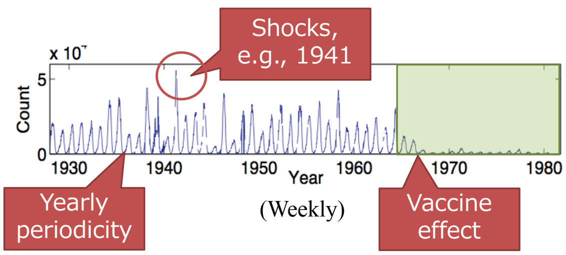


Kumamoto U | CMU CS

Motivation - Applications

- Medical (epidemic) time-series data

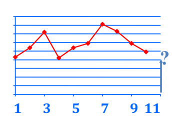
  e.g., measles cases in the U.S.

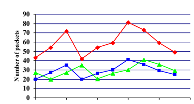


Kumamoto U | CMU CS

Wish list

- Problem 1: find patterns/**rules**
- Problem 2: **forecast**
- Problem 3: find patterns/rules/forecast, with **many** time sequences



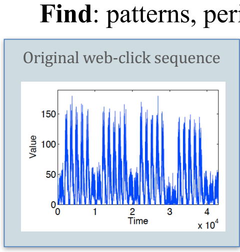


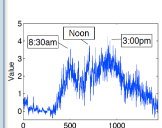
Kumamoto U | CMU CS

Problem #1

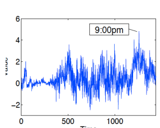
Given: time-series data (e.g., #clicks over time)

Find: patterns, periodicities, and/or compress





Weekday component

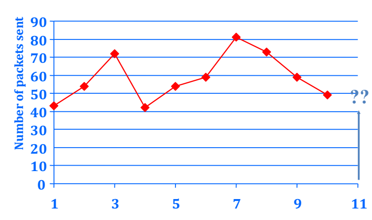


Weekend component

Kumamoto U | CMU CS

Problem #2

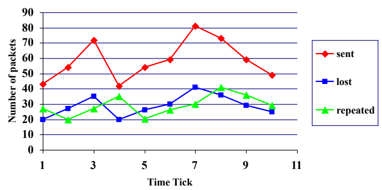
Given x_t, x_{t-1}, …, **forecast** x_{t+1}



Kumamoto U CMU CS

Problem #3

- **Given**: A set of **correlated** time sequences
- **Forecast** '**Repeated(t)**'

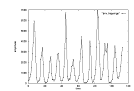


Kumamoto U CMU CS

Important observations

Patterns, outliers, modeling, forecasting and similarity indexing are closely related:

- For forecasting, we need
  - patterns/rules/models
  - similar past settings
- For outliers, we need to have forecasts
  - (outlier = too far away from our forecast)



Kumamoto U CMU CS

Important topics NOT in this tutorial:

- Continuous queries
  - [Babu+Widom] [Gehrke+] [Madden+]
- Categorical data streams
  - [Hatonen+96]
- Outlier detection (discontinuities)
  - [Breunig+00]

Kumamoto U CMU CS

Roadmap

- Motivation
- Similarity Search and Indexing
- Feature extraction
- Linear forecasting
- Streaming pattern discovery
- Automatic mining

Kumamoto U CMU CS

Roadmap

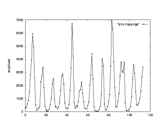
- Motivation
- Similarity Search and Indexing
  - distance functions: Euclidean, time-warping
  - indexing
- Feature extraction
- Linear forecasting
- Streaming pattern discovery
- Automatic mining

Kumamoto U CMU CS

Importance of distance functions

Subtle, but **absolutely necessary**:

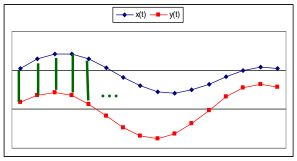
- A 'must' for similarity search, indexing and clustering



Two major families
- Euclidean and Lp norms
- Time warping and variations

Kumamoto U — CMU CS

Euclidean and Lp



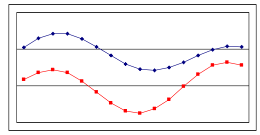
$$D(\vec{x},\vec{y}) = \sum_{i=1}^{n} (x_i - y_i)^2$$

$$L_p(\vec{x},\vec{y}) = \sum_{i=1}^{n} |x_i - y_i|^p$$

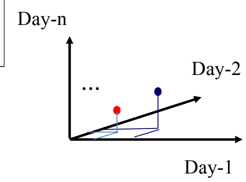
- L_1: city-block = Manhattan
- L_2 = Euclidean
- L_∞

Kumamoto U — CMU CS

Observation #1



- Time sequence
  -> n-dimensional vector

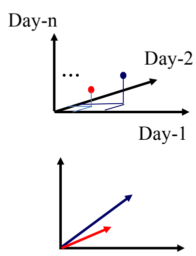


delay coordinate

Kumamoto U — CMU CS

Observation #2

- Euclidean distance is closely related to
  - cosine similarity
  - dot product
  - 'cross-correlation' function



Kumamoto U — CMU CS

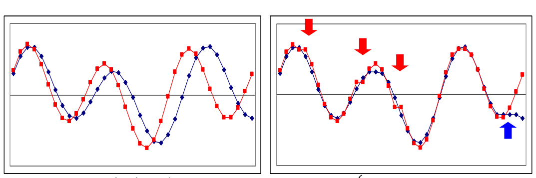
Time Warping

- allow accelerations - decelerations
  - (with or w/o penalty)
- THEN compute the (Euclidean) distance (+ penalty)
- related to the string-editing distance
- fast search methods [Yi+98] [Keogh+02] [Sakurai+05]

Kumamoto U — CMU CS

Time Warping

- Allow sequences to be stretched along the time axis
  1. minimize the distance of sequences
  2. insert 'stutters' into a sequence
  3. THEN compute the (Euclidean) distance



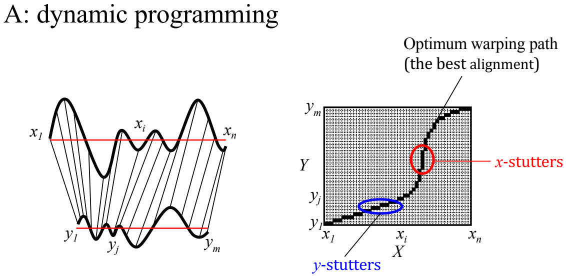
original 'stutters':

Kumamoto U — CMU CS

Time Warping

Q: how to compute it?

A: dynamic programming



Time Warping — DETAILS

Q: how to compute it?

A: dynamic programming

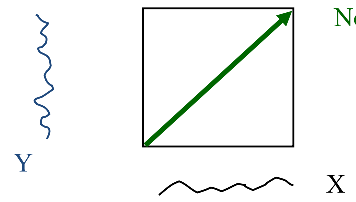
$$X=\{x_1, x_2, \ldots, x_i\},\ Y=\{y_1, y_2, \ldots, y_j\}$$

$$D_{dtw}(X,Y) = f(n,m)$$

$$f(i,j) = \|x_i - y_j\| + \min \begin{cases} f(i,j-1) & x\text{-stutter} \\ f(i-1,j) & y\text{-stutter} \\ f(i-1,j-1) & \text{no stutter} \end{cases}$$

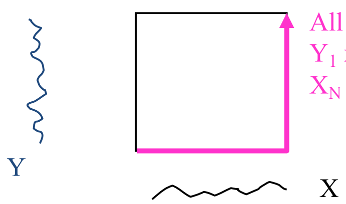
Time Warping

- Time warping matrix & optimal path:



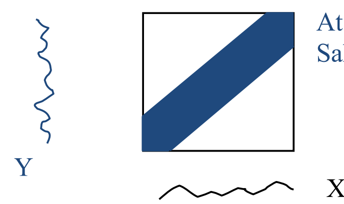
Time Warping

- Time warping matrix & optimal path:



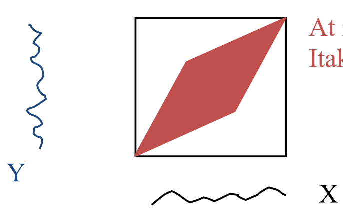
Time Warping - variations

- Time warping matrix & optimal path:



Time Warping - variations

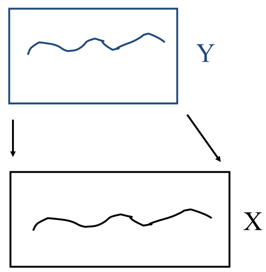
- Time warping matrix & optimal path:



Time warping

- Complexity: O(M*N) - quadratic on the length of the strings
- **Many** variations (penalty for stutters; limit on the number/percentage of stutters; …)
- popular in voice processing [Rabiner+Juang]

A variation: Uniform axis scaling



- Stretch / shrink time axis of Y, up to p%, for free
- THEN compute Euclidean distance
- [Keogh+, VLDB04]

Other Distance functions

- piece-wise linear/flat approx.; compare pieces [Keogh+01] [Faloutsos+97]
- 'cepstrum' (for voice [Rabiner+Juang])
  - do DFT; take log of amplitude; do DFT again!
- Allow for small gaps [Agrawal+95]

Related work

- Chen + Ng [vldb' 04]: ERP 'Edit distance with Real Penalty': give a penalty to stutters
- Keogh+ [kdd' 04]: VERY NICE, based on information theory: compress each sequence (quantize + Lempel-Ziv), using the **other** sequences' LZ tables
- Rakthanmanon+ [kdd' 12]: EXCELLENT Software, the UCR Suite for ultrafast subsequence search

Conclusions

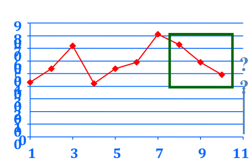
- Prevailing distances:
  - Euclidean and
  - time-warping

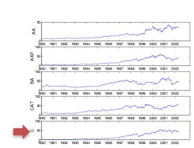
Roadmap

- Motivation
- Similarity Search and Indexing
  - distance functions: Euclidean, time-warping
  - indexing
- Feature extraction
- Linear forecasting
- Streaming pattern discovery
- Automatic mining

Indexing

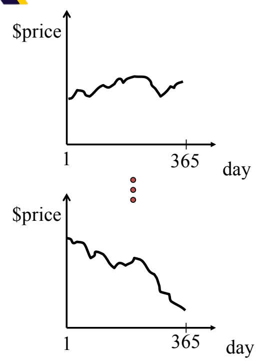
- Given a set of time sequences,
- Find the ones similar to a desirable query sequence

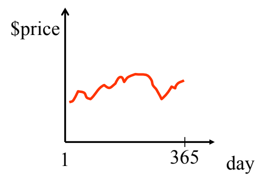




Kumamoto U — CMU CS

Indexing





distance function: by expert

(Euclidean; DTW; …)

Kumamoto U — CMU CS

Idea: 'GEMINI'

Eg., '*find stocks similar to MSFT*'

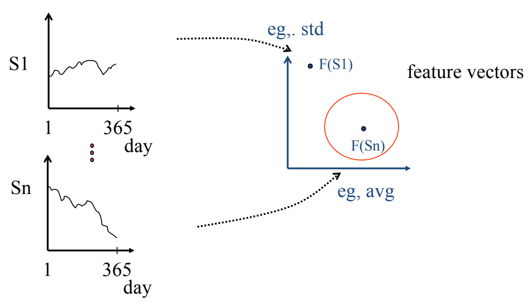
Seq. scanning: too slow

How to accelerate the search?

[Faloutsos96]

Kumamoto U — CMU CS

'GEMINI' - Pictorially



Kumamoto U — CMU CS

GEMINI

Solution: Quick-and-dirty' filter:

- extract *d* features (numbers, eg., avg., etc.)
- map into a point in the *d*-dimensional feature space
- organize points with off-the-shelf spatial access method ('SAM' – R-tree, etc)
- discard false alarms

Kumamoto U — CMU CS

Examples of GEMINI

- Time sequences: DFT (up to 100 times faster) [SIGMOD94];
- [Kanellakis+], [Mendelzon+]

Kumamoto U — CMU CS

Indexing - SAMs

Q: How do Spatial Access Methods (SAMs) work?

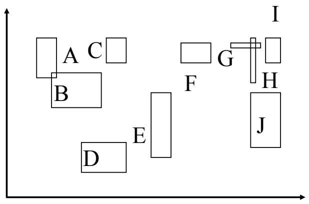
A: they group nearby points (or regions) together, on nearby disk pages, and answer spatial queries quickly ('range queries', 'nearest neighbor' queries etc)

For example:

Kumamoto U CMU CS

R-trees

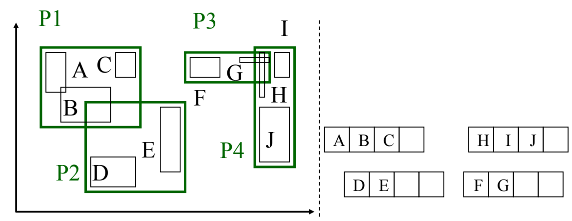
- [Guttman84] eg., w/ fanout 4: group nearby rectangles to parent MBRs; each group -> disk page



Kumamoto U CMU CS

R-trees

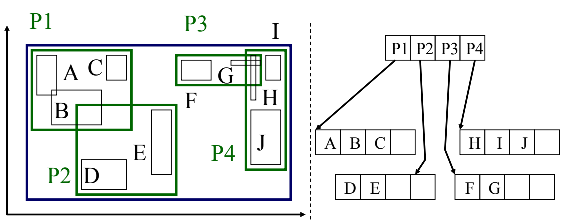
- eg., w/ fanout 4:



Kumamoto U CMU CS

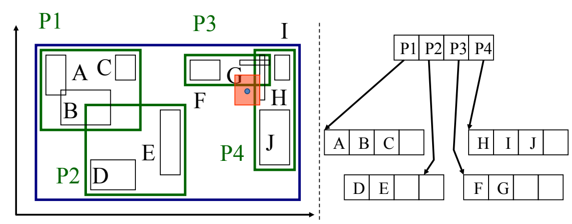
R-trees

- eg., w/ fanout 4:



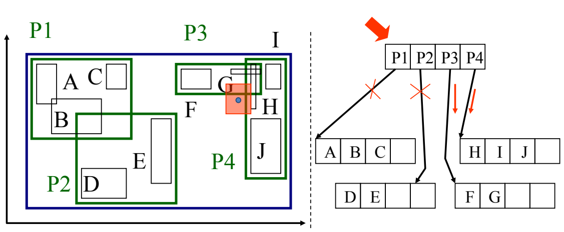
Kumamoto U CMU CS

R-trees - range search?



Kumamoto U CMU CS

R-trees - range search?



Kumamoto U CMU CS

Conclusions

- Fast indexing: through GEMINI
  - feature extraction and
  - (off the shelf) Spatial Access Methods [Gaede+98]

Kumamoto U CMU CS

Roadmap

- Motivation
- Similarity Search and Indexing
- Feature extraction
- Linear forecasting
- Streaming pattern discovery
- Automatic mining

Kumamoto U CMU CS

Roadmap

- Motivation
- Similarity Search and Indexing
- Feature extraction
  - DFT, DWT (data independent)
  - SVD, ICA (data independent)
  - MDS, FastMap
- Linear forecasting
- Streaming pattern discovery
- Automatic mining

Kumamoto U CMU CS

DFT: definition

- For a sequence $x_0, x_1, \ldots x_{n-1}$
- the (**n-point**) Discrete Fourier Transform is
- $X_0, X_1, \ldots X_{n-1}$:

$$X_f = 1/\sqrt{n} \sum_{t=0}^{n-1} x_t * \exp(-j 2\pi \, tf/n) \qquad f = 0, \ldots, n-1$$

$$(j = \sqrt{-1})$$

inverse DFT

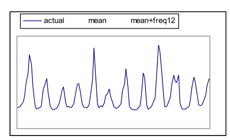
$$x_t = 1/\sqrt{n} \sum_{t=0}^{n-1} X_f * \exp(+j 2\pi \, tf/n)$$

Kumamoto U CMU CS

DFT: Amplitude spectrum

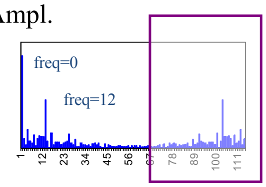
Amplitude: $A_f^2 = \mathrm{Re}^2(X_f) + \mathrm{Im}^2(X_f)$

count



year

Ampl.



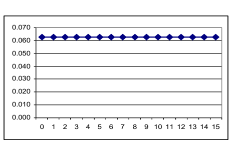
Freq.

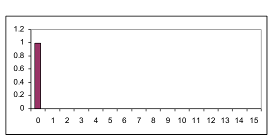
Kumamoto U CMU CS

DFT: examples

- Flat

Amplitude



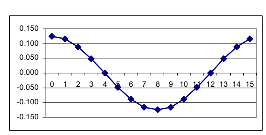


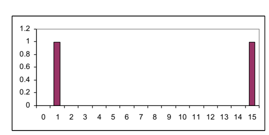
time freq

Kumamoto U CMU CS

DFT: examples

- Low frequency sinusoid



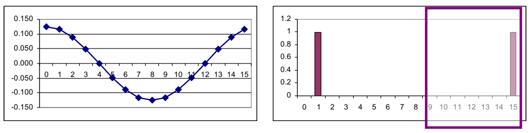


time freq

Kumamoto U — CMU CS

DFT: examples

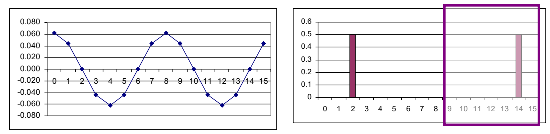
- Sinusoid - symmetry property: $X_f = X^*_{n-f}$



Kumamoto U — CMU CS

DFT: examples

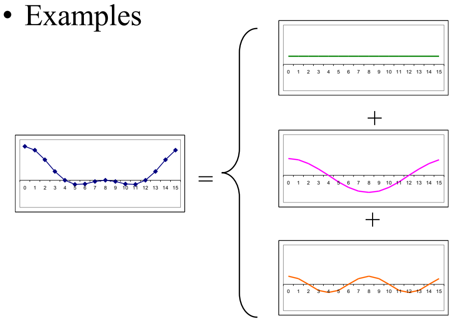
- Higher freq. sinusoid



Kumamoto U — CMU CS

DFT: examples

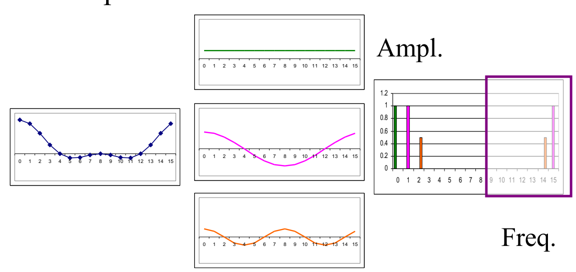
- Examples



Kumamoto U — CMU CS

DFT: examples

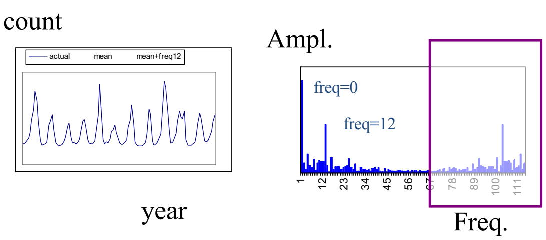
- Examples



Kumamoto U — CMU CS

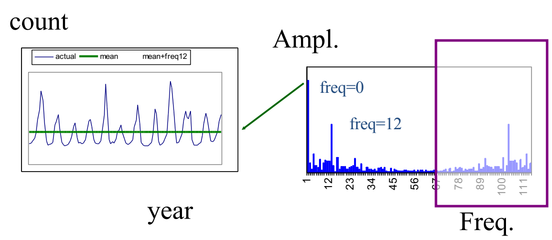
DFT: Amplitude spectrum

Amplitude: $A_f^2 = \mathrm{Re}^2(X_f) + \mathrm{Im}^2(X_f)$



Kumamoto U — CMU CS

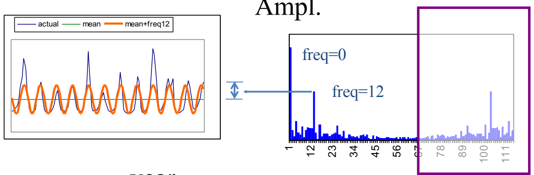
DFT: Amplitude spectrum



DFT: Amplitude spectrum

count

Ampl.

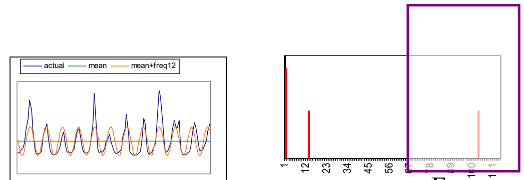


year

Freq.

DFT: Amplitude spectrum

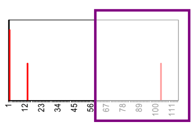
- Excellent approximation, with only 2 frequencies!
- So what?



Freq.

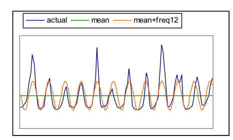
DFT: Amplitude spectrum

- Excellent approximation, with only 2 frequencies!
- So what?
- A1: **(lossy) compression**
- A2: pattern discovery



DFT: Amplitude spectrum

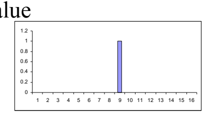
- Excellent approximation, with only 2 frequencies!
- So what?
- A1: (lossy) compression
- A2: **pattern discovery**



Wavelets - DWT

- DFT is great - but, how about compressing a spike?

value

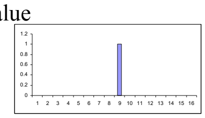


time

Wavelets - DWT

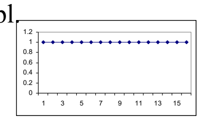
- DFT is great - but, how about compressing a spike?
- A: Terrible - all DFT coefficients needed!

value



time

Ampl.



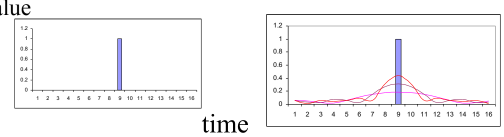
Freq.

Kumamoto U CMU CS

Wavelets - DWT

- DFT is great - but, how about compressing a spike?
- A: Terrible - all DFT coefficients needed!

value



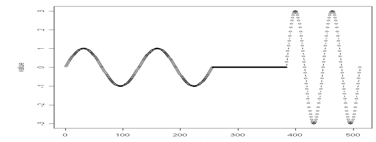
time

Kumamoto U CMU CS

Wavelets - DWT

- Similarly, DFT suffers on short-duration waves (eg., baritone, silence, soprano)

value



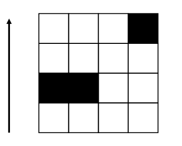
time

Kumamoto U CMU CS

Wavelets - DWT

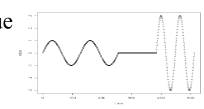
- Solution#1: Short window Fourier transform (SWFT)
- But: how short should be the window?

freq



time

value



time

Kumamoto U CMU CS

Wavelets - DWT

- Answer: **multiple** window sizes! -> DWT

`Multi-scale windows': brilliant idea that we'll see several times in this tutorial (BRAID, TriMine, etc)

Kumamoto U CMU CS

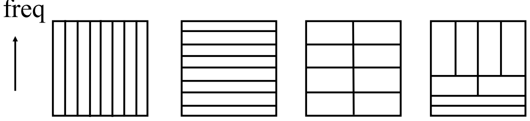
Wavelets - DWT

- Answer: **multiple** window sizes! -> DWT

Multi-scale windows

Time domain DFT SWFT DWT

freq

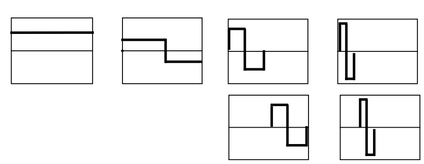


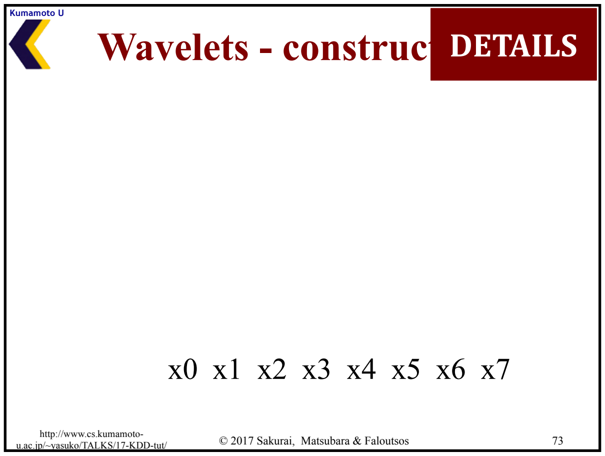
time

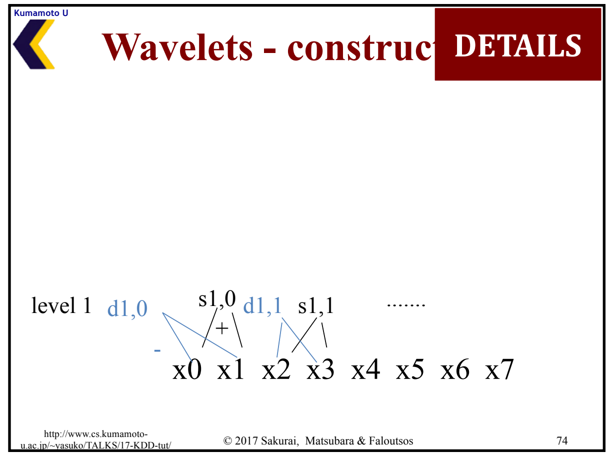
Kumamoto U CMU CS

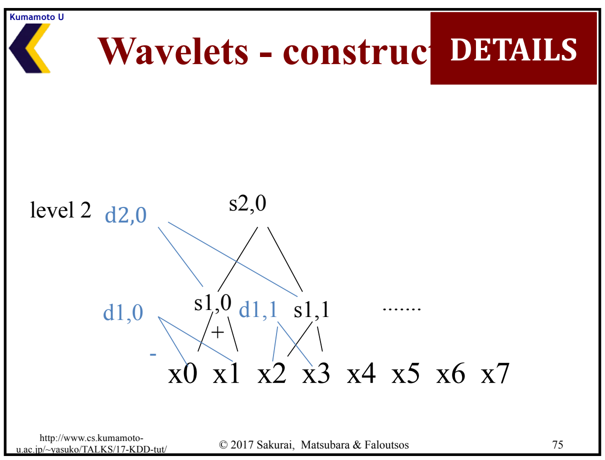
Haar Wavelets

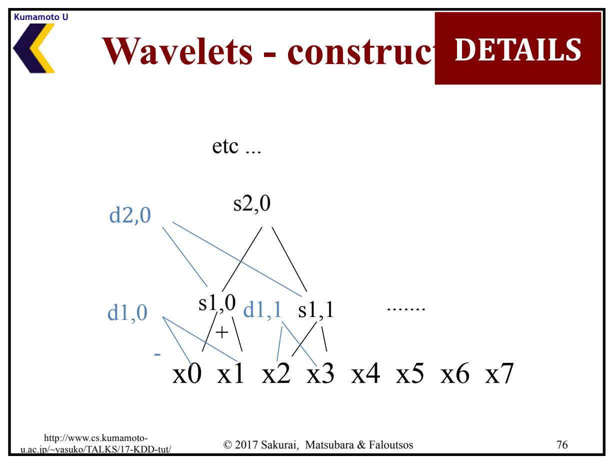
- subtract sum of left half from right half
- repeat recursively for quarters, eight-ths, ...

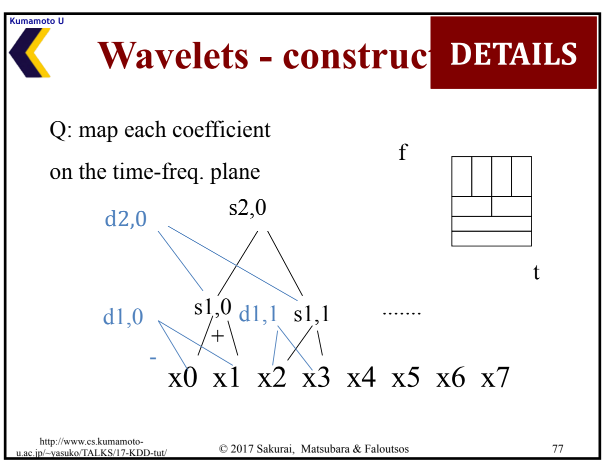


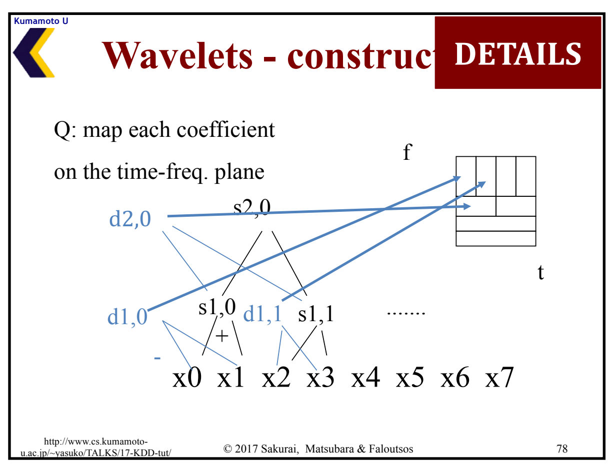

Kumamoto U
Wavelets - construc
DETAILS
x0 x1 x2 x3 x4 x5 x6 x7
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
73


Kumamoto U
Wavelets - construc
DETAILS
level 1
d1,0
s1,0
d1,1
s1,1
....
+
-
x0 x1 x2 x3 x4 x5 x6 x7
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
74


Kumamoto U
Wavelets - construc
DETAILS
level 2
d2,0
s2,0
d1,0
s1,0
d1,1
s1,1
....
+
-
x0 x1 x2 x3 x4 x5 x6 x7
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
75


Kumamoto U
Wavelets - construc
DETAILS
etc ...
d2,0
s2,0
d1,0
s1,0
d1,1
s1,1
....
+
-
x0 x1 x2 x3 x4 x5 x6 x7
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
76


Kumamoto U
Wavelets - construc
DETAILS
Q: map each coefficient
on the time-freq. plane
f
t
d2,0
s2,0
d1,0
s1,0
d1,1
s1,1
....
+
-
x0 x1 x2 x3 x4 x5 x6 x7
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
77


Kumamoto U
Wavelets - construc
DETAILS
Q: map each coefficient
on the time-freq. plane
f
t
d2,0
s2,0
d1,0
s1,0
d1,1
s1,1
....
+
-
x0 x1 x2 x3 x4 x5 x6 x7
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
78

Wavelets - construction DETAILS

Observation1:

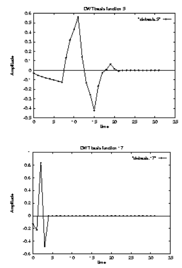
'+' can be some weighted addition

'-' is the corresponding weighted difference ('Quadrature mirror filters')

Observation2: unlike DFT/DCT,

there are *many* wavelet bases: Haar, Daubechies-4, Daubechies-6, Coifman, Morlet, Gabor, ...

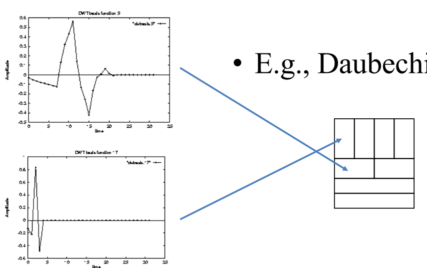
Wavelets - how do they look like?



- E.g., Daubechies-4

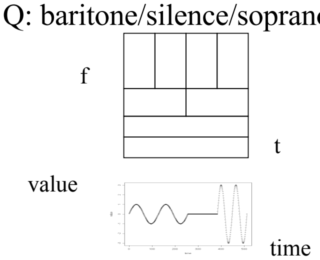
Wavelets - how do they look like?

- E.g., Daubechies-4



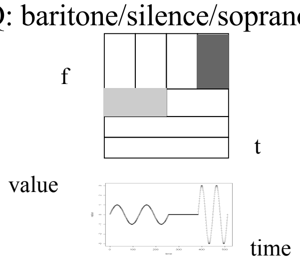
Wavelets - Drill#1:

- Q: baritone/silence/soprano - DWT?



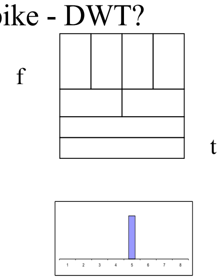
Wavelets - Drill#1:

- Q: baritone/silence/soprano - DWT?



Wavelets - Drill#2:

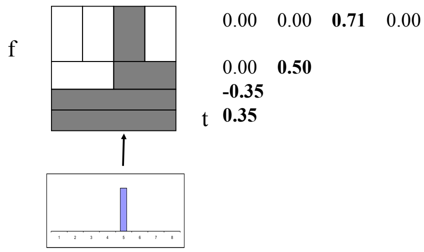
- Q: spike - DWT?



Kumamoto U | CMU CS

Wavelets - Drill#2:

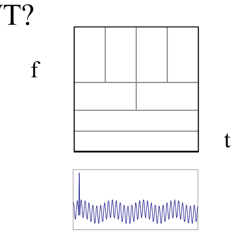
- Q: spike - DWT?



Kumamoto U | CMU CS

Wavelets - Drill#3:

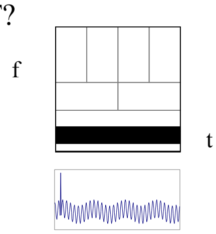
- Q: weekly + daily periodicity, + spike - DWT?



Kumamoto U | CMU CS

Wavelets - Drill#3:

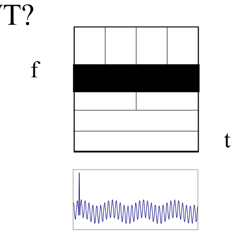
- Q: **weekly** + daily periodicity, + spike - DWT?



Kumamoto U | CMU CS

Wavelets - Drill#3:

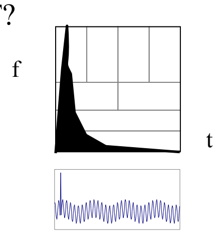
- Q: weekly + **daily** periodicity, + spike - DWT?



Kumamoto U | CMU CS

Wavelets - Drill#3:

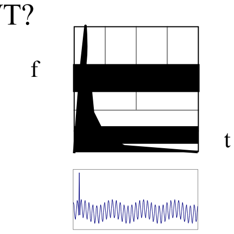
- Q: weekly + daily periodicity, + **spike** - DWT?



Kumamoto U | CMU CS

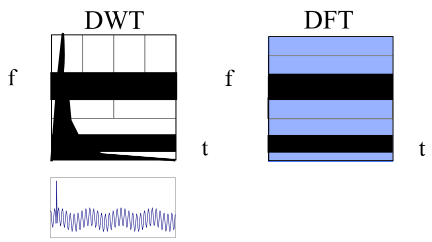
Wavelets - Drill#3:

- Q: weekly + daily periodicity, + spike - DWT?



Wavelets - Drill#3:

- Q: DFT?



Advantages of Wavelets

- Better compression (better RMSE with same number of coefficients - used in JPEG-2000)
- fast to compute (usually: O(*n*)!)
- very good for 'spikes'

DFT & DWT: conclusions

- **DFT** spots periodicities (with the '**amplitude spectrum**')
  - can be quickly computed (O(*n* log *n*)), thanks to the FFT algorithm.
  - **standard** tool in signal processing (speech, image etc signals)
  - (closely related to DCT and JPEG)

DFT & DWT: conclusions

- **DWT**: multi-resolution
  - very suitable for self-similar traffic
  - used for summarization of streams [Gilbert+01], db histograms, etc
- DFT&DWT: powerful tools for **compression**, **pattern detection** in real signals
  - included in math packages (matlab, 'R', mathematica, … - often in spreadsheets!)

Resources - software and urls

- http://www.dsptutor.freeuk.com/jsanalyser/FFTSpectrumAnalyser.html : Nice java applets for FFT
- http://www.relisoft.com/freeware/freq.html voice frequency analyzer (needs microphone)

Resources: software and urls

- *xwpl:* open source wavelet package from Yale, with excellent GUI
- http://monet.me.ic.ac.uk/people/gavin/java/waveletDemos.html : wavelets and scalograms

Kumamoto U CMU CS

Books

- William H. Press, Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery: *Numerical Recipes in C*, Cambridge University Press, 1992, 2nd Edition. (Great description, intuition and code for DFT, DWT)
- C. Faloutsos: *Searching Multimedia Databases by Content*, Kluwer Academic Press, 1996 (introduction to DFT, DWT)

Kumamoto U CMU CS

Additional Reading

- [Gilbert+01] Anna C. Gilbert, Yannis Kotidis and S. Muthukrishnan and Martin Strauss, *Surfing Wavelets on Streams: One-Pass Summaries for Approximate Aggregate Queries*, VLDB 2001

Kumamoto U CMU CS

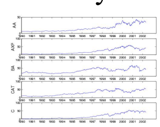
Roadmap

- Motivation
- Similarity Search and Indexing
- Feature extraction
  - DFT, DWT, DCT (data independent)
  - SVD, ICA (data independent)
  - MDS, FastMap
- Linear forecasting
- Streaming pattern discovery
- Automatic mining

Kumamoto U CMU CS

SVD

- Singular Value Decomposition
- THE optimal method for dimensionality reduction
  - (under the Euclidean metric)



- Given: many time sequences
- Find: the latent ('hidden') variables

Kumamoto U CMU CS

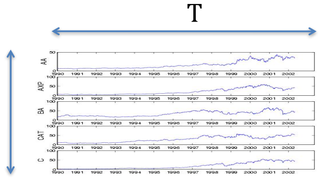
SVD

Two (equivalent) interpretations:
- Geometric (each sequence -> point in T-d space)
- Matrix algebra (N x T matrix)

eg, stock market data

T

N



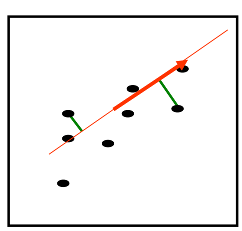
AA: Alcoa
AXP: American Express
BA: Boeing
CAT: Caterpillar
C: Citi Group

Kumamoto U CMU CS

Singular Value Decomposition (SVD)

- SVD (~LSI ~ KL ~ PCA ~ spectral analysis...) – Geometric interpretation

day2



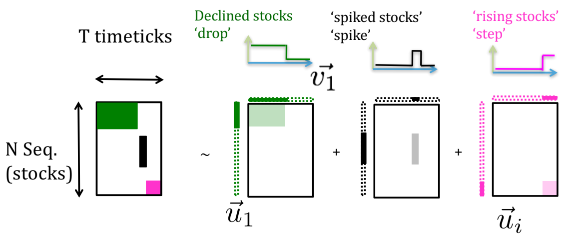
day1

LSI: S. Dumais; M. Berry
KL: eg, Duda+Hart
PCA: eg., Jolliffe
Details: [Press+], [Faloutsos96]

Kumamoto U CMU CS

SVD – matrix interpretation

- SVD -> matrix factorization: finds blocks



Kumamoto U CMU CS

SVD

- **<u>Extremely</u>** useful tool
  - (also behind PageRank/google and Kleinberg's algorithm for hubs and authorities)

Kumamoto U CMU CS

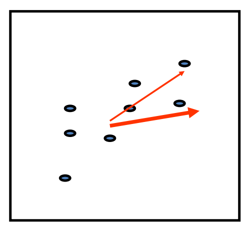
SVD

- **<u>Extremely</u>** useful tool
  - (also behind PageRank/google and Kleinberg's algorithm for hubs and authorities)
- But may be slow: O($N * M * M$) if $N>M$
- any approximate, faster method?

Kumamoto U CMU CS

SVD shorcuts

- random projections (Johnson-Lindenstrauss thm [Papadimitriou+ pods98])



Kumamoto U CMU CS

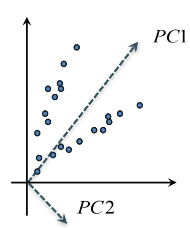
Random projections

- pick 'enough' random directions (will be ~orthogonal, in high-d!!)
- distances are preserved probabilistically, within epsilon
- (also, use as a pre-processing step for SVD [Papadimitriou+ PODS98])

Kumamoto U CMU CS

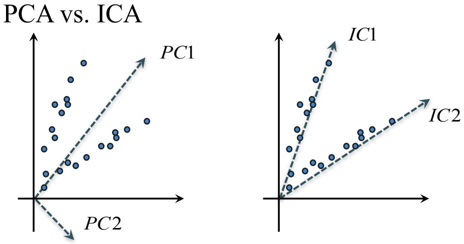
SVD & improvement

- Q: Can we do even better?
- A: sometimes, yes – by shooting for sparsity



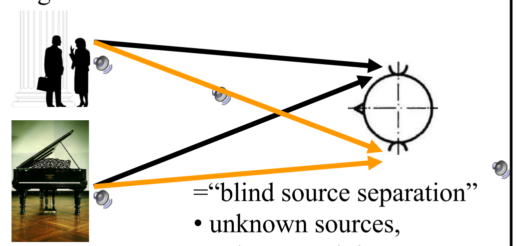
Independent Component Analysis (ICA)

- PCA (or SVD) sometimes misses essential features
  - PCA vs. ICA



A.k.a.: BSS = cocktail party problem Find hidden variables

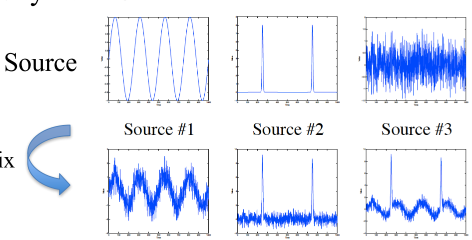
- Untangle two sound sources



="blind source separation"
- unknown sources,
- unknown mixing

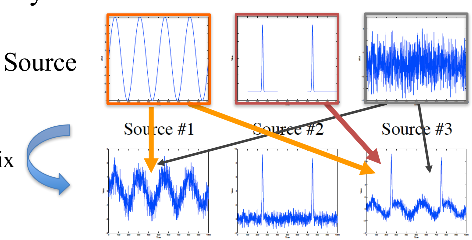
ICA

- Why not PCA



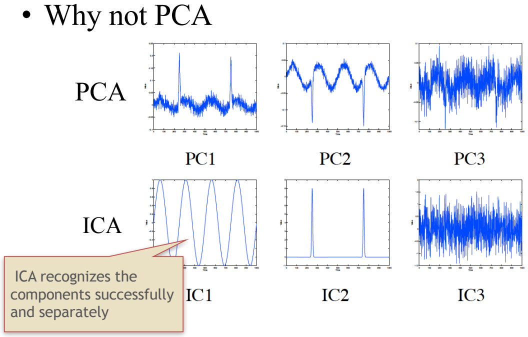
ICA

- Why not PCA



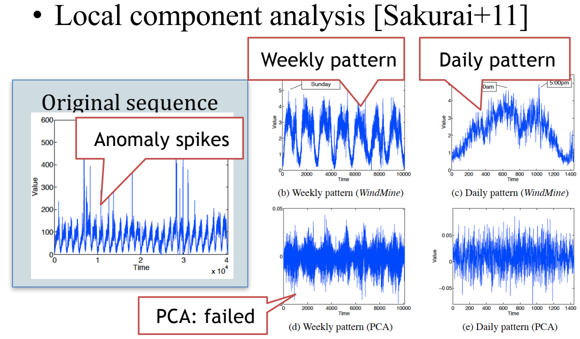
ICA

- Why not PCA



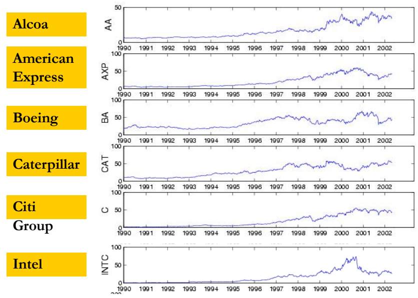
Hidden variables

- Local component analysis [Sakurai+11]



Kumamoto U CMU CS

Hidden variables

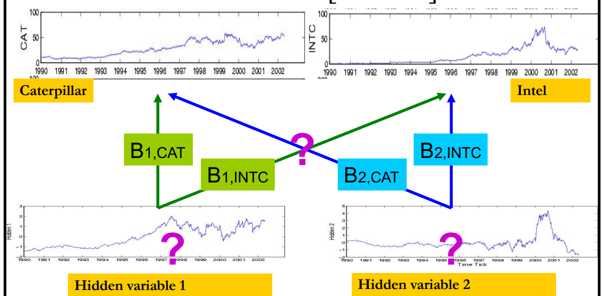


http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/ © 2017 Sakurai, Matsubara & Faloutsos 115

Kumamoto U CMU CS

Hidden variables

- Find hidden variables [Pan+04]

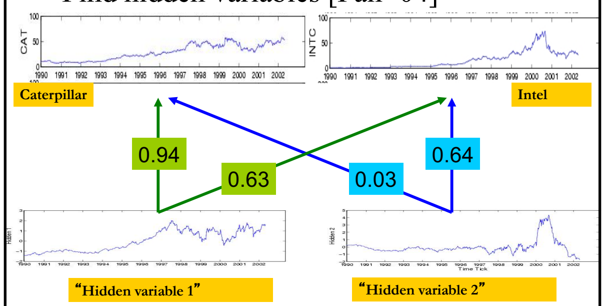


http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/ © 2017 Sakurai, Matsubara & Faloutsos 116

Kumamoto U CMU CS

Hidden variables

- Find hidden variables [Pan+04]

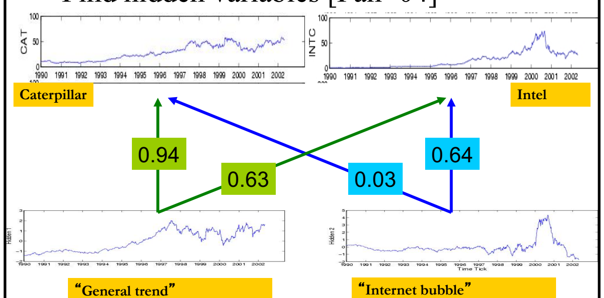


http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/ © 2017 Sakurai, Matsubara & Faloutsos 117

Kumamoto U CMU CS

Hidden variables

- Find hidden variables [Pan+04]



http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/ © 2017 Sakurai, Matsubara & Faloutsos 118

Kumamoto U CMU CS

Motivation: Find hidden variables

- ICA: also known as 'Blind Source Separation'
- 'cocktail party problem'
  - in a party, we can hear two concurrent conversations,
  - but separate them (and tune-in on one of them only)
- http://www.cnl.salk.edu/~tewon/Blind/blind_audio.html
- (in stocks: one 'discussion' is the general economy trend; the other 'discussion' is the tech-stock boom

http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/ © 2017 Sakurai, Matsubara & Faloutsos 119

Kumamoto U CMU CS

Citation

- *AutoSplit: Fast and Scalable Discovery of Hidden Variables in Stream and Multimedia Databases,* Jia-Yu Pan, Hiroyuki Kitagawa, Christos Faloutsos and Masafumi Hamamoto, PAKDD 2004, Sydney, Australia.
- *WindMine: Fast and Effective Mining of Web-click Sequences*, Yasushi Sakurai, Lei Li, Yasuko Matsubara, Christos Faloutsos, SDM 2011, Mesa, Arizona.

http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/ © 2017 Sakurai, Matsubara & Faloutsos 120

Kumamoto U — CMU CS

Roadmap

- Motivation
- Similarity Search and Indexing
- Feature extraction
  - DFT, DWT, DCT (data independent)
  - SVD, ICA (data independent)
  - MDS, FastMap
- Linear forecasting
- Streaming pattern discovery
- Automatic mining

Kumamoto U — CMU CS

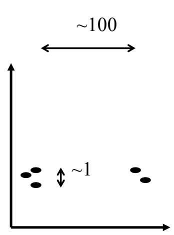
MDS / FastMap

- but, what if we have NO points to start with? (eg. Time-warping distance)
- A: Multi-dimensional Scaling (MDS) ; FastMap

Kumamoto U — CMU CS

MDS/FastMap

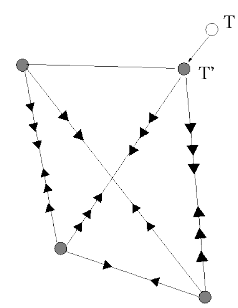
	O1	O2	O3	O4	O5
O1	0	1	1	100	100
O2	1	0	1	100	100
O3	1	1	0	100	100
O4	100	100	100	0	1
O5	100	100	100	1	0



Kumamoto U — CMU CS

MDS

Multi Dimensional Scaling



Kumamoto U — CMU CS

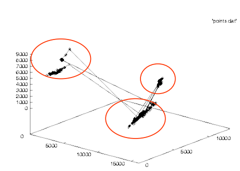
FastMap

- Multi-dimensional scaling (MDS) can do that, but in O(N**2) time
- FastMap [Faloutsos+95] takes O(N) time

Kumamoto U — CMU CS

FastMap: Application

VideoTrails [Kobla+97]

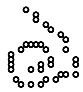


scene-cut detection (about 10% errors)

Kumamoto U CMU CS

Variations

- Isomap [Tenenbaum, de Silva, Langford, 2000]
- LLE (Local Linear Embedding) [Roweis, Saul, 2000]
- MVE (Minimum Volume Embedding) [Shaw & Jebara, 2007]



http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/ © 2017 Sakurai, Matsubara & Faloutsos 127

Kumamoto U CMU CS

Conclusions - Practitioner's guide

Similarity search in time sequences

1) establish/choose distance (Euclidean, time-warping,…)

2) extract features (SVD, ICA, DWT), and use an SAM (R-tree/variant, or a Metric Tree M-tree)

2') for high intrinsic dimensionalities, consider sequential scan (it might win…)

http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/ © 2017 Sakurai, Matsubara & Faloutsos 128

Kumamoto U CMU CS

Books

- William H. Press, Saul A. Teukolsky, William T. Vetterling and Brian P. Flannery: *Numerical Recipes in C*, Cambridge University Press, 1992, 2nd Edition. (Great description, intuition and code for SVD)
- C. Faloutsos: *Searching Multimedia Databases by Content*, Kluwer Academic Press, 1996 (introduction to SVD, and GEMINI)

http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/ © 2017 Sakurai, Matsubara & Faloutsos 129

Kumamoto U CMU CS

References

- Agrawal, R., K.-I. Lin, et al. (Sept. 1995). Fast Similarity Search in the Presence of Noise, Scaling and Translation in Time-Series Databases. Proc. of VLDB, Zurich, Switzerland.
- Babu, S. and J. Widom (2001). "Continuous Queries over Data Streams." SIGMOD Record 30(3): 109-120.
- Breunig, M. M., H.-P. Kriegel, et al. (2000). LOF: Identifying Density-Based Local Outliers. SIGMOD Conference, Dallas, TX.
- Berry, Michael: http://www.cs.utk.edu/~lsi/

http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/ © 2017 Sakurai, Matsubara & Faloutsos 130

Kumamoto U CMU CS

References

- Ciaccia, P., M. Patella, et al. (1997). M-tree: An Efficient Access Method for Similarity Search in Metric Spaces. VLDB.
- Foltz, P. W. and S. T. Dumais (Dec. 1992). "Personalized Information Delivery: An Analysis of Information Filtering Methods." Comm. of ACM (CACM) 35(12): 51-60.
- Guttman, A. (June 1984). R-Trees: A Dynamic Index Structure for Spatial Searching. Proc. ACM SIGMOD, Boston, Mass.

http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/ © 2017 Sakurai, Matsubara & Faloutsos 131

Kumamoto U CMU CS

References

- Gaede, V. and O. Guenther (1998). "Multidimensional Access Methods." Computing Surveys 30(2): 170-231.
- Gehrke, J. E., F. Korn, et al. (May 2001). On Computing Correlated Aggregates Over Continual Data Streams. ACM Sigmod, Santa Barbara, California.

http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/ © 2017 Sakurai, Matsubara & Faloutsos 132

Kumamoto U

References

CMU CS

- Gunopulos, D. and G. Das (2001). Time Series Similarity Measures and Time Series Indexing. SIGMOD Conference, Santa Barbara, CA.
- Eamonn J. Keogh, Themis Palpanas, Victor B. Zordan, Dimitrios Gunopulos, Marc Cardle: Indexing Large Human-Motion Databases. VLDB 2004: 780-791

Kumamoto U

References

CMU CS

- Hatonen, K., M. Klemettinen, et al. (1996). Knowledge Discovery from Telecommunication Network Alarm Databases. ICDE, New Orleans, Louisiana.
- Jolliffe, I. T. (1986). Principal Component Analysis, Springer Verlag.

Kumamoto U

References

CMU CS

- Keogh, E. J., K. Chakrabarti, et al. (2001). Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases. SIGMOD Conference, Santa Barbara, CA.
- Kobla, V., D. S. Doermann, et al. (Nov. 1997). VideoTrails: Representing and Visualizing Structure in Video Sequences. ACM Multimedia 97, Seattle, WA.

Kumamoto U

References

CMU CS

- Oppenheim, I. J., A. Jain, et al. (March 2002). A MEMS Ultrasonic Transducer for Resident Monitoring of Steel Structures. SPIE Smart Structures Conference SS05, San Diego.
- Papadimitriou, C. H., P. Raghavan, et al. (1998). Latent Semantic Indexing: A Probabilistic Analysis. PODS, Seattle, WA.
- Rabiner, L. and B.-H. Juang (1993). Fundamentals of Speech Recognition, Prentice Hall.

Kumamoto U

References

CMU CS

- Traina, C., A. Traina, et al. (October 2000). Fast feature selection using the fractal dimension,. XV Brazilian Symposium on Databases (SBBD), Paraiba, Brazil.

Kumamoto U

References

CMU CS

- Dennis Shasha and Yunyue Zhu *High Performance Discovery in Time Series: Techniques and Case Studies* Springer 2004
- Yunyue Zhu, Dennis Shasha ``*StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time*'' VLDB, August, 2002. pp. 358-369.
- Samuel R. Madden, Michael J. Franklin, Joseph M. Hellerstein, and Wei Hong. *The Design of an Acquisitional Query Processor for Sensor Networks*. SIGMOD, June 2003, San Diego, CA.

Kumamoto U CMU CS

References

- Lawrence Saul & Sam Roweis. *An Introduction to Locally Linear Embedding* (draft)
- Sam Roweis & Lawrence Saul. *Nonlinear dimensionality reduction by locally linear embedding*. Science, v.290 no.5500 , Dec.22, 2000. pp.2323--2326.
- B. Shaw and T. Jebara. "*Minimum Volume Embedding*" . Artificial Intelligence and Statistics, AISTATS, March 2007.

Kumamoto U CMU CS

References

- Josh Tenenbaum, Vin de Silva and John Langford. *A Global Geometric Framework for Nonlinear dimensionality Reduction*. Science 290, pp. 2319-2323, 2000.

Kumamoto U CMU CS

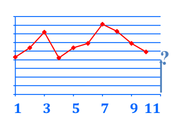
Roadmap

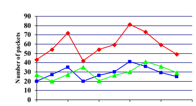
- Motivation
- Similarity Search and Indexing
- Feature extraction
- ➡ Linear forecasting
- Streaming pattern discovery
- Automatic mining

Kumamoto U CMU CS

Wish list

- Problem 1: find patterns/**rules**
- ➡ Problem 2: **forecast**
- Problem 3: find patterns/rules/forecast, with **many** time sequences



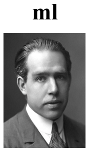


Kumamoto U CMU CS

Forecasting

"Prediction is very difficult, especially about the future." - Niels Bohr

http://www.hfac.uh.edu/MediaFutures/thoughts.html



Kumamoto U CMU CS

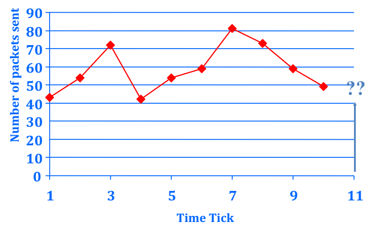
Roadmap

- Motivation
- Similarity Search and Indexing
- Feature extraction
- Linear forecasting
  - ➡ Auto-regression: Least Squares; RLS
  - Co-evolving time sequences
- Streaming pattern discovery
- Automatic mining

Kumamoto U — CMU CS

Problem: Forecasting

- Example: give x_{t-1}, x_{t-2}, …, forecast x_t



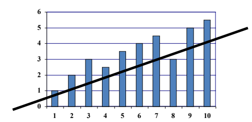
Kumamoto U — CMU CS

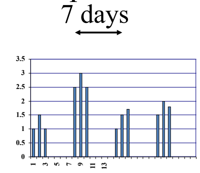
Forecasting: Preprocessing

MANUALLY:

remove trends

spot periodicities





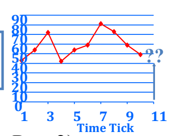
Kumamoto U — CMU CS

Problem: Forecast

- Solution: try to express
  x_t
  as a linear function of the past: x_{t-2}, x_{t-2}, …,
  (up to a window of w)

Formally:

$$\boxed{x_t \approx a_1 x_{t-1} + \ldots + a_w x_{t-w} + noise}$$

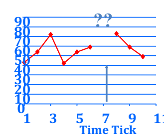


(if we **know** it is a non-linear model, see Part 2)

Kumamoto U — CMU CS

(Problem: Back-cast; interpolate)

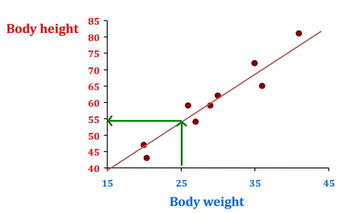
- Solution - interpolate: try to express
  x_t
  as a linear function of the past AND the future:
  $x_{t+1}, x_{t+2}, \ldots x_{t+w\,future};\ x_{t-1}, \ldots x_{t-w\,past}$
  (up to windows of w_{past} , w_{future})
- EXACTLY the same algo's



Kumamoto U — CMU CS

Background: Linear Regression

patient	weight	height
1	27	43
2	43	54
3	54	72
…	…	…
N	25	??



- express what we don't know (= 'dependent variable')
- as a linear function of what we know (= 'indep. variable(s)')

Kumamoto U — CMU CS

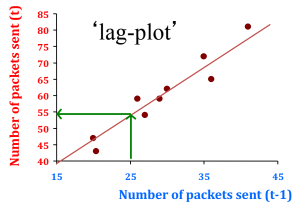
Linear Auto Regression:

Time	Packets Sent(t)
1	43
2	54
3	72
…	…
N	??

Kumamoto U CMU CS

Linear Auto Regression:

Time	Packets Sent (t-1)	Packets Sent(t)
1	-	43
2	43	54
3	54	72
…	…	…
N	25	??

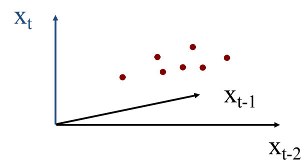


- lag w=1
- Dependent variable = # of packets sent (S [t])
- Independent variable = # of packets sent (S[t-1])

Kumamoto U CMU CS

More details:

- Q1: Can it work with window w>1?
- A1: YES!

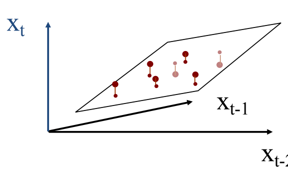


eg, w=2

Kumamoto U CMU CS

More details:

- Q1: Can it work with window w>1?
- A1: YES! (we'll fit a hyper-plane, then!)

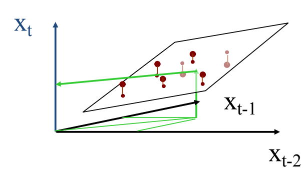


eg, w=2

Kumamoto U CMU CS

More details:

- Q1: Can it work with window w>1?
- A1: YES! (we'll fit a hyper-plane, then!)



eg, w=2

Kumamoto U

More details: DETAILS

- Q1: Can it work with window w>1?
- A1: YES! The problem becomes:

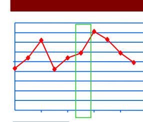
$$\mathbf{X}_{[N \times w]} \times \mathbf{a}_{[w \times 1]} = \mathbf{y}_{[N \times 1]}$$

- OVER-CONSTRAINED
  - **a** is the vector of the regression coefficients
  - **X** has the N values of the w indep. variables
  - **y** has the N values of the dependent variable

Kumamoto U

More details: DETAILS

- $\mathbf{X}_{[N \times w]} \times \mathbf{a}_{[w \times 1]} = \mathbf{y}_{[N \times 1]}$



Ind-var1 Ind-var-w

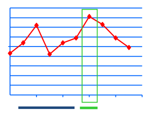
time

$$\begin{bmatrix} X_{11}, X_{12}, \cdots, X_{1w} \\ X_{21}, X_{22}, \ldots, X_{2w} \\ \vdots \\ \vdots \\ \vdots \\ X_{N1}, X_{N2}, \ldots, X_{Nw} \end{bmatrix} \times \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_w \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ \vdots \\ y_N \end{bmatrix}$$

Kumamoto U

More details: DETAILS

- $\mathbf{X}_{[N \times w]} \times \mathbf{a}_{[w \times 1]} = \mathbf{y}_{[N \times 1]}$



Ind-var1 Ind-var-w

$$\text{time} \downarrow \begin{bmatrix} X_{11}, X_{12}, \cdots, X_{1w} \\ X_{21}, X_{22}, \ldots, X_{2w} \\ \vdots \\ \vdots \\ \vdots \\ X_{N1}, X_{N2}, \ldots, X_{Nw} \end{bmatrix} \times \begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_w \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ \vdots \\ \vdots \\ y_N \end{bmatrix}$$

Kumamoto U

More details DETAILS

- Q2: How to estimate $a_1, a_2, \ldots a_w = \mathbf{a}$?
- A2: with Least Squares fit

$$\boxed{\mathbf{a} = (\mathbf{X}^T \times \mathbf{X})^{-1} \times (\mathbf{X}^T \times \mathbf{y})}$$

- (Moore-Penrose pseudo-inverse)
- **a** is the vector that minimizes the RMSE from **y**

Kumamoto U

Even more details DETAILS

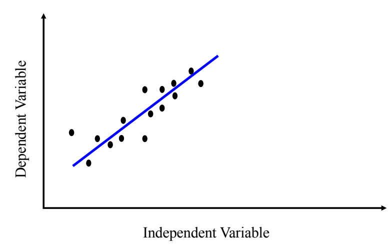
- Q3: Can we estimate **a** incrementally?
- A3: Yes, with the brilliant, classic method of 'Recursive Least Squares' (RLS) (see, e.g., [Yi+00], for details) - pictorially:

[Yi+00] Byoung-Kee Yi et al.: *Online Data Mining for Co-Evolving Time Sequences*, ICDE 2000.

Kumamoto U CMU CS

Even more details

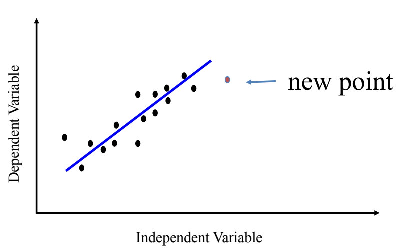
- Given:



Kumamoto U CMU CS

Even more details

- Given:

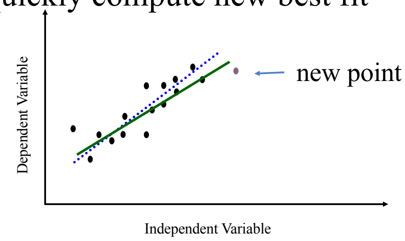


Kumamoto U CMU CS

Even more details

Recursive Least Squares (RLS):

quickly compute new best fit



Even more details

- Straightforward Least Squares
  - Needs huge matrix (**growing** in size) $O(N \times w)$
  - Costly matrix operation $O(N \times w^2)$
- Recursive LS
  - Need much smaller, fixed size matrix $O(w \times w)$
  - Fast, incremental computation $O(1 \times w^2)$

49,000,000 ⟷ 49

$N = 10^6$, $w = 1\text{-}100$

Even more details

- Straightforward Least Squares
  - Needs huge matrix (**growing** in size) $O(N \times w)$
  - Costly matrix operation $O(N \times w^2)$
- Recursive LS
  - Need much smaller, fixed size matrix $O(w \times w)$
  - Fast, incremental computation $O(1 \times w^2)$

49,000,000 ⟷ 49

$N = 10^6$, $w = 1\text{-}100$

RLS: GREAT for streams

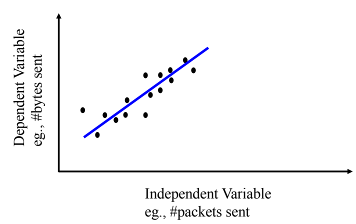
Even more details

DETAILS

- Q4: can we 'forget' the older samples?
- A4: Yes - RLS can easily handle that [Yi+00]:

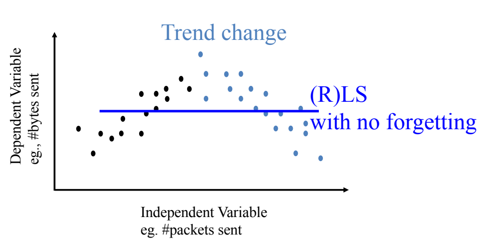
Adaptability - 'forgetting'

DETAILS



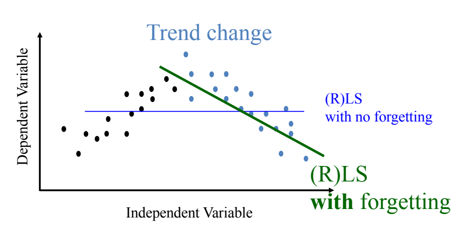
Adaptability - 'forgetting'

DETAILS



Adaptability - 'forgetting'

DETAILS



- RLS: can *trivially* handle 'forgetting'

How to choose 'w' ?

- Quick & dirty answer: w=1 or w=2
- Better answer: Model selection (say, with BIC or MDL – see later)
- Even better answer: **multi-scale windows** [Papadimitriou+, vldb2003]

Spiros Papadimitriou, Anthony Brockwell and Christos Faloutsos *Adaptive, Hands-Off Stream Mining* VLDB 2003, Berlin, Germany, Sept. 2003

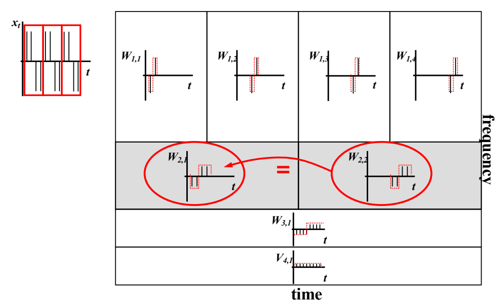
How to choose 'w' ?

- goal: capture arbitrary periodicities
- with NO human intervention
- on a semi-infinite stream

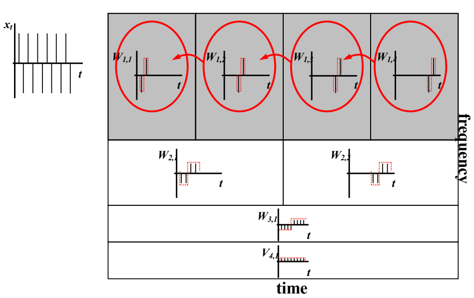
Answer:

- 'AWSOM' (Arbitrary Window Stream fOrecasting Method) [Papadimitriou+, vldb2003]
- idea: do AR on each wavelet level
- in detail:

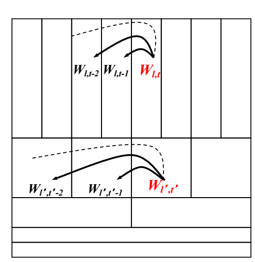
AWSOM



AWSOM



AWSOM - idea



$W_{l,t} = \beta_{l,1}W_{l,t-1} + \beta_{l,2}W_{l,t-2} + \ldots$

$W_{l',t'} = \beta_{l',1}W_{l',t'-1} + \beta_{l',2}W_{l',t'-2} + \ldots$

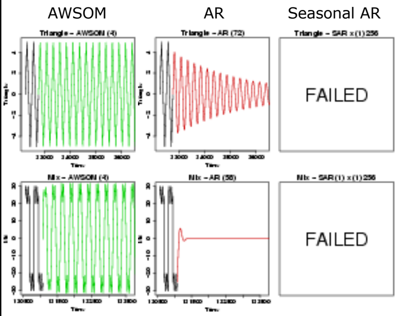
Kumamoto U CMU CS

More details…

- Update of wavelet coefficients (incremental)
- Update of linear models (incremental; RLS)
- Feature selection (single-pass)
  - Not all correlations are significant
  - Throw away the insignificant ones ("noise")

Kumamoto U CMU CS

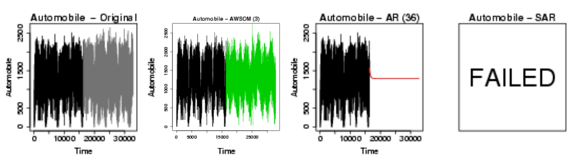
Results - Synthetic data



- Triangle pulse
- Mix (sine + square)
- AR captures wrong trend (or none)
- Seasonal AR estimation fails

Kumamoto U CMU CS

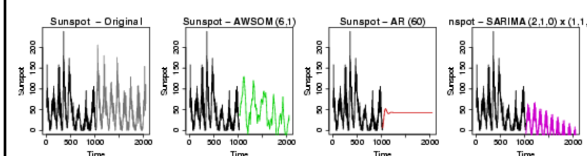
Results - Real data



- Automobile traffic
  - Daily periodicity
  - Bursty "noise" at smaller scales
- AR fails to capture any trend
- Seasonal AR estimation fails

Kumamoto U CMU CS

Results - real data



- Sunspot intensity
  - Slightly time-varying "period"
- AR captures wrong trend
- Seasonal ARIMA
  - wrong downward trend, despite help by human!

Kumamoto U CMU CS

Complexity

Skip

- Model update

  Space: $O(lgN + mk^2) \approx O(lgN)$

  Time: $O(k^2) \approx O(1)$
- Where
  - N: number of points (so far)
  - k: number of regression coefficients; fixed
  - m: number of linear models; $O(lgN)$

Kumamoto U CMU CS

Roadmap

- Motivation
- Similarity Search and Indexing
- Feature extraction
- Streaming pattern discovery
- Linear forecasting
  - Auto-regression: Least Squares; RLS
  - Co-evolving time sequences
- Automatic mining

Kumamoto U CMU CS

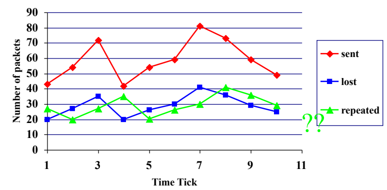
Roadmap

- Motivation
- Similarity Search and Indexing
- Feature extraction
- Linear forecasting
  - Auto-regression: Least Squares; RLS
  - Co-evolving time sequences
- Streaming pattern discovery
- Automatic mining

Kumamoto U CMU CS

Co-Evolving Time Sequences

- Given: A set of **correlated** time sequences
- Forecast '**Repeated(t)**'



Kumamoto U CMU CS

Solution:

Q: what should we do?

A: Least Squares, with

- Dep. Variable: Repeated(t)
- Indep. Variables:
  - Sent(t-1), …, Sent(t-w);
  - Lost(t-1), …, Lost(t-w);
  - Repeated(t-1), ...
- (named: 'MUSCLES' [Yi+00])

Kumamoto U CMU CS

Practitioner's guide

- AR(IMA) methodology: prevailing method for linear forecasting
- Brilliant method of Recursive Least Squares for fast, incremental estimation.
- See [Box-Jenkins]

Kumamoto U CMU CS

Resources: software and urls

- MUSCLES: Prof. Byoung-Kee Yi: **http://www.postech.ac.kr/~bkyi/** or **christos@cs.cmu.edu**
- free-ware: 'R' for stat. analysis (clone of Splus) **http://cran.r-project.org/**

Kumamoto U CMU CS

Books

- George E.P. Box and Gwilym M. Jenkins and Gregory C. Reinsel, *Time Series Analysis: Forecasting and Control*, Prentice Hall, 1994 (the classic book on ARIMA, 3rd ed.)
- Brockwell, P. J. and R. A. Davis (1987). Time Series: Theory and Methods. New York, Springer Verlag.

Additional Reading

- [Papadimitriou+ vldb2003] Spiros Papadimitriou, Anthony Brockwell and Christos Faloutsos *Adaptive, Hands-Off Stream Mining* VLDB 2003, Berlin, Germany, Sept. 2003
- [Yi+00] Byoung-Kee Yi et al.: *Online Data Mining for Co-Evolving Time Sequences*, ICDE 2000. (Describes MUSCLES and Recursive Least Squares)

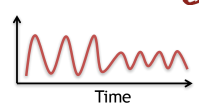
Outline

- Motivation
- Similarity Search and Indexing
- Feature extraction
- Linear forecasting
- Streaming pattern discovery
- Automatic mining

Stream mining

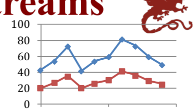
- Applications
  - Sensor monitoring
  - Network analysis
  - Financial and/or business transaction data
  - Web access and media service logs
  - Moving object tracking
  - Industrial manufacturing

Stream mining



- Requirements
  - **Fast**
    high performance and quick response
  - **Nimble**
    low memory consumption, single scan
  - **Accurate**
    good approximation for pattern discovery and feature extraction

Monitoring data streams



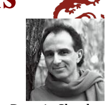
- Correlation coefficient

$$\rho = \frac{\sum_{t=1}^{n}(x_t - \bar{x})\cdot(y_t - \bar{y})}{\sigma(x)\cdot\sigma(y)} \qquad \sigma(x) = \sqrt{\sum_{t=1}^{n}(x_t - \bar{x})^2}$$

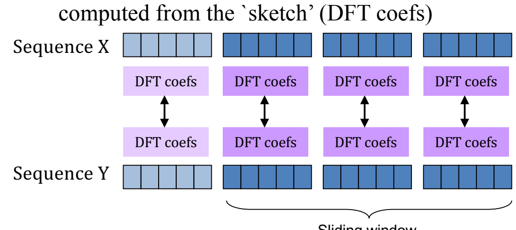
- Correlation coefficient and the (Euclidean) distance

$$\rho = 1 - \frac{1}{2}\sum_{t=1}^{n}(\hat{x}_t - \hat{y}_t)^2 \qquad \hat{x}_t = (x_t - \bar{x})/\sigma(x)$$

Monitoring data streams

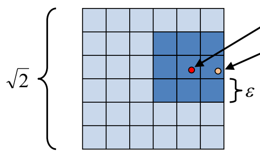

Dennis Shasha

- Correlation monitoring [Zhu+, vldb02]
  - DFT coefficients for each basic window
  - Correlation coefficient of each sliding window computed from the `sketch' (DFT coefs)



Monitoring data streams

- Grid structure (to avoid checking all pairs)
  - DFT coefficients yields a vector
  - High correlation -> closeness in the vector space

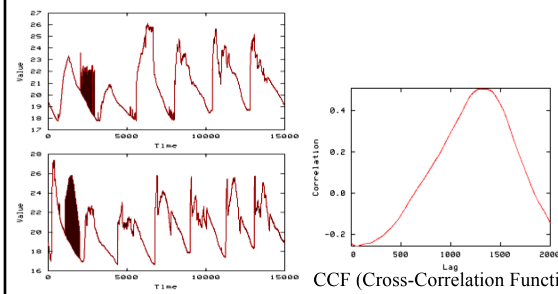


Correlation coefficients and the Euclidean distance

$$\rho = 1 - \frac{1}{2}\sum_{t=1}^{n}(\hat{x}_t - \hat{y}_t)^2$$

Monitoring data streams

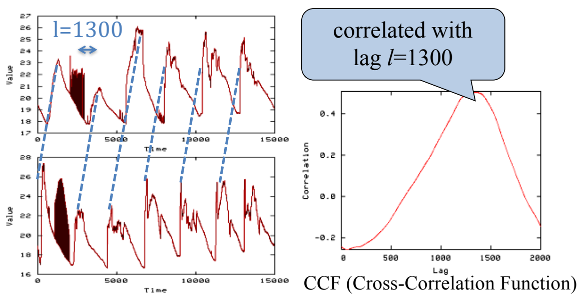
- Lag correlation [Sakurai+, sigmod05]



CCF (Cross-Correlation Function)

Monitoring data streams

- Lag correlation [Sakurai+, sigmod05]



CCF (Cross-Correlation Function)

Lag correlation

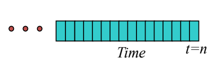
- Definition of 'score', absolute value of $R(l)$

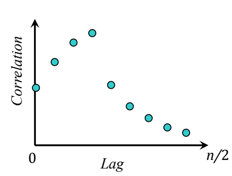
$$score(l) = |R(l)| \qquad R(l) = \frac{\sum_{t=l+1}^{n}(x_t - \bar{x})(y_{t-l} - \bar{y})}{\sqrt{\sum_{t=l+1}^{n}(x_t - \bar{x})^2}\sqrt{\sum_{t=1}^{n-l}(y_t - \bar{y})^2}}$$

- Lag correlation
  - Given a threshold γ, $score(l) > \gamma$
  - A local maximum
  - The earliest such maximum, if more maxima exist

Lag correlation

- Why not naïve?
  - Compute correlation coefficient for each lag
    l = {0, 1, 2, 3, …, n/2}
- But
  - $O(n)$ space
  - $O(n^2)$ time
  - or $O(n \log n)$ time w/ FFT

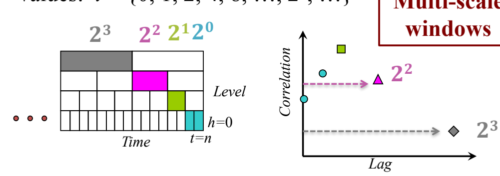




Lag correlation

- BRAID
  - Geometric lag probing + smoothing
  - Use colored windows
  - Keep track of only a geometric progression of the lag values: l = {0, 1, 2, 4, 8, …, 2^h, …}

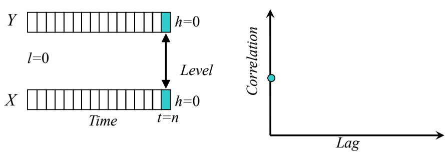
Multi-scale windows



Kumamoto U CMU CS

Lag correlation

- BRAID
  - Geometric lag probing + smoothing
  - Keep track of only a geometric progression of the lag values: $l = \{0, 1, 2, 4, 8, \ldots, 2^h, \ldots\}$

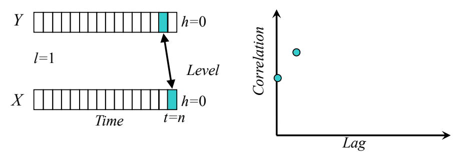


http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/ © 2017 Sakurai, Matsubara & Faloutsos 199

Kumamoto U CMU CS

Lag correlation

- BRAID
  - Geometric lag probing + smoothing
  - Keep track of only a geometric progression of the lag values: $l = \{0, 1, 2, 4, 8, \ldots, 2^h, \ldots\}$

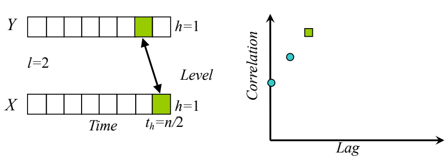


http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/ © 2017 Sakurai, Matsubara & Faloutsos 200

Kumamoto U CMU CS

Lag correlation

- BRAID
  - Geometric lag probing + smoothing
  - Keep track of only a geometric progression of the lag values: $l = \{0, 1, 2, 4, 8, \ldots, 2^h, \ldots\}$

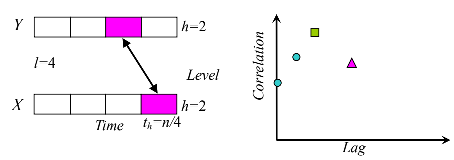


http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/ © 2017 Sakurai, Matsubara & Faloutsos 201

Kumamoto U CMU CS

Lag correlation

- BRAID
  - Geometric lag probing + smoothing
  - Keep track of only a geometric progression of the lag values: $l = \{0, 1, 2, 4, 8, \ldots, 2^h, \ldots\}$

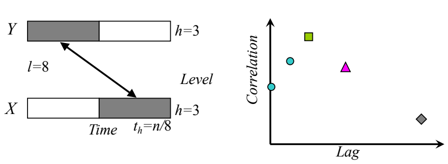


http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/ © 2017 Sakurai, Matsubara & Faloutsos 202

Kumamoto U CMU CS

Lag correlation

- BRAID
  - Geometric lag probing + smoothing
  - Keep track of only a geometric progression of the lag values: $l = \{0, 1, 2, 4, 8, \ldots, 2^h, \ldots\}$

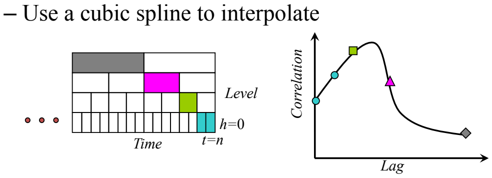


http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/ © 2017 Sakurai, Matsubara & Faloutsos 203

Kumamoto U CMU CS

Lag correlation

- BRAID
  - Geometric lag probing + smoothing
  - Keep track of only a geometric progression of the lag values: $l = \{0, 1, 2, 4, 8, \ldots, 2^h, \ldots\}$
  - Use a cubic spline to interpolate

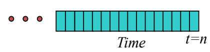


http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/ © 2017 Sakurai, Matsubara & Faloutsos 204

Kumamoto U — CMU CS

Lag correlation

- Why not naïve?
  - Compute correlation coefficient for each lag $l = \{0, 1, 2, 3, \ldots, n/2\}$
- But
  - $O(n)$ space
  - $O(n^2)$ time
  - or $O(n \log n)$ time w/ F



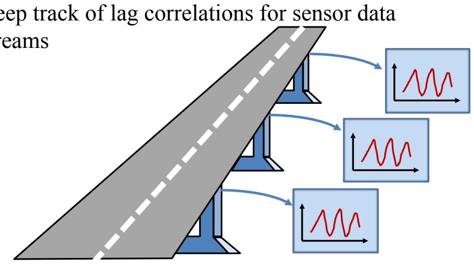
BRAID
- $O(\log n)$ space
- $O(1)$ time

Multi-scale windows

Kumamoto U — CMU CS

BRAID in the real world

- Bridge structural health monitoring
  - Structural monitoring using vibration/shock sensors
  - Keep track of lag correlations for sensor data streams



Kumamoto U — CMU CS

BRAID in the real world

- Bridge structural health monitoring
  - Goal: real-time anomaly detection for disaster prevention
  - Several thousands readings (per sec) from several hundreds sensor nodes

Structural health monitoring

Vibration/shock sensor

- Uses BRAID
- Metropolitan Expressway (Tokyo, Japan)

Kumamoto U — CMU CS

BRAID in the real world

- Bridge structural health monitoring with BRAID

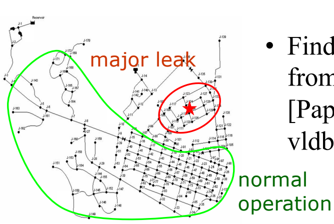
Metropolitan Expressway (Tokyo, Japan)

Tokyo Gate Bridge (Tokyo, Japan)

Can Tho Bridge (Vietnam)

Kumamoto U — CMU CS

Feature extraction from streams



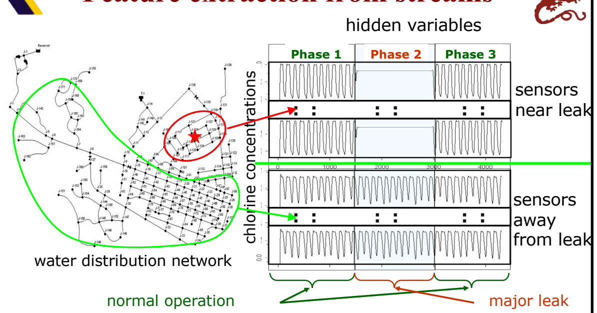
water distribution network

- Find hidden variables from streams [Papadimitriou+, vldb2005]

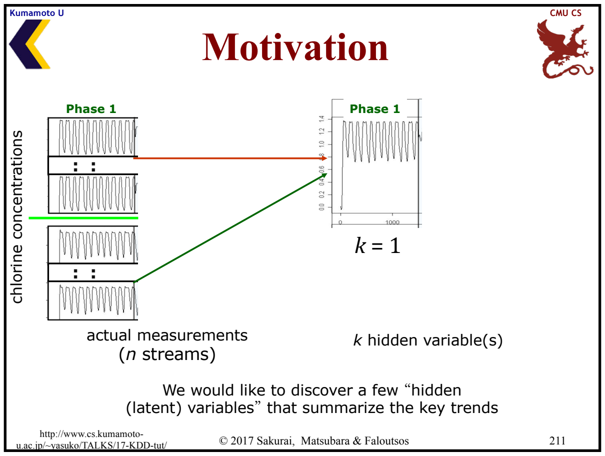
May have hundreds of measurements, but it is **unlikely they are completely unrelated**!

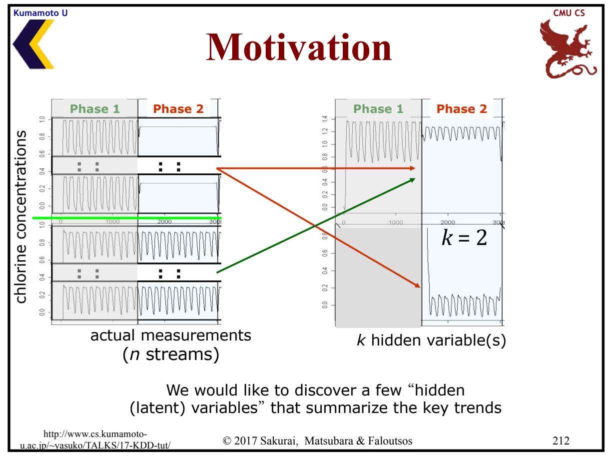
Kumamoto U — CMU CS

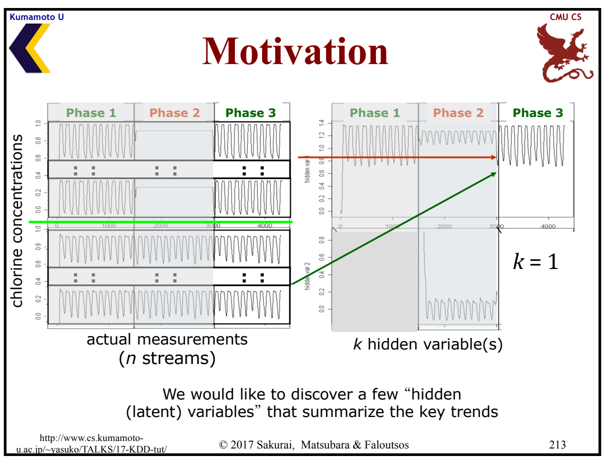
Feature extraction from streams

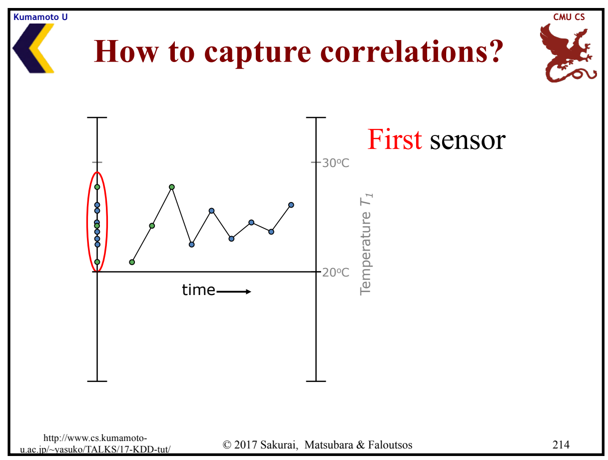


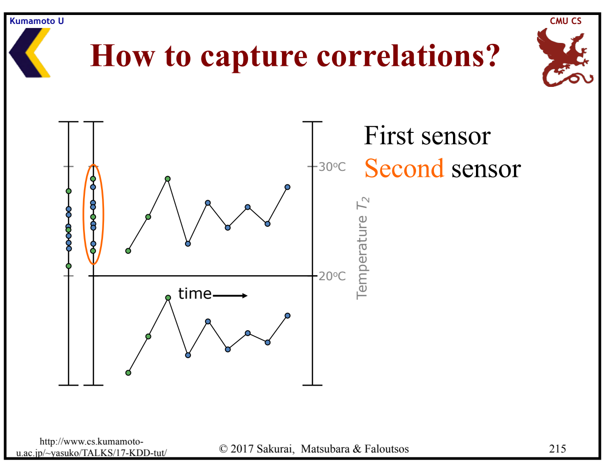
May have hundreds of measurements, but it is **unlikely they are completely unrelated**!

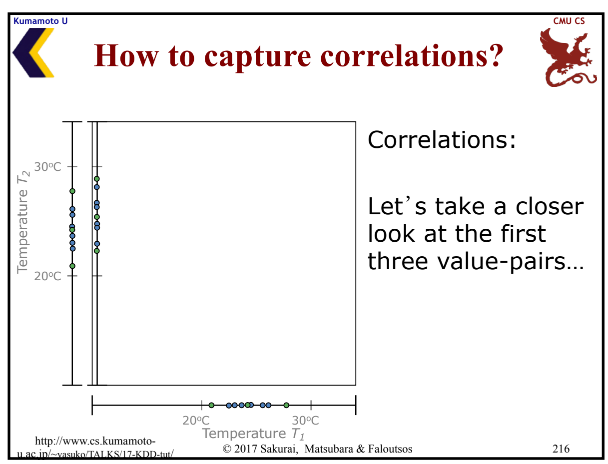

Kumamoto U
CMU CS
Motivation
Phase 1
Phase 1
chlorine concentrations
k = 1
actual measurements
(n streams)
k hidden variable(s)
We would like to discover a few "hidden
(latent) variables" that summarize the key trends
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
211


Kumamoto U
CMU CS
Motivation
Phase 1
Phase 2
Phase 1
Phase 2
chlorine concentrations
k = 2
actual measurements
(n streams)
k hidden variable(s)
We would like to discover a few "hidden
(latent) variables" that summarize the key trends
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
212

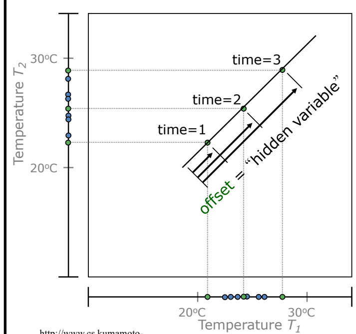

Kumamoto U
CMU CS
Motivation
Phase 1
Phase 2
Phase 3
Phase 1
Phase 2
Phase 3
chlorine concentrations
k = 1
actual measurements
(n streams)
k hidden variable(s)
We would like to discover a few "hidden
(latent) variables" that summarize the key trends
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
213


Kumamoto U
CMU CS
How to capture correlations?
First sensor
30°C
20°C
time
Temperature T1
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
214


Kumamoto U
CMU CS
How to capture correlations?
First sensor
Second sensor
30°C
20°C
time
Temperature T2
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
215


Kumamoto U
CMU CS
How to capture correlations?
Correlations:
Let's take a closer
look at the first
three value-pairs…
Temperature T2
30°C
20°C
20°C
30°C
Temperature T1
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
216

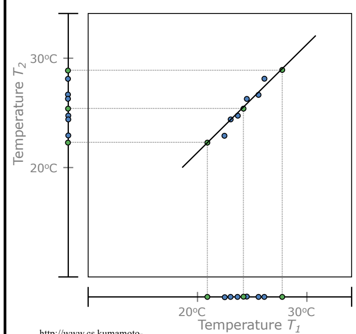
How to capture correlations?



First three lie (almost) on a line in the space of value-pairs…

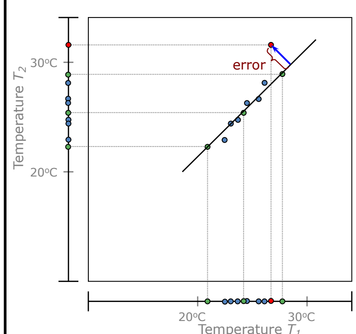
- O(n) numbers for the slope, and
- *One* number for each value-pair (offset on line)

How to capture correlations?



Other pairs also follow the same pattern: they lie (approximately) on this line

Incremental update

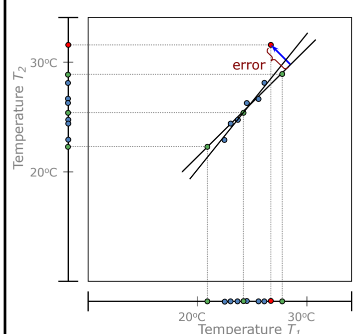


For each new point

- Project onto current line
- Estimate error

New value

Incremental update

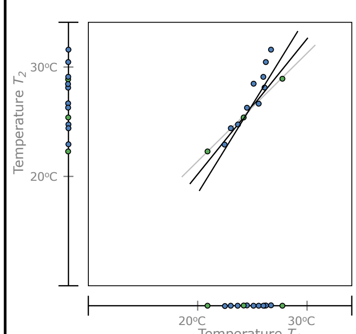


For each new point

- Project onto current line
- Estimate error
- Rotate line in the direction of the error and in proportion to its magnitude

➔O(n) time

Incremental update



For each new point

- Project onto current line
- Estimate error
- Rotate line in the direction of the error and in proportion to its magnitude

Related work

- Wavelet over streams [Gilbert+,vldb01] [Guha+,vldb04]
- Fourier representations [Gilbert+, stoc02]
- KNN [Koudas+, 04] [Korn+, vldb02]
- Histograms [Guha+, stoc01]
- Clustering [Guha+, focs00] [Aggarwal+, vldb03]
- Sketches [Indyk+, vldb00] [Cormode+, J. Algorithms 05]
- …
- …

Related work

- Heavy hitters [Cormode+, vldb03]
- Data embedding [Indyk+, focs00]
- Burst detection [Zhu+, kdd03]
- Segmentation [Keogh+, icdm01]
- Multiple scale analysis [Papadimitriou+, sigmod06]
- Fractal [Korn+, sigmod06]
- Time warping [Sakurai+, icde07]…
- …

Outline

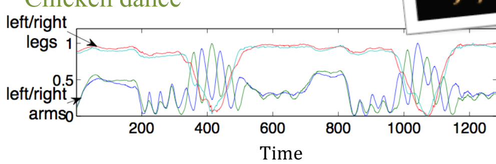
- Motivation
- Similarity Search and Indexing
- Feature extraction
- Streaming pattern discovery
- Linear forecasting
- Automatic mining

Motivation

Given: co-evolving time-series

– e.g., MoCap (leg/arm sensors)

“Chicken dance”

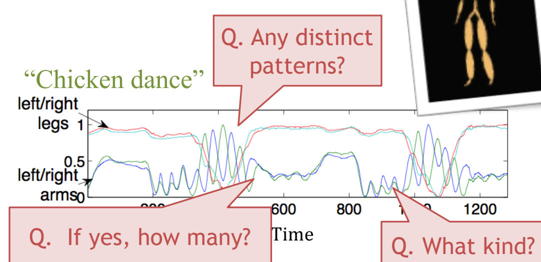


Motivation

Given: co-evolving time-series

– e.g., MoCap (leg/arm sensors)

“Chicken dance”

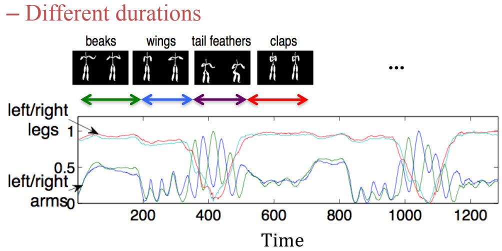


Motivation

Challenges: co-evolving sequences

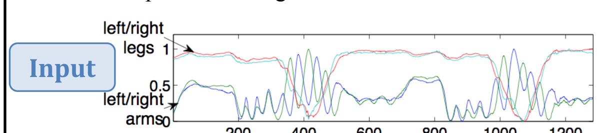
– Unknown # of patterns (e.g., beaks)

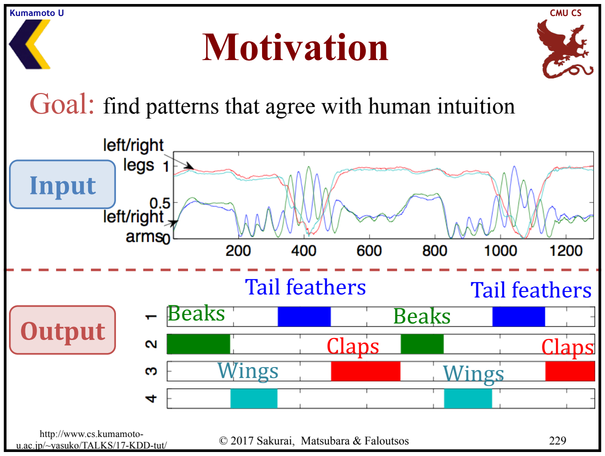
– Different durations

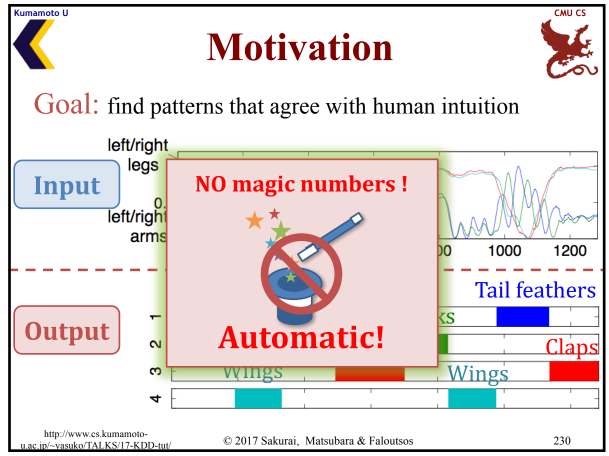


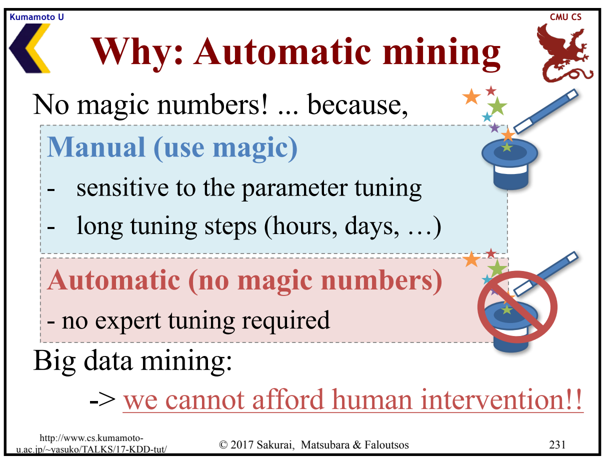
Motivation

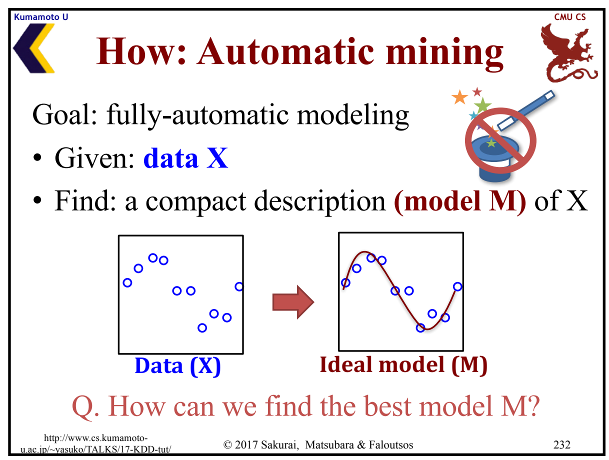
Goal: find patterns that agree with human intuition

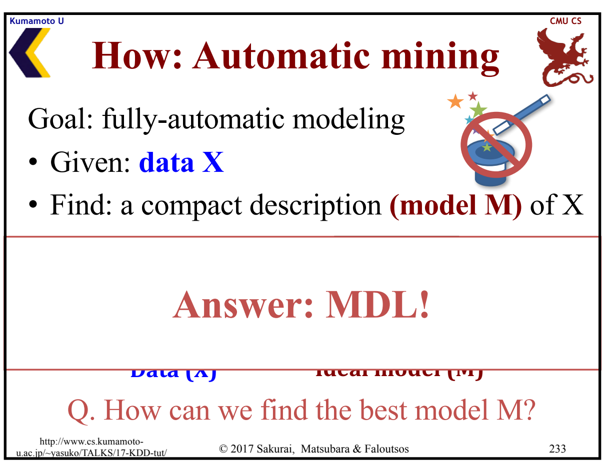


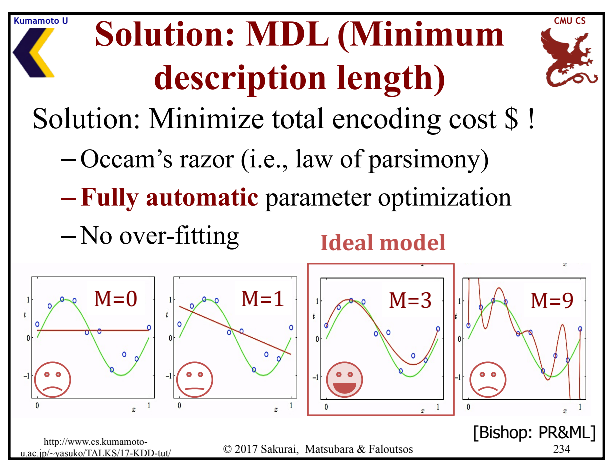

Kumamoto U
CMU CS
Motivation
Goal: find patterns that agree with human intuition
Input
left/right legs
left/right arms
1
0.5
0
200
400
600
800
1000
1200
Output
Tail feathers
Tail feathers
Beaks
Beaks
Claps
Claps
Wings
Wings
1
2
3
4
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
229

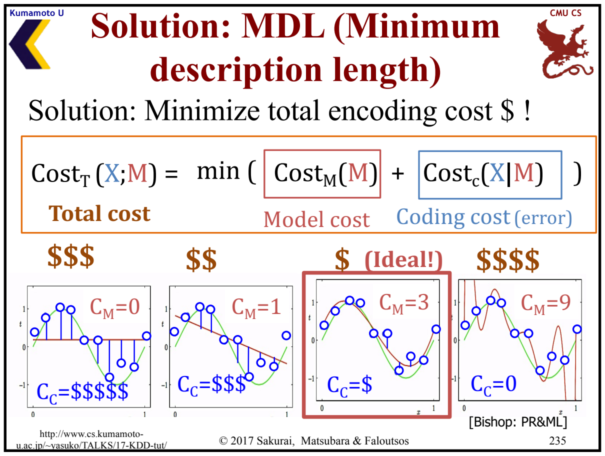

Kumamoto U
CMU CS
Motivation
Goal: find patterns that agree with human intuition
Input
left/right legs
left/right arms
NO magic numbers !
Automatic!
1000
1200
Output
Tail feathers
Claps
Wings
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
230

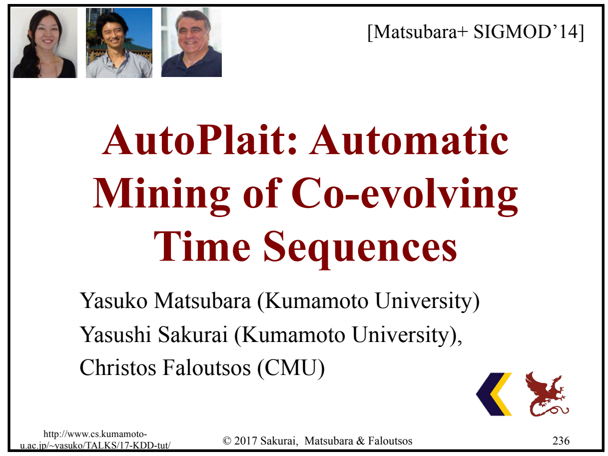

Kumamoto U
CMU CS
Why: Automatic mining
No magic numbers! ... because,
Manual (use magic)
- sensitive to the parameter tuning
- long tuning steps (hours, days, …)
Automatic (no magic numbers)
- no expert tuning required
Big data mining:
-> we cannot afford human intervention!!
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
231

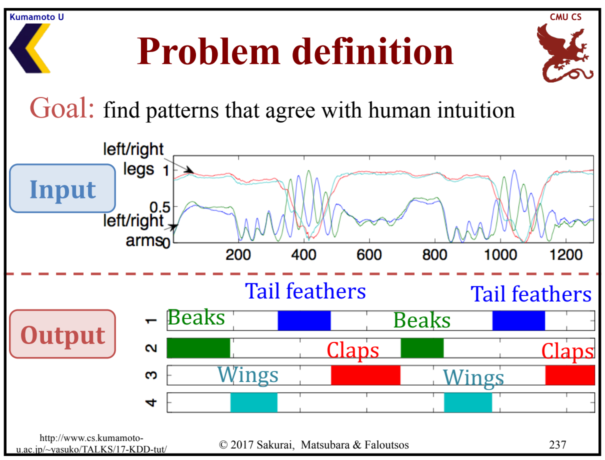

Kumamoto U
CMU CS
How: Automatic mining
Goal: fully-automatic modeling
• Given: data X
• Find: a compact description (model M) of X
Data (X)
Ideal model (M)
Q. How can we find the best model M?
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
232

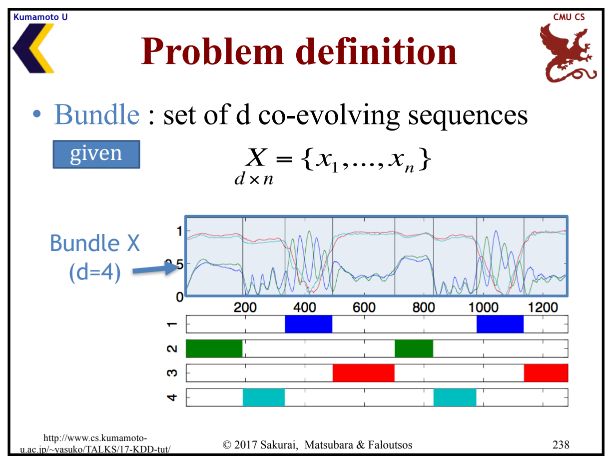

Kumamoto U
CMU CS
How: Automatic mining
Goal: fully-automatic modeling
• Given: data X
• Find: a compact description (model M) of X
Answer: MDL!
Q. How can we find the best model M?
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
233

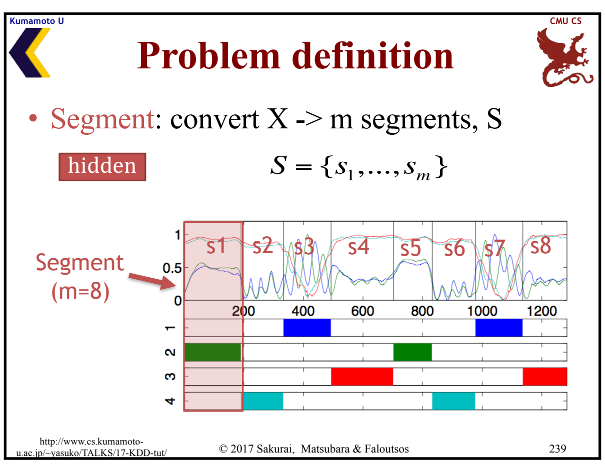

Kumamoto U
CMU CS
Solution: MDL (Minimum description length)
Solution: Minimize total encoding cost $!
– Occam's razor (i.e., law of parsimony)
– Fully automatic parameter optimization
– No over-fitting
Ideal model
M=0
M=1
M=3
M=9
[Bishop: PR&ML]
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
234

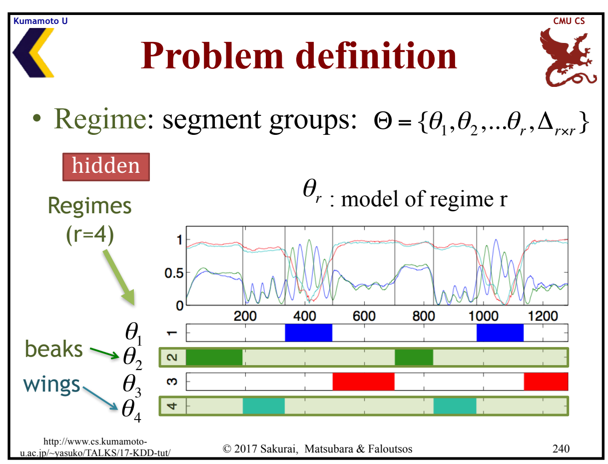

Solution: MDL (Minimum description length)
Solution: Minimize total encoding cost $!
Cost_T (X;M) = min (Cost_M(M) + Cost_c(X|M))
Total cost
Model cost
Coding cost (error)
$$$
$$
$ (Ideal!)
$$$$
C_M=0
C_C=$$$$$
C_M=1
C_C=$$$
C_M=3
C_C=$
C_M=9
C_C=0
[Bishop: PR&ML]
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
235


[Matsubara+ SIGMOD'14]
AutoPlait: Automatic Mining of Co-evolving Time Sequences
Yasuko Matsubara (Kumamoto University)
Yasushi Sakurai (Kumamoto University),
Christos Faloutsos (CMU)
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
236


Problem definition
Goal: find patterns that agree with human intuition
Input
left/right legs
left/right arms
Output
Tail feathers
Beaks
Claps
Wings
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
237


Problem definition
• Bundle : set of d co-evolving sequences
given
X = {x_1,...,x_n}
d×n
Bundle X (d=4)
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
238


Problem definition
• Segment: convert X -> m segments, S
hidden
S = {s_1,...,s_m}
Segment (m=8)
s1 s2 s3 s4 s5 s6 s7 s8
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
239


Problem definition
• Regime: segment groups: Θ = {θ_1, θ_2, ...θ_r, Δ_rxr}
hidden
θ_r : model of regime r
Regimes (r=4)
beaks
wings
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
240

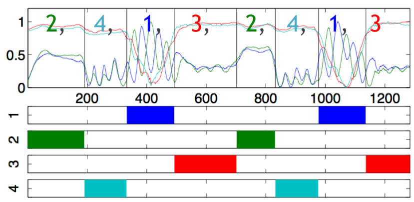
Kumamoto U | CMU CS

Problem definition

- Segment-membership: assignment

hidden $F = \{f_1, \ldots, f_m\}$

F = {



}

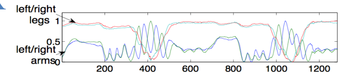
Segment-membership (m=8)

Kumamoto U | CMU CS

Problem definition

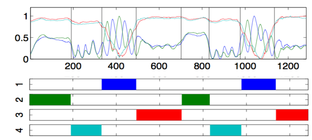
- Given: bundle X

$X = \{x_1, \ldots, x_n\}$



- Find: compact description C of X

$C = \{m, r, S, \Theta, F\}$

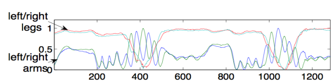


Kumamoto U | CMU CS

Problem definition

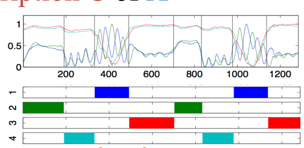
- Given: bundle X

$X = \{x_1, \ldots, x_n\}$



- Find: compact description C of X

$C = \{m, r, \boxed{S}, \boxed{\Theta}, \boxed{F}\}$



m segments

r regimes

Segment-membership

Kumamoto U | CMU CS

Main ideas

Goal: compact description of X

$C = \{m, r, S, \Theta, F\}$

without user intervention!!

Challenges:

Q1. How to generate 'informative' regimes ?

Q2. How to decide # of regimes/segments ?

Kumamoto U | CMU CS

Main ideas

Goal: compact description of X

$C = \{m, r, S, \Theta, F\}$

without user intervention!!

Challenges:

Q1. How to generate 'informative' regimes ?

Idea (1): Multi-level chain model

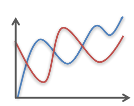
Q2. How to decide # of regimes/segments ?

Idea (2): Model description cost

Kumamoto U | CMU CS

Idea (1): MLCM: multi-level chain model

Q1. How to generate 'informative' regimes ?



Sequences

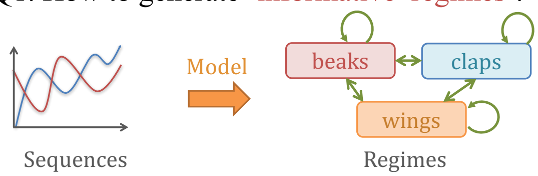
Model →

beaks | claps | wings

Regimes

Idea (1): MLCM: multi-level chain model

Q1. How to generate 'informative' regimes ?

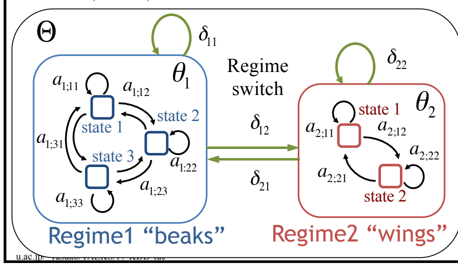


Idea (1): Multi-level chain model
- –HMM-based probabilistic model
- –with "across-regime" transitions

Idea (1): MLCM: multi-level chain model

Details

$\Theta = \{\underbrace{\theta_1, \theta_2, \ldots \theta_r}_{\text{r regimes (HMMs)}}, \underbrace{\Delta_{r \times r}}_{\text{across-regime transition prob.}}\}$ $\quad (\theta_i = \underbrace{\{\pi, A, B\}}_{\text{Single HMM parameters}})$



Regimes r=2
Regime 1 (k=3)
Regime 2 (k=2)

Idea (2): model description cost

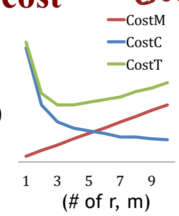
Q2. How to decide # of regimes/segments ?

Idea (2): Model description cost
- Minimize encoding cost
- find "optimal" # of segments/regimes

Idea (2): model description cost

Idea: Minimize encoding cost!

$\min \left(\underbrace{\boxed{Cost_M(M)}}_{\text{Model cost}} + \underbrace{\boxed{Cost_C(X|M)}}_{\text{Coding cost}} \right)$



Good compression ⟳ Good description

Idea (2): model description cost

Details

Total cost of bundle X, given C

$C = \{m, r, S, \Theta, F\}$

$$Cost_T(\boldsymbol{X}; \mathcal{C}) = Cost_T(\boldsymbol{X}; m, r, \mathcal{S}, \boldsymbol{\Theta}, \mathcal{F})$$
$$= \log^*(n) + \log^*(d) + \log^*(m) + \log^*(r) + m\log(r)$$
$$+ \sum_{i=1}^{m-1} \log^* |s_i| + Cost_M(\boldsymbol{\Theta}) + Cost_C(\boldsymbol{X}|\boldsymbol{\Theta}) \qquad (6)$$

Idea (2): model description cost

Details

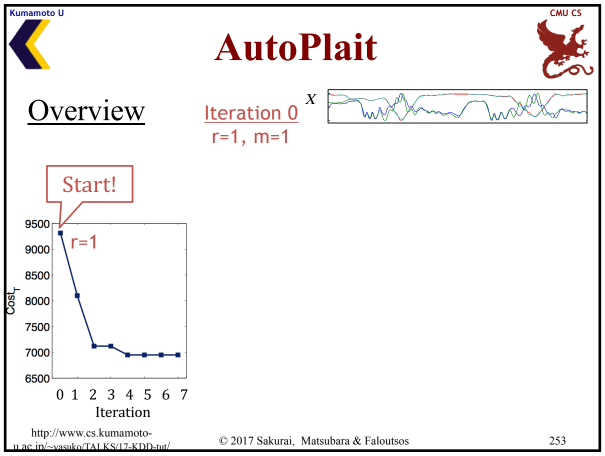
Total cost of bundle X, given C

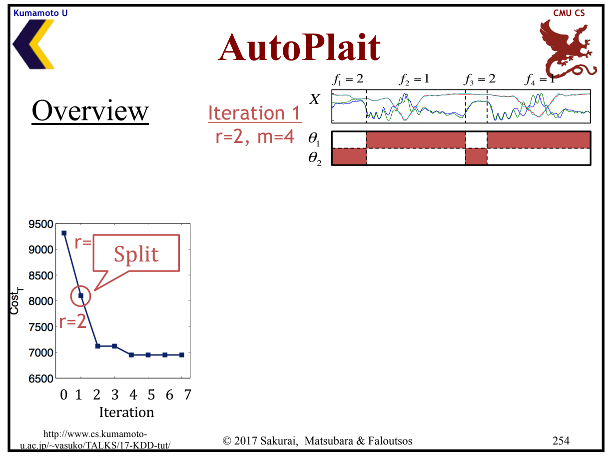
$C = \{m, r, S, \Theta, F\}$

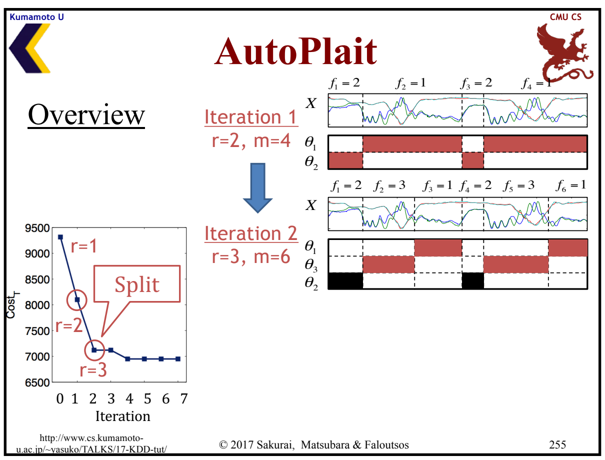
duration/dimensions # of segments/regimes segment-membership F

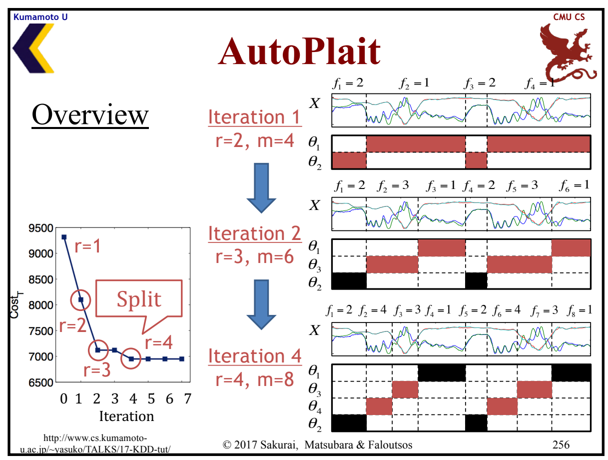
$$Cost_T(\boldsymbol{X}; \mathcal{C}) = Cost_T(\boldsymbol{X}; m, r, \mathcal{S}, \boldsymbol{\Theta}, \mathcal{F})$$
$$= \boxed{\log^*(n) + \log^*(d)} + \boxed{\log^*(m) + \log^*(r)} + \boxed{m\log(r)}$$
$$+ \boxed{\sum_{i=1}^{m-1} \log^* |s_i|} + \boxed{Cost_M(\boldsymbol{\Theta})} + \boxed{Cost_C(\boldsymbol{X}|\boldsymbol{\Theta})} \qquad (6)$$

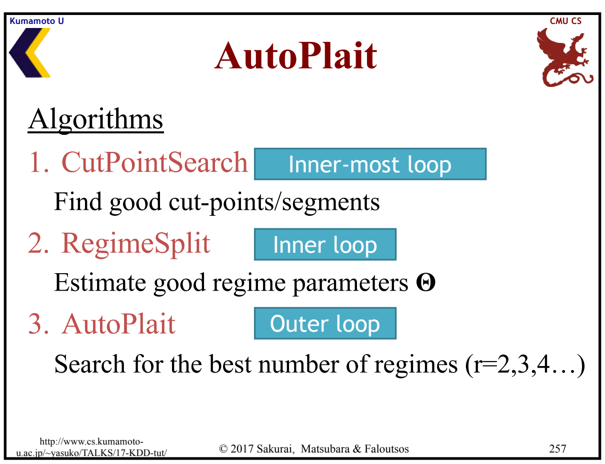
segment lengths Model description cost of Θ Coding cost of X given Θ

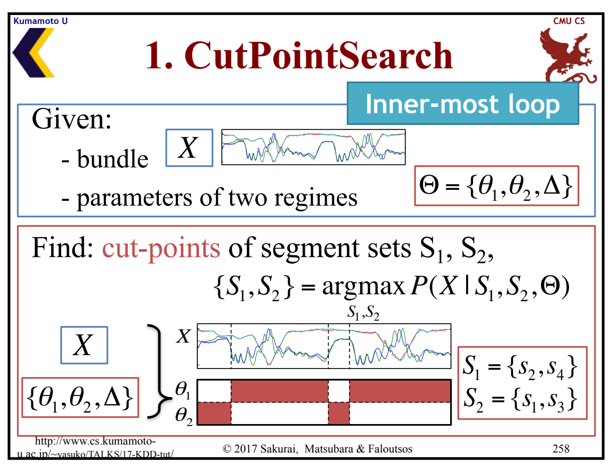

AutoPlait
Overview
Iteration 0
r=1, m=1
X
Start!
r=1
Cost_T
Iteration
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
253


AutoPlait
Overview
Iteration 1
r=2, m=4
Split
r=2
Cost_T
Iteration
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
254


AutoPlait
Overview
Iteration 1
r=2, m=4
Iteration 2
r=3, m=6
r=1
Split
r=2
r=3
Cost_T
Iteration
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
255


AutoPlait
Overview
Iteration 1
r=2, m=4
Iteration 2
r=3, m=6
Iteration 4
r=4, m=8
r=1
Split
r=2
r=3
r=4
Cost_T
Iteration
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
256

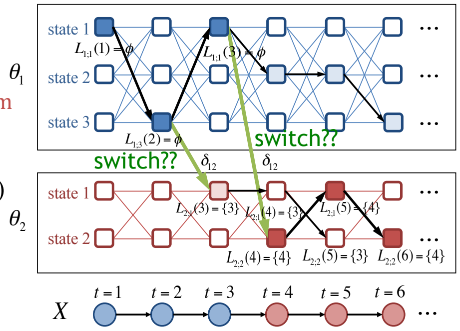

AutoPlait
Algorithms
1. CutPointSearch
Inner-most loop
Find good cut-points/segments
2. RegimeSplit
Inner loop
Estimate good regime parameters Θ
3. AutoPlait
Outer loop
Search for the best number of regimes (r=2,3,4…)
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
257


1. CutPointSearch
Inner-most loop
Given:
- bundle
- parameters of two regimes
Θ = {θ1, θ2, Δ}
Find: cut-points of segment sets S1, S2,
{S1, S2} = argmax P(X | S1, S2, Θ)
S1 = {s2, s4}
S2 = {s1, s3}
http://www.cs.kumamoto-u.ac.jp/~yasuko/TALKS/17-KDD-tut/
© 2017 Sakurai, Matsubara & Faloutsos
258

1. CutPointSearch

DP algorithm to compute likelihood:

$P(X \mid \Theta)$



1. CutPointSearch

Theoretical analysis

Scalability

- It takes $O(ndk^2)$ time (only single scan)
  - n: length of X
  - d: dimension of X
  - k: # of hidden states in regime

Accuracy

It guarantees the optimal cut points

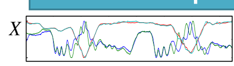
- (Details in paper)

2. RegimeSplit

Inner loop

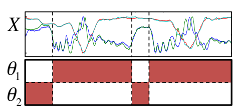
Given:

- bundle X



Find: two regimes

1. find cut-points of segment sets: S_1, S_2
2. estimate parameters of two regimes:

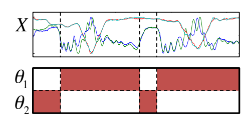


$\Theta = \{\theta_1, \theta_2, \Delta\}$

2. RegimeSplit

Two-phase iterative approach

- Phase 1: (CutPointSearch)
  - Split segments into two groups : S_1, S_2
- Phase 2: (BaumWelch)
  - Update model parameters: $\Theta = \{\theta_1, \theta_2, \Delta\}$



$S_1 = \{s_2, s_4\}$
$S_2 = \{s_1, s_3\}$ Phase 1

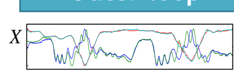
$\{\theta_1, \theta_2, \Delta\}$ Phase 2

3. AutoPlait

Outer loop

Given:

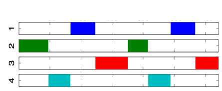
- bundle X



Find: r regimes (r=2, 3, 4, …)

- i.e., find full parameter set

$C = \{m, r, S, \Theta, F\}$

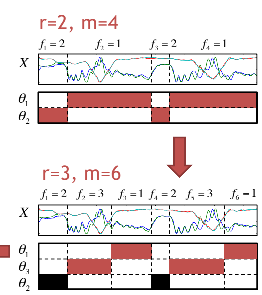


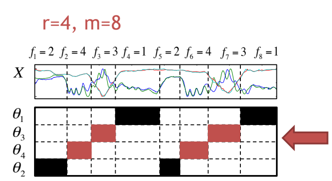
3. AutoPlait

Split regimes r=2,3,…, as long as cost keeps decreasing

- Find appropriate # of regimes

$r = \min_r Cost_T(X; m, r, S, \Theta, F)$



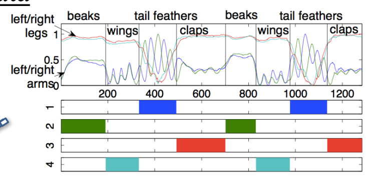


Kumamoto U | CMU CS

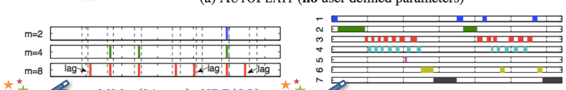
Sense-making

MoCap data

AutoPlait (NO magic numbers)



(a) AutoPlait (**no** user defined parameters)

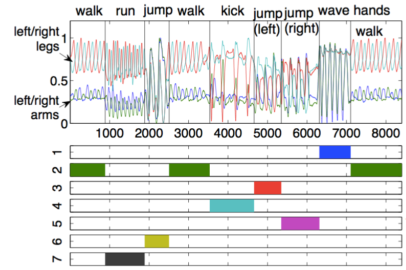


DynaMMo (Li et al., KDD'09)   pHMM (Wang et al., SIGMOD'11)

Kumamoto U | CMU CS

Sense-making

MoCap data

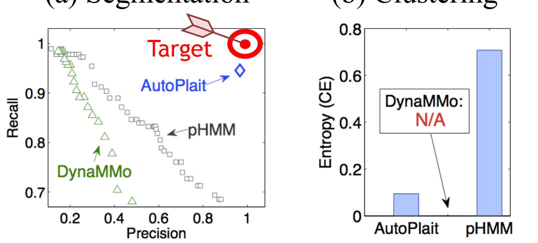


AutoPlait (NO magic numbers)

Kumamoto U | CMU CS

Q2. Accuracy

(a) Segmentation   (b) Clustering



(a) Precision and recall (higher is better)   (b) CE score (lower is better)

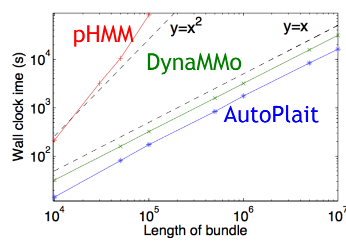
AutoPlait needs "no magic numbers"

Kumamoto U | CMU CS

Q3. Scalability

Wall clock time vs. data size (length) : n

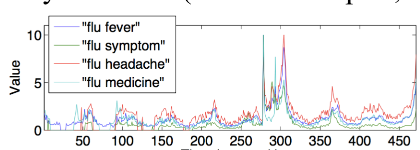
AutoPlait scales linearly, i.e., O(n)



Kumamoto U | CMU CS

App. Event discovery (GoogleTrend)

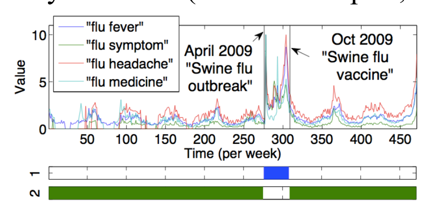
Anomaly detection (flu-related topics,10 years)



Kumamoto U | CMU CS

App. Event discovery (GoogleTrend)

Anomaly detection (flu-related topics,10 years)



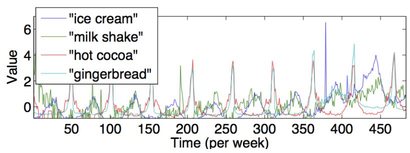
(a) Flu-related topics (regimes $r = 2$)

AutoPlait detects 1 unusual spike in 2009 (i.e., swine flu)

Kumamoto U CMU CS

App. Event discovery (GoogleTrend)

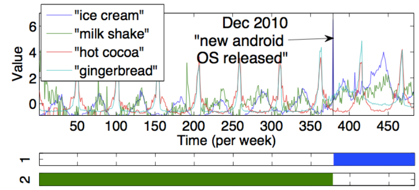
Turning point detection (seasonal sweets topics)



Kumamoto U CMU CS

App. Event discovery (GoogleTrend)

Turning point detection (seasonal sweets topics)



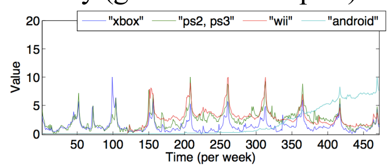
(b) Seasonal sweets topics (regimes $r = 2$)

Trend suddenly changed in 2010 (release of android OS "Ginger bread", "Ice Cream Sandwich")

Kumamoto U CMU CS

App. Event discovery (GoogleTrend)

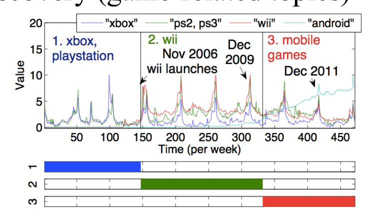
Trend discovery (game-related topics)



Kumamoto U CMU CS

App. Event discovery (GoogleTrend)

Trend discovery (game-related topics)



(c) Game-related topics (regimes $r = 3$)

It discovers 3 phases of "game console war" (Xbox&PlayStation/Wii/Mobile social games)

Kumamoto U CMU CS

Industrial contribution

- Automobile sensor data
  - location, velocity, longitudinal/lateral acceleration

Kumamoto U CMU CS

Code at

- http://www.cs.kumamoto-u.ac.jp/~yasuko/software.html

Kumamoto U CMU CS

Part 1 – Conclusions

- Motivation
- Similarity Search and Indexing
- Feature extraction
- Linear forecasting
- Streaming pattern discovery
- Automatic mining

Kumamoto U CMU CS

Part 1 – Conclusions

- Motivation
- Similarity Search and Indexing
  - Euclidean/time-warping
  - extract features
  - index (SAM, R-tree)
- Feature extraction
  - SVD, ICA, DFT, DWT (multi-scale windows)

Kumamoto U CMU CS

Part 1 – Conclusions

- Linear forecasting
  - AR, RLS
- Streaming pattern discovery
  - RLS, "incremental" wavelet transform
  - Multi-scale windows
- Automatic mining
  - MDL

Kumamoto U CMU CS

References

- Yunyue Zhu, Dennis Shasha ``*StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time*'' VLDB, August, 2002. pp. 358-369.
- Spiros Papadimitriou, Jimeng Sun, Christos Faloutsos. *Streaming Pattern Discovery in Multiple Time-Series*. VLDB 2005.
- Yasushi Sakurai, Spiros Papadimitriou, Christos Faloutsos. *BRAID: Stream Mining through Group Lag Correlations*. SIGMOD 2005.
- Anna C. Gilbert, Yannis Kotidis, S. Muthukrishnan, Martin Strauss. *Surfing Wavelets on Streams: One-Pass Summaries for Approximate Aggregate Queries*. VLDB 2001.
- Sudipto Guha, Nick Koudas, Amit Marathe, Divesh Srivastava. *Merging the Results of Approximate Match Operations*. VLDB 2004.
- Anna C. Gilbert, Sudipto Guha, Piotr Indyk, S. Muthukrishnan, Martin Strauss. *Near-optimal sparse fourier representations via sampling*. STOC 2002.

Kumamoto U CMU CS

References

- Nick Koudas, Beng Chin Ooi, Kian-Lee Tan, Rui Zhang. *Approximate NN queries on Streams with Guaranteed Error/performance Bounds*. VLDB 2004.
- Flip Korn, S. Muthukrishnan, Divesh Srivastava. *Reverse Nearest Neighbor Aggregates Over Data Streams*. VLDB 2002.
- Sudipto Guha, Nick Koudas, Kyuseok Shim. *Data-streams and histograms*. STOC 2001.
- Sudipto Guha, Nina Mishra, Rajeev Motwani, Liadan O'Callaghan. *Clustering Data Streams*. FOCS 2000.
- Charu C. Aggarwal, Jiawei Han, Jianyong Wang, Philip S. Yu. *A Framework for Clustering Evolving Data Streams*. VLDB 2003.

Kumamoto U CMU CS

References

- Piotr Indyk, Nick Koudas, S. Muthukrishnan. *Identifying Representative Trends in Massive Time Series Data Sets Using Sketches*. VLDB 2000.
- Graham Cormode, S. Muthukrishnan. *An improved data stream summary: the count-min sketch and its applications*. J. Algorithms 55 (1), 2005.
- Graham Cormode, Flip Korn, S. Muthukrishnan, Divesh Srivastava. *Finding Hierarchical Heavy Hitters in Data Streams*. VLDB 2003.
- Piotr Indyk. *Stable Distributions, Pseudorandom Generators, Embeddings and Data Stream Computation*. FOCS 2000.
- Yunyue Zhu, Dennis Shasha. *Efficient elastic burst detection in data streams*. KDD 2003.

Kumamoto U

CMU CS

References

- Eamonn J. Keogh, Selina Chu, David M. Hart, Michael J. Pazzani. *An Online Algorithm for Segmenting Time Series*. ICDM 2001.
- Spiros Papadimitriou, Philip S. Yu. *Optimal multi-scale patterns in time series streams*. SIGMOD 2006.
- Flip Korn, S. Muthukrishnan, Yihua Wu. *Modeling skew in data streams*. SIGMOD 2006.
- Yasushi Sakurai, Christos Faloutsos, Masashi Yamamuro. *Stream Monitoring under the Time Warping Distance*. ICDE 2007.

Part 1

Similarity search, pattern discovery and summarization

Yasushi Sakurai (Kumamoto University)
Yasuko Matsubara (Kumamoto University)
Christos Faloutsos (Carnegie Mellon University)