
Sensors & wavelets

Christos Faloutsos
www.cs.cmu.edu/~christos

Roadmap

1) Roots: System R and Ingres
2) Implementation: buffering, indexing, q-opt
   ...
7) Data Analysis - data mining
   ...
   - sensors, time series, indexing and wavelets
   - sensors and forecasting
8) Benchmarks
9) Vision statements
   extras (streams/sensors, graphs, multimedia, web, fractals)

Outline

• Motivation
• Similarity Search and Indexing
• DSP (Digital Signal Processing)
• Linear Forecasting
• Bursty traffic - fractals and multifractals
• Non-linear forecasting
• Conclusions

Problem definition

• Given: one or more sequences
  \[ x_1, x_2, \ldots, x_i, \ldots \]
  \[ y_1, y_2, \ldots, y_n, \ldots \]
• Find
  - similar sequences; forecasts
  - patterns; clusters; outliers

Motivation - Applications

• Financial, sales, economic series
• Medical
  - ECGs + blood pressure etc monitoring
  - reactions to new drugs
  - elderly care
Motivation - Applications (cont’d)

- ‘Smart house’
  - sensors monitor temperature, humidity, air quality
- video surveillance

Motivation - Applications (cont’d)

- civil/automobile infrastructure
  - bridge vibrations [Oppenheim+02]
  - road conditions / traffic monitoring

Stream Data: automobile traffic

# cars

Automobile traffic

time

Motivation - Applications (cont’d)

- Weather, environment/anti-pollution
  - volcano monitoring
  - air/water pollutant monitoring

Stream Data: Sunspots

# sunspots per month

Sunspot Data

time

Motivation - Applications (cont’d)

- Computer systems
  - ‘Active Disks’ (buffering, prefetching)
  - web servers (ditto)
  - network traffic monitoring
  - ...
Stream Data: Disk accesses

Settings & Applications

- One or more sensors, collecting time-series data

Settings & Applications

Each sensor collects data \((x_1, x_2, \ldots, x_n, \ldots)\)

Settings & Applications

Some sensors ‘report’ to others or to the central site

Settings & Applications

Goal #1:
Finding patterns in a single time sequence

Settings & Applications

Goal #2:
Finding patterns in many time sequences
Problem #1:
Goal: given a signal (e.g., #packets over time)
Find: patterns, periodicities, and/or compress

Problem #2: Forecast
Given $x_r, x_{r+1}, \ldots$, forecast $x_{r+i}$

Problem #2': Similarity search
Eg., Find a 3-tick pattern, similar to the last one

Problem #3:
- Given: A set of correlated time sequences
- Forecast ‘Sent(t)’

Differences from DSP/Stat
- Semi-infinite streams
  - we need on-line, ‘any-time’ algorithms
- Can not afford human intervention
  - need automatic methods
- sensors have limited memory / processing / transmitting power
  - need for (lossy) compression

Important observations
Patterns, rules, forecasting and similarity indexing are closely related:
- To do forecasting, we need
  - to find patterns/rules
  - to find similar settings in the past
- to find outliers, we need to have forecasts
  - (outlier = too far away from our forecast)
Important topics NOT in this tutorial:

- Continuous queries
  - [Babu+ Widom] [Gehrke+][Madden+03]
- Categorical data streams
  - [Hatonen+96]
- Outlier detection (discontinuities)
  - [Breunig+00]

Outline

- Motivation
- Similarity Search and Indexing
  - distance functions: Euclidean; Time-warping
  - indexing
  - feature extraction
- DSP
- ...

Importance of distance functions

Subtle, but absolutely necessary:
- A 'must' for similarity indexing (-> forecasting)
- A 'must' for clustering

Two major families
  - Euclidean and \( L_p \) norms
  - Time warping and variations

Euclidean and \( L_p \)

\[
D(\bar{x}, \bar{y}) = \sum_{i=1}^{n} (x_i - y_i)^2
\]

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

- \( L_1 \): city-block = Manhattan
- \( L_2 \): Euclidean
- \( L_{\infty} \)

Observation #1

- Time sequence -> n-d vector
Observation #2

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

Time Warping

- allow accelerations - decelerations
  – (with or w/o penalty)
- THEN compute the (Euclidean) distance (+ penalty)
- related to the string-editing distance

Q: how to compute it?
A: dynamic programming

\[ D(i, j) = \text{cost to match prefix of length } i \text{ of first sequence } x \text{ with prefix of length } j \text{ of second sequence } y \]

Thus, with no penalty for stutter, for sequences
\[ x_1, x_2, \ldots, x_L, \quad y_1, y_2, \ldots, y_J \]

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

Time warping

- Complexity: \( O(M \times N) \) - 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]
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]
  See tutorial by [Gunopulos Das, SIGMOD01]

Conclusions

Prevailing distances:
- Euclidean and
- time-warping

Outline

- Motivation
- Similarity Search and Indexing
  - distance functions
  - indexing
  - feature extraction
- DSP
- ...

Indexing

Problem:
- given a set of time sequences,
- find the ones similar to a desirable query sequence

Idea: ‘GEMINI’

Eg., ‘find stocks similar to MSFT’
Seq. scanning: too slow
How to accelerate the search?
‘GEMINI’ - Pictorially

GEMINI

Solution: Quick-and-dirty’ filter:
- extract \( n \) features (numbers, eg., avg., etc.)
- map into a point in \( n \)-d feature space
- organize points with off-the-shelf spatial access method (‘SAM’)
- discard false alarms

Examples of GEMINI

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

Examples of GEMINI

- Even on other-than-sequence data:
  - Images (QBIC) [JIIS94]
  - tumor-like shapes [VLDB96]
  - video [Informedia + S-R-trees]
  - automobile part shapes [Kriegel+97]

Indexing - SAMs

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

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:
R-trees

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

Conclusions

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

- very good for compressing real signals
- more details on DFT/DCT/DWT: later

DFT and stocks

- Dow Jones Industrial index, 6/18/2001-12/21/2001
- just 3 DFT coefficients give very good approximation

SVD

- **THE** optimal method for dimensionality reduction
  - (under the Euclidean metric)

Outline

- Motivation
- Similarity Search and Indexing
  - distance functions
  - indexing
  - feature extraction
    - DFT, DWT, DCT (data independent)
    - SVD etc (data dependent)
    - MDS, FastMap

Singular Value Decomposition (SVD)

- SVD (~LSI ~ KL ~ PCA ~ spectral analysis...)
  - LSI: S. Dumais; M. Berry
  - KL: eg, Duda+Hart
  - PCA: eg., Jolliffe
  - Details: [Press+], [Faloutsos96]
SVD

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

SVD shortcuts

- random projections (Johnson-Lindenstrauss thm [Papadimitriou+ PODS98])

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])

Feature extraction - w/ fractals

- Main idea: drop those attributes that don’t affect the intrinsic (‘fractal’) dimensionality [Traina+, SBBD 2000]
- ie., drop attributes that depend on others (linearly or non-linearly!)

Outline

- Motivation
- Similarity Search and Indexing
  - distance functions
  - indexing
  - feature extraction
    - DFT, DWT, DCT (data independent)
    - SVD (data dependent)
    - MDS, FastMap
MDS / FastMap

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

FastMap

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

FastMap: Application

VideoTrails [Kobla+97]

scene-cut detection (about 10% errors)
Conclusions - Practitioner’s guide
Similarity search in time sequences
1) establish/choose distance (Euclidean, time-warping,...)
2) extract features (SVD, DWT, MDS), and use an SAM (R-tree/variant) or a Metric Tree (M-tree)
2') for high intrinsic dimensionalities, consider sequential scan (it might win…)

Books
- C. Faloutsos: Searching Multimedia Databases by Content, Kluwer Academic Press, 1996 (introduction to SVD, and GEMINI)

References
- Berry, Michael: http://www.cs.utk.edu/~lsi/

References

References

References
References


References


References

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

Part 2

DSP (Digital Signal Processing)

Outline

- Motivation
- Similarity Search and Indexing
  - DSP (DFT, DWT)
    - Linear Forecasting
    - Bursty traffic - fractals and multifractals
    - Non-linear forecasting
    - Conclusions

Outline

- DFT
  - Definition of DFT and properties
  - how to read the DFT spectrum
- DWT
  - Definition of DWT and properties
  - how to read the DWT scalogram
Introduction - Problem#1

Goal: given a signal (e.g., packets over time)
Find: patterns and/or compress

Count

lynx caught per year
(packets per day;
automobiles per hour)

year

What does DFT do?

A: highlights the periodicities

DFT: definition

- For a sequence $x_0, x_1, \ldots, x_{n-1}$
- the (n-point) Discrete Fourier Transform is
- $X_f = \frac{1}{\sqrt{n}} \sum_{k=0}^{n-1} x_k \exp(-j2\pi ft/n), \quad f = 0, \ldots, n-1$
- inverse DFT

DFT: examples

- Good news: Available in all symbolic math packages, e.g., in ‘mathematica’
- $x = [1, 2, 1, 2]$;
- $X = \text{Fourier}(x)$;
- Plot[ Abs[X] ];
DFT: examples

- Low frequency sinusoid

- Higher frequency sinusoid

- Sinusoid - symmetry property: $X_f = X^{n_f}$

Outline

- Motivation
- Similarity Search and Indexing
- DSP
- Linear Forecasting
- Bursty traffic - fractals and multifractals
- Non-linear forecasting
- Conclusions
Outline

- Motivation
- Similarity Search and Indexing
- DSP
  - DFT
    - Definition of DFT and properties
  - DWT

DFT: Amplitude spectrum

Amplitude: \( A_j^2 = \text{Re}^2(X_j) + \text{Im}^2(X_j) \)

Outline

- Motivation
- Similarity Search and Indexing
- DSP
  - DFT
    - Definition of DFT and properties
    - how to read the DFT spectrum
  - DWT

DFT: Amplitude spectrum

Amplitude: \( A_j^2 = \text{Re}^2(X_j) + \text{Im}^2(X_j) \)

DFT: Amplitude spectrum

- excellent approximation, with only 2 frequencies!
- so what?

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

DFT - Conclusions

- It 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)

Outline

- Motivation
- Similarity Search and Indexing
- DSP
  - DFT
  - DWT
    - Definition of DWT and properties
    - how to read the DWT scalogram

Problem #1:

Goal: given a signal (eg., #packets over time)
Find: patterns, periodicities, and/or **compress**

Wavelets - DWT

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

Wavelets - DWT

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

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

Wavelets - DWT

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

Wavelets - DWT

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

Wavelets - DWT

- Answer: multiple window sizes! -> DWT

Haar Wavelets

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

Wavelets - construction

x0 x1 x2 x3 x4 x5 x6 x7
Wavelets - construction

level 1  d1,0
       \__________
       |          |
       |          |
x0  x1  x2  x3  x4  x5  x6  x7

level 0  s1,0

Wavelets - construction

level 2  d2,0
          \_______
          |       |
          |       |
x0  x1  x2  x3  x4  x5  x6  x7

level 1  d1,0
          \_____
          |    |
          |    |
x0  x1  x2  x3  x4  x5  x6  x7

level 0  s1,0

Wavelets - construction

etc ...

d2,0  s2,0
       \_____
       |    |
       |    |
x0  x1  x2  x3  x4  x5  x6  x7

level 1  d1,0
          \_____
          |    |
          |    |
x0  x1  x2  x3  x4  x5  x6  x7

level 0  s1,0

Q: map each coefficient on the time-freq. plane

f

Haar waves - code

```perl
#!/usr/bin/perl5

use strict;
use warnings;

use Math::Complex;  
my $eps = 1e-12;  
my ($half, $level, $t, ...) = @ARGV;  
my ($smooth, $isReal, $isComplex) = @ARGV;  

my $smooth = $smooth || 0;  
my $scale = 0;  
my ($coeff, $dcoeff) = @ARGV;  

my @val = ();  
my $i = 0;
```

Q: map each coefficient on the time-freq. plane

f

Haar waves - code

```perl
#!/usr/bin/perl5

use strict;
use warnings;

use Math::Complex;  
my $eps = 1e-12;  
my ($half, $level, $t, ...) = @ARGV;  
my ($smooth, $isReal, $isComplex) = @ARGV;  

my $smooth = $smooth || 0;  
my $scale = 0;  
my ($coeff, $dcoeff) = @ARGV;  

my @val = ();  
my $i = 0;
```
Wavelets - construction

Observation 1:
- '+' can be some weighted addition
- '-' is the corresponding weighted difference ('Quadrature mirror filters')

Observation 2: 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 - drill #1:

- Q: baritone/silence/soprano - DWT?

Outline

- Motivation
- Similarity Search and Indexing
- DSP
  - DFT
  - DWT
  - Definition of DWT and properties
  - how to read the DWT scalogram
Wavelets - Drill#1:

- Q: baritone/soprano - DWT?

Wavelets - Drill#2:

- Q: spike - DWT?

Wavelets - Drill#3:

- Q: weekly + daily periodicity, + spike - DWT?
Wavelets - Drill#3:

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

Wavelets - Drill#3:

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

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’
- Mammalian eye and ear: Gabor wavelets

Overall Conclusions

- DFT, DCT spot periodicities
- DWT: multi-resolution - matches processing of mammalian ear/eye better
- All three: powerful tools for compression, pattern detection in real signals
- All three: included in math packages
  - (matlab, ‘R’, mathematica, … - often in spreadsheets!)

Overall Conclusions

- DWT: very suitable for self-similar traffic
- DWT: used for summarization of streams [Gilbert+01], db histograms etc
Resources - software and urls


Resources: software and urls

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

Books


Additional Reading