

ECML PKDD >> 2005
16th european conference on machine learning
9th european conference on principles and practice of knowledge discovery in databases
porto > portugal > 3 to 7 october



A practical Time-Series Tutorial with MATLAB

Michalis Vlachos
IBM T.J. Watson Research Center
Hawthorne, NY, 10532



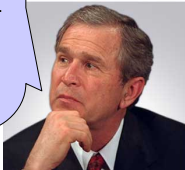
Tutorial | Time-Series with Matlab

About this tutorial

- The goal of this tutorial is to show you that **time-series research** (or research in general) can be made fun, when it involves visualizing **ideas**, that can be achieved with **concise programming**.
- **Matlab** enables us to do that.



Will I be able to use this MATLAB right away after the tutorial?



I am definately smarter than *her*, but I am not a time-series person, per-se. I wonder what I gain from this tutorial...

2

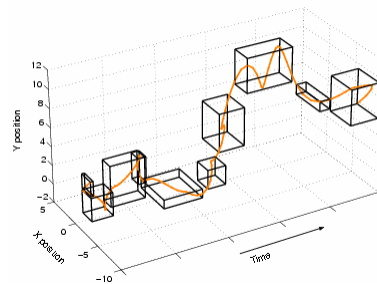
Disclaimer

- I am **not** affiliated with Mathworks in any way
- ... but I **do** like using Matlab a lot
 - since it makes my life easier
- **Errors and bugs are most likely contained in this tutorial.**
- **I might be responsible for some of them.**

3

Timeline of tutorial

- **Matlab introduction**
 - I will try to convince you that Matlab is cool
 - Brief introduction to its many features
- **Time-series with Matlab**
 - Introduction
 - Time-Series Representations
 - Distance Measures
 - Lower Bounding
 - Clustering/Classification/Visualization
 - Applications



4

What this tutorial is NOT about

- Moving averages
- Autoregressive models
- Forecasting/Prediction
- Stationarity
- Seasonality
- ..or about complex mathematical formulas

Let's not escape into mathematics. Let's stick with reality.

Oh...buy my books too!



Michael Crichton

PART I: Matlab Introduction

Why does anyone need Matlab?

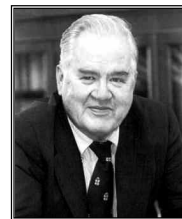
- **Matlab enables the efficient**
Exploratory Data Analysis (EDA)

“Science progresses through observation”
-- Isaac Newton



Isaac Newton

“The greatest value of a picture is that it forces us to notice what we never expected to see”
-- John Tukey



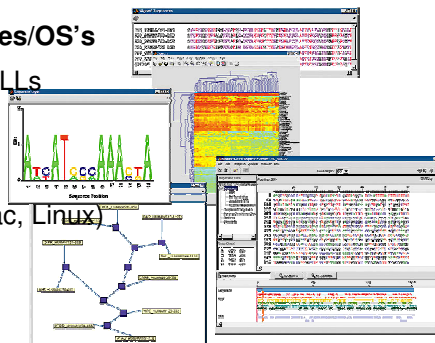
John Tukey

7

Matlab



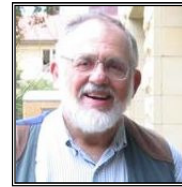
- **Interpreted Language**
 - Easy code maintenance (code is very compact)
 - Very fast array/vector manipulation
 - Support for OOP
- **Easy plotting and visualization**
- **Easy Integration with other Languages/OS's**
 - Interact with C/C++, COM Objects, DLLs
 - Build in Java support (and compiler)
 - Ability to make executable files
 - Multi-Platform Support (Windows, Mac, Linux)
- **Extensive number of Toolboxes**
 - Image, Statistics, Bioinformatics, etc



8

History of Matlab (MATrix LABoratory)

"The most important thing in the programming language is the name.
I have recently invented a very good name and now I am looking for a
suitable language". -- R. Knuth

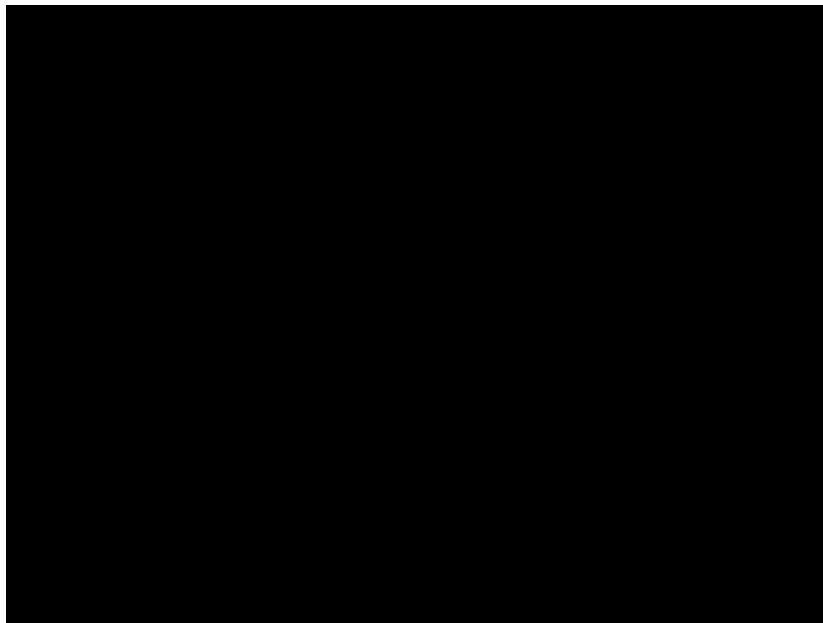


Cleve Moler

Programmed by Cleve Moler as an interface for EISPACK & LINPACK

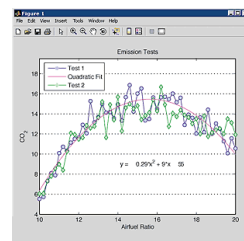
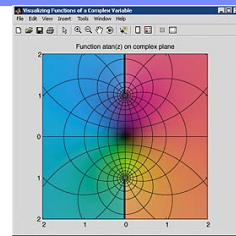
- **1957:** Moler goes to Caltech. Studies numerical Analysis
- **1961:** Goes to Stanford. Works with G. Forsythe on Laplacian eigenvalues.
- **1977:** First edition of Matlab; 2000 lines of Fortran
 - 80 functions (now more than 8000 functions)
- **1979:** Met with Jack Little in Stanford. Started working on porting it to C
- **1984:** Mathworks is founded

[Video: http://www.mathworks.com/company/aboutus/founders/origins_of_matlab_wm.html](http://www.mathworks.com/company/aboutus/founders/origins_of_matlab_wm.html)



Current State of Matlab/Mathworks

- **Matlab, Simulink, Stateflow**
- **Matlab version 7, service pack 2**
- **Used in variety of industries**
 - Aerospace, defense, computers, communication, biotech
- **Mathworks still is privately owned**
- **Used in >3,500 Universities, with >500,000 users worldwide**
- **2004 Revenue: 300 M.**
- **2004 Employees: 1,000**
- **Pricing:**
 - ~2000\$ (Commercial use),
 - ~100\$ (Student Edition)

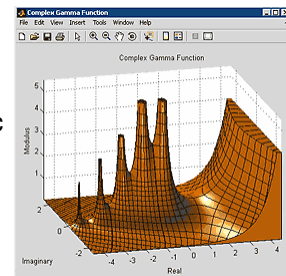
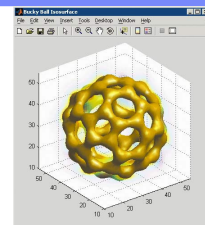


Money is better than poverty, if only for financial reasons. –Woody Allen

11

Who needs Matlab?

- **R&D companies for easy application deployment**
- **Professors**
 - Lab assignments
 - Matlab allows focus on *algorithms* not on language features
- **Students**
 - Batch processing of files
 - No more incomprehensible perl code!
 - Great environment for testing ideas
 - Quick coding of ideas, then porting to C/Java etc
 - Easy visualization
 - It's cheap! (for students at least...)



12

Starting up Matlab

Personally I'm always ready to learn, although I do not always like being taught.

Sir Winston Churchill

- **Dos/Unix like directory navigation**

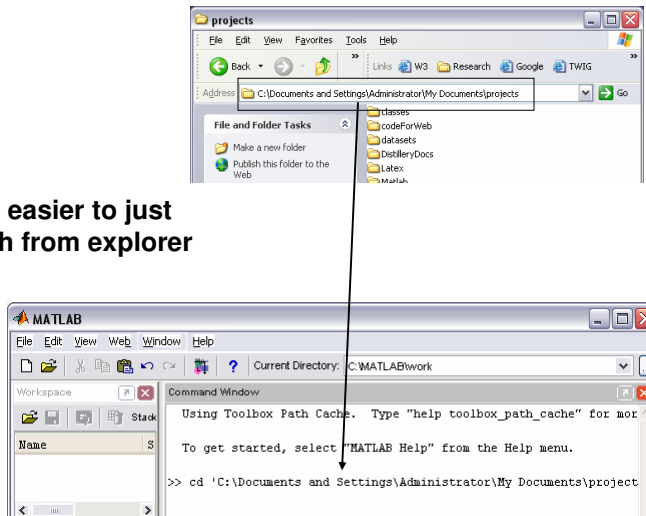
- **Commands like:**

- cd
- pwd
- mkdir

- **For navigation it is easier to just copy/paste the path from explorer**

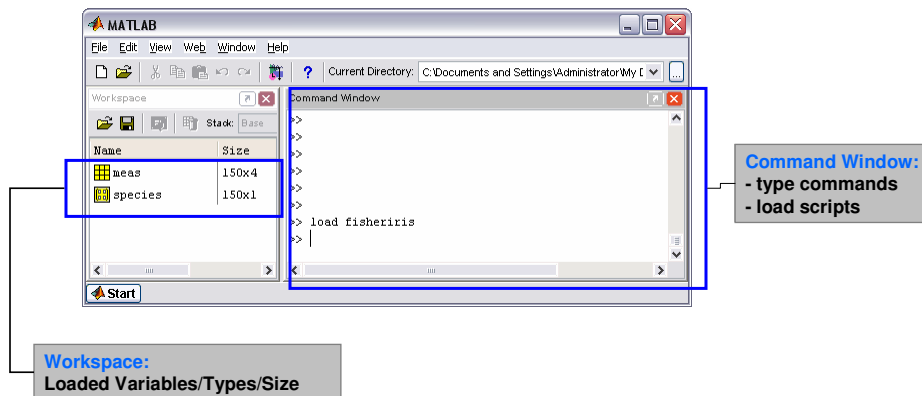
E.g.:

cd 'c:\documents\'



13

Matlab Environment



14

Tutorial | Time-Series with Matlab

Matlab Environment

The screenshot shows the MATLAB environment with the following components:

- Workspace:** A table showing loaded variables and their sizes.

Name	Size
meas	150x4
species	150x1
- Command Window:** A text area for entering commands. It contains the command `load fisheriris`. A callout box indicates: "Command Window: type commands - load scripts".
- Help Navigator:** A window showing the MATLAB Help Navigator. It includes a search bar and a list of topics. A callout box states: "Help contains a comprehensive introduction to all functions".

15

Tutorial | Time-Series with Matlab

Matlab Environment

The screenshot shows the MATLAB environment with the following components:

- Workspace:** A table showing loaded variables and their sizes.

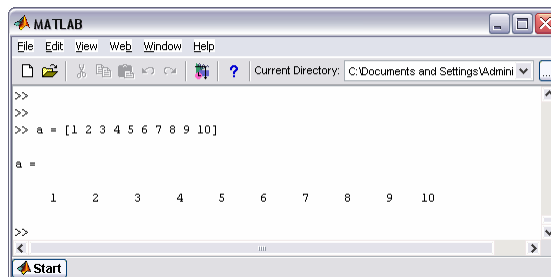
Name	Size
meas	150x4
species	150x1
- Command Window:** A text area for entering commands. It contains the command `load fisheriris`. A callout box indicates: "Command Window: type commands - load scripts".
- Examples of XYZ Plots in MATLAB:** A window showing a 3D mesh plot of a surface. The plot is titled "Examples of XYZ Plots in MATLAB" and shows a 3D surface with axes ranging from -10 to 30. A callout box states: "Excellent demos and tutorial of the various features and toolboxes".

16

Starting with Matlab

- Everything is **arrays**
- Manipulation of arrays is **faster** than regular manipulation with for-loops

```
a = [1 2 3 4 5 6 7 9 10] % define an array
```

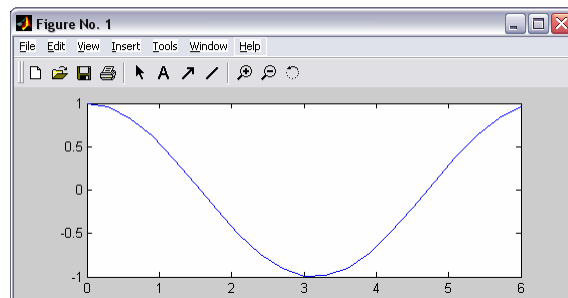


17

Populating arrays

- Plot sinusoid function

```
a = [0:0.3:2*pi] % generate values from 0 to 2pi (with step of 0.3)
b = cos(a) % access cos at positions contained in array [a]
plot(a,b) % plot a (x-axis) against b (y-axis)
```



Related:

```
linspace(-100,100,15); % generate 15 values between -100 and 100
```

18

Array Access

Access array elements

```
>> a(1)
ans =
    0
```

```
>> a(1:3)
ans =
    0    0.3000    0.6000
```

Set array elements

```
>> a(1) = 100
```

```
>> a(1:3) = [100 100 100]
```

```
MATLAB
File Edit View Web Window Help
Current Directory: C:\MATLAB\work
>> a(1:3) = [100 100 100]
a =
Columns 1 through 11
100.0000 100.0000 100.0000 0.9000 1.2000 1.5000 1.8000 2.1000 2.4000 2.7000 3.0000
Columns 12 through 21
3.3000 3.6000 3.9000 4.2000 4.5000 4.8000 5.1000 5.4000 5.7000 6.0000
>>
```

19

2D Arrays

Can access whole columns or rows

— Let's define a 2D array

```
>> a = [1 2 3; 4 5 6]
a =
```

```
    1    2    3
    4    5    6
```

```
>> a(2,2)
```

```
ans =
    5
```

```
>> a(1,:)
ans =
```

```
    1    2    3
```

```
>> a(:,1)
ans =
```

```
    1
    4
```

Row-wise access

Column-wise access



A good listener is not only popular everywhere, but after a while he gets to know something. —Wilson Mizner²⁰

Column-wise computation

- For arrays greater than 1D, all computations happen **column-by-column**

```
>> a = [1 2 3; 3 2 1]
a =
```

```
    1    2    3
    3    2    1
```

```
>> mean(a)
```

```
ans =

    2.0000    2.0000    2.0000
```

```
>> max(a)
```

```
ans =

    3    2    3
```

```
>> sort(a)
```

```
ans =

    1    2    1
    3    2    3
```

21

Concatenating arrays

- Column-wise or row-wise

```
>> a = [1 2 3];
>> b = [4 5 6];
>> c = [a b]
```

Row next to row

```
c =

    1    2    3    4    5    6
```

```
>> a = [1;2];
>> b = [3;4];
>> c = [a b]
```

Column next to column

```
c =

    1    3
    2    4
```

```
>> a = [1 2 3];
>> b = [4 5 6];
>> c = [a; b]
```

Row below to row

```
c =

    1    2    3
    4    5    6
```

```
>> a = [1;2];
>> b = [3;4];
>> c = [a; b]
```

Column below column

```
c =

    1
    2
    3
    4
```

22

Initializing arrays

Create array of ones [ones]

```
>> a = ones(1,3)
a =
    1    1    1

>> a = ones(1,3)*inf
a =
   Inf   Inf   Inf
```

```
>> a = ones(2,2)*5;
a =
    5    5
    5    5
```

Create array of zeroes [zeros]

- Good for initializing arrays

```
>> a = zeros(1,4)
a =
    0    0    0    0
```

```
>> a = zeros(3,1) + [1 2 3]'
a =
    1
    2
    3
```

23

Reshaping and Replicating Arrays

Changing the array shape [reshape]

- (eg, for easier column-wise computation)

```
>> a = [1 2 3 4 5 6]'; % make it into a column
>> reshape(a,2,3)

ans =
    1    3    5
    2    4    6
```

reshape(X,[M,N]):
[M,N] matrix of
columnwise version
of X

Replicating an array [repmat]

```
>> a = [1 2 3];
>> repmat(a,1,2)

ans =    1    2    3    1    2    3

>> repmat(a,2,1)
ans =
    1    2    3
    1    2    3
```

repmat(X,[M,N]):
make [M,N] tiles of X

24

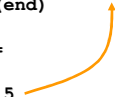
Useful Array functions

▪ Last element of array [**end**]

```
>> a = [1 3 2 5];
>> a(end)

ans =

    5
```



```
>> a = [1 3 2 5];
>> a(end-1)

ans =

    2
```


▪ Length of array [**length**]

```
>> length(a)

ans =

    4
```

Length = 4

a =

1	3	2	5
---	---	---	---

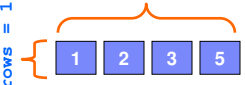

▪ Dimensions of array [**size**]

```
>> [rows, columns] = size(a)
rows = 1
columns = 4
```

columns = 4

rows = 1

1	2	3	5
---	---	---	---



25

Useful Array functions

▪ Find a specific element [**find**] **

```
>> a = [1 3 2 5 10 5 2 3];
>> b = find(a==2)

b =

    3    7
```

▪ Sorting [**sort**] ***

```
>> a = [1 3 2 5];
>> [s,i]=sort(a)

s =

    1    2    3    5

i =

    1    3    2    4
```

a =

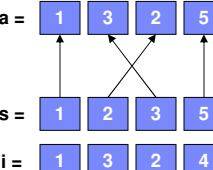
1	3	2	5
---	---	---	---

s =

1	2	3	5
---	---	---	---

i =

1	3	2	4
---	---	---	---



Indicates the index
where the element
came from

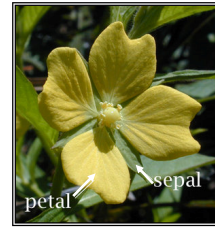
26

Visualizing Data and Exporting Figures

Use Fisher's Iris dataset

```
>> load fisheriris
```

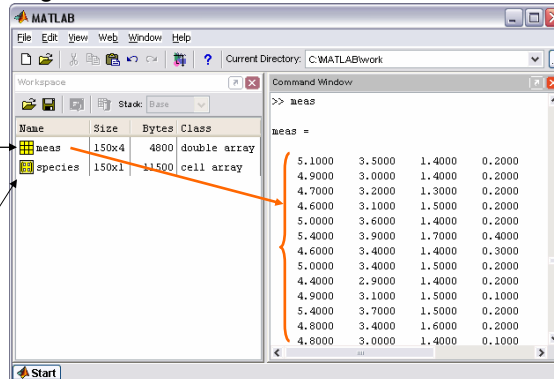
- 4 dimensions, 3 species
- Petal length & width, sepal length & width
- Iris:
 - virginica/versicolor/setosa



meas (150x4 array):
Holds 4D measurements

```
...
'versicolor'
'versicolor'
'versicolor'
'versicolor'
'versicolor'
'veirginica'
'veirginica'
'veirginica'
'veirginica'
```

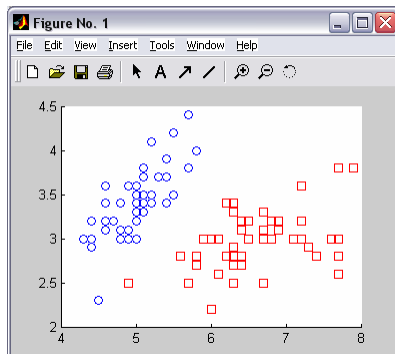
species (150x1 cell array):
Holds name of species for the specific measurement



27

Visualizing Data (2D)

```
>> idx_setosa = strcmp(species, 'setosa'); % rows of setosa data
>> idx_virginica = strcmp(species, 'virginica'); % rows of virginica
>>
>> setosa = meas(idx_setosa, [1:2]);
>> virgin = meas(idx_virginica, [1:2]);
>> scatter(setosa(:,1), setosa(:,2)); % plot in blue circles by default
>> hold on;
>> scatter(virgin(:,1), virgin(:,2), 'rs'); % red[r] squares[s] for these
```



idx_setosa

```
...
1
1
1
0
0
0
...
```

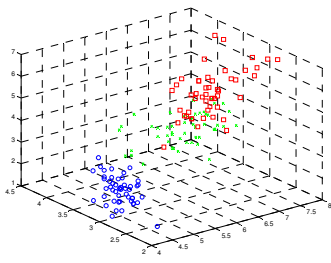
An array of zeros and ones indicating the positions where the keyword 'setosa' was found

28

Visualizing Data (3D)

```
>> idx_setosa = strcmp(species, 'setosa'); % rows of setosa data
>> idx_virginica = strcmp(species, 'virginica'); % rows of virginica
>> idx_versicolor = strcmp(species, 'versicolor'); % rows of versicolor

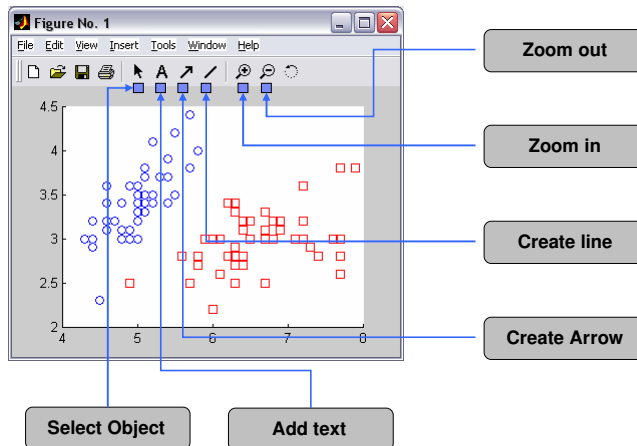
>> setosa = meas(idx_setosa, [1:3]);
>> virgin = meas(idx_virginica, [1:3]);
>> versi = meas(idx_versicolor, [1:3]);
>> scatter3(setosa(:,1), setosa(:,2), setosa(:,3)); % plot in blue circles by default
>> hold on;
>> scatter3(virgin(:,1), virgin(:,2), virgin(:,3), 'rs'); % red[r] squares[s] for these
>> scatter3(versi(:,1), versi(:,2), versi(:,3), 'gx'); % green x's
```



```
>> grid on; % show grid on axis
>> rotate3D on; % rotate with mouse
```

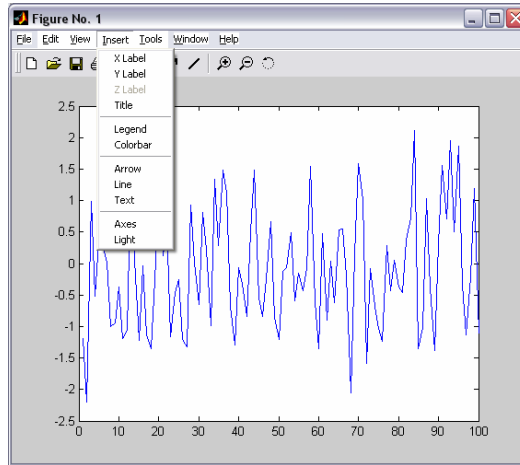
29

Changing Plots Visually



30

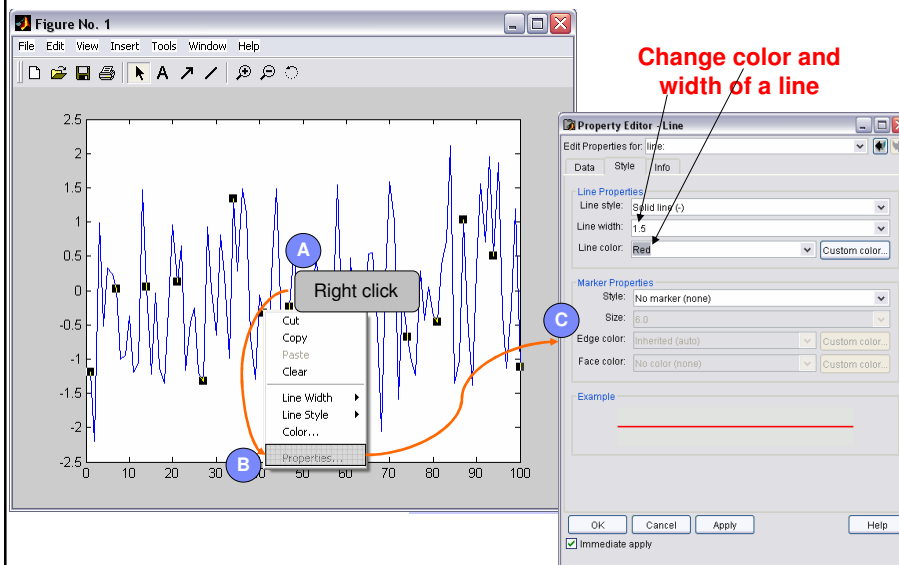
Changing Plots Visually



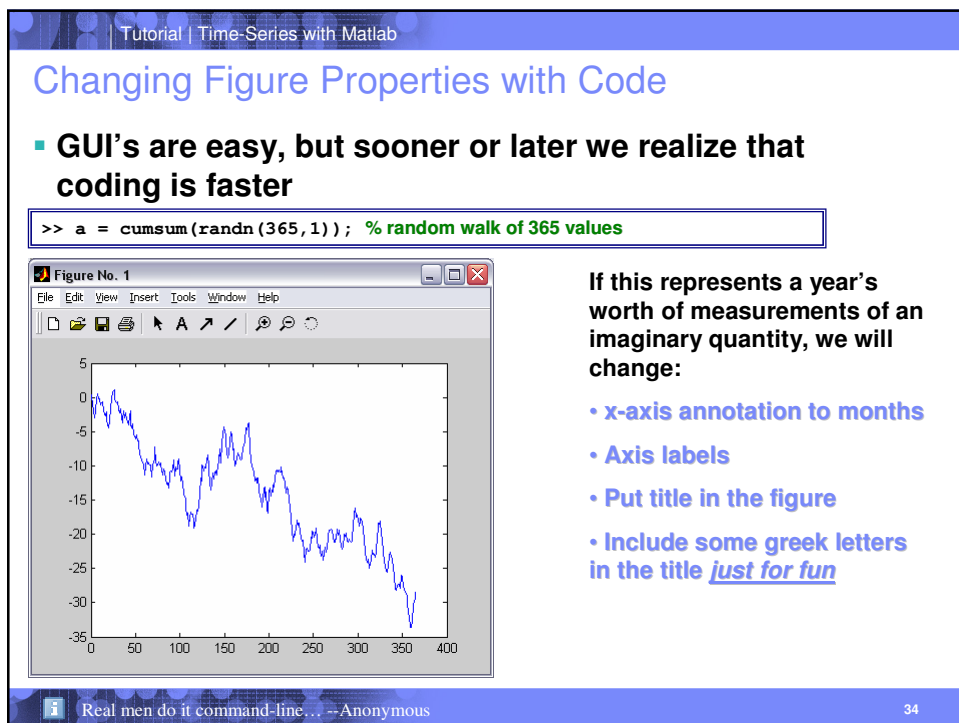
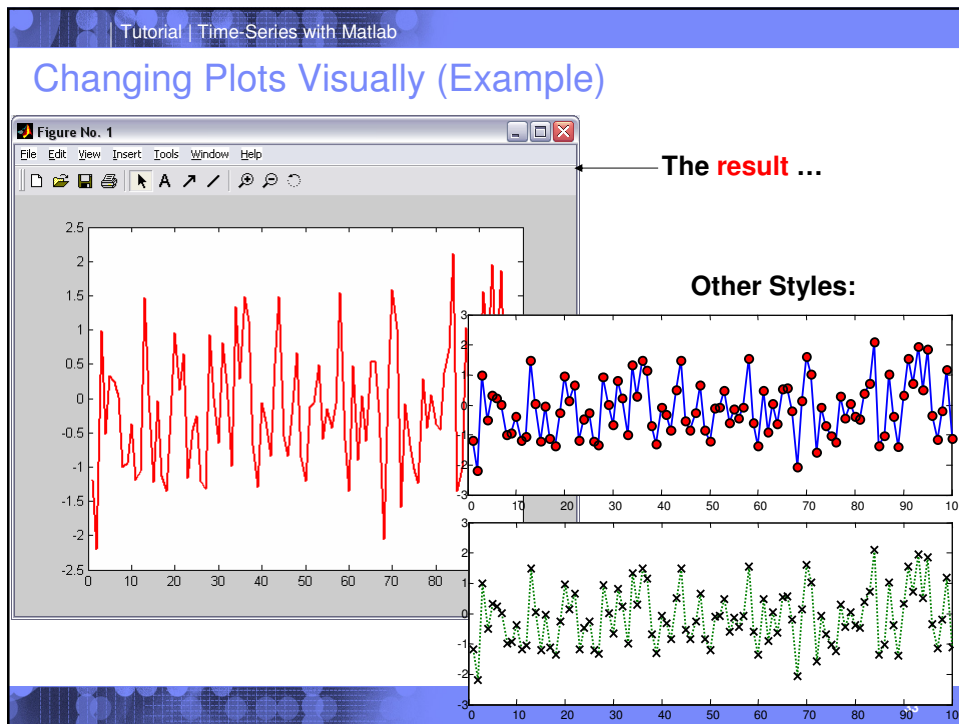
- **Add titles**
- **Add labels on axis**
- **Change tick labels**
- **Add grids to axis**
- **Change color of line**
- **Change thickness/ Linestyle**
- **etc**

31

Changing Plots Visually (Example)



32

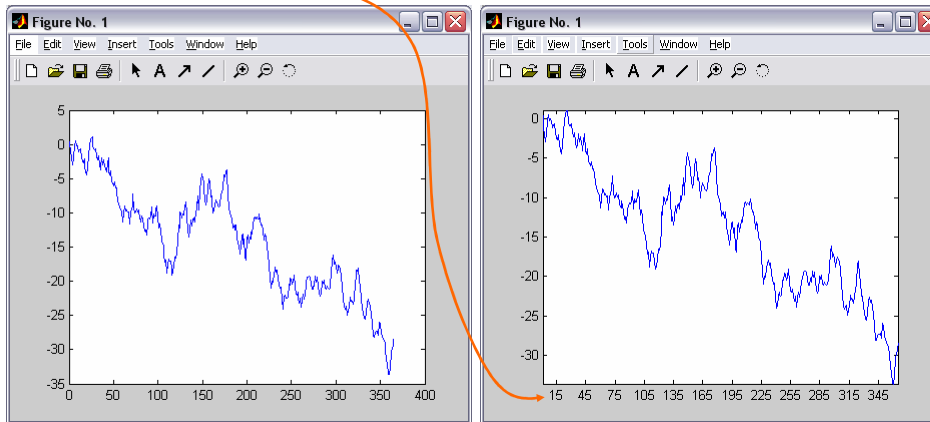


Changing Figure Properties with Code

Axis annotation to months

```
>> axis tight; % irrelevant but useful...
>> xx = [15:30:365];
>> set(gca, 'xtick', xx)
```

The **result** ...



Real men do it command-line... --Anonymous

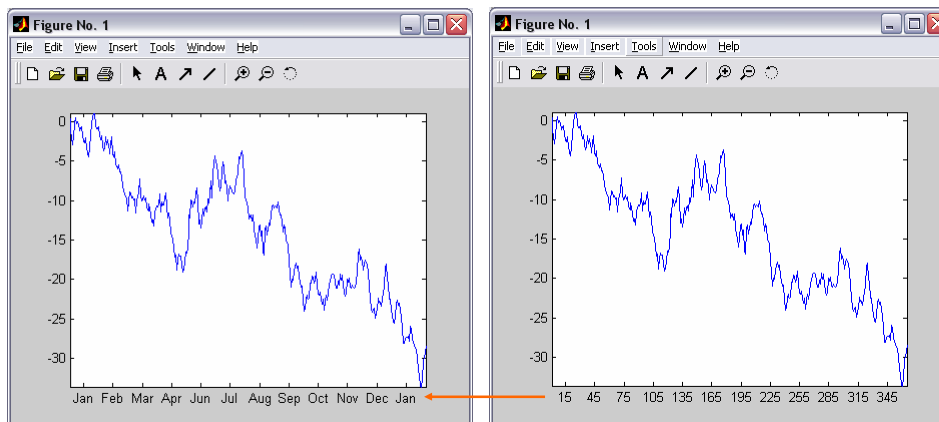
35

Changing Figure Properties with Code

Axis annotation to months

```
>> set(gca, 'xticklabel', ['Jan'; ...
    'Feb'; 'Mar'])
```

The **result** ...



Real men do it command-line... --Anonymous

36

Changing Figure Properties with Code

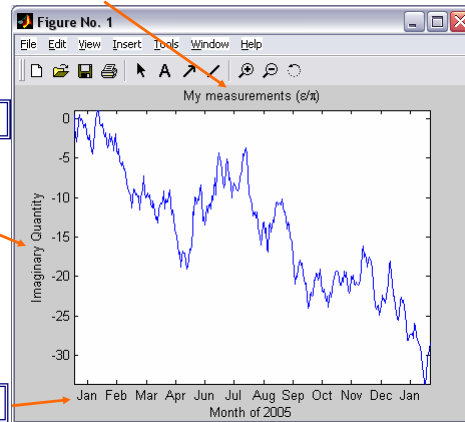
Axis labels and title

```
>> title('My measurements (\epsilon/\pi)')
```

```
>> ylabel('Imaginary Quantity')
```

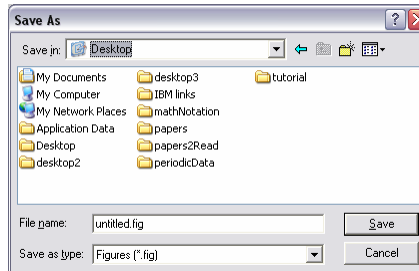
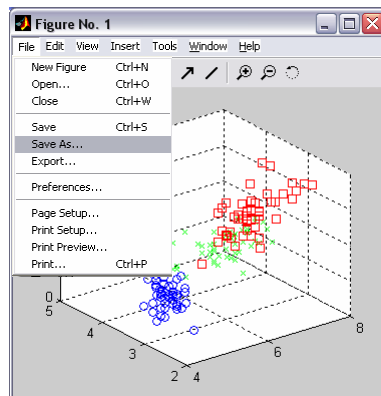
```
>> xlabel('Month of 2005')
```

Other latex examples:
 α , β , $e^{-\alpha}$ etc



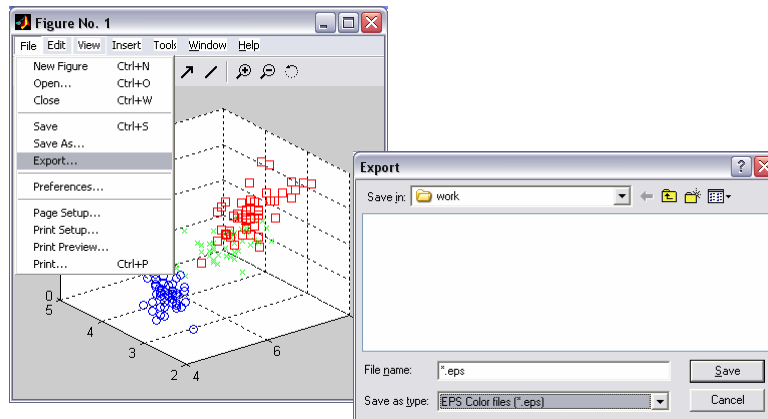
Saving Figures

Matlab allows to save the figures (.fig) for later processing



.fig can be later
opened through
Matlab

Exporting Figures

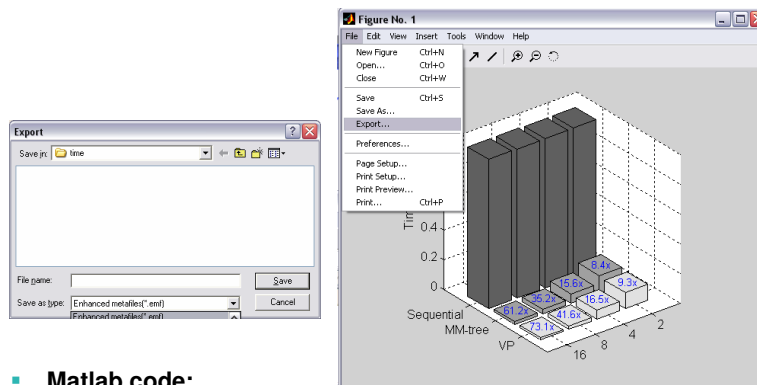


Export to:
emf, eps, jpg, etc

39

Exporting figures (code)

- You can also achieve the same result with Matlab code

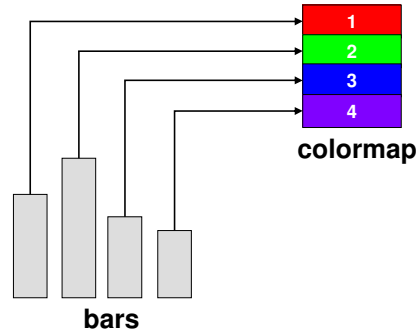


- Matlab code:

```
% extract to color eps
print -depsc myImage.eps; % from command-line
print(gcf, '-depsc', 'myImage') % using variable as name
```

40

Visualizing Data - 2D Bars

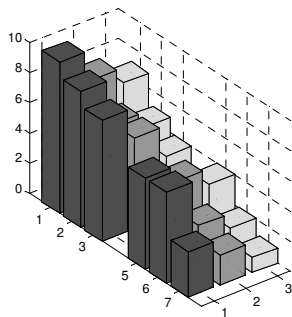


```
time = [100 120 80 70]; % our data
h = bar(time); % get handle
cmap = [1 0 0; 0 1 0; 0 0 1; .5 0 1]; % colors
colormap(cmap); % create colormap

cdata = [1 2 3 4]; % assign colors
set(h, 'CDataMapping', 'direct', 'CData', cdata);
```

41

Visualizing Data - 3D Bars

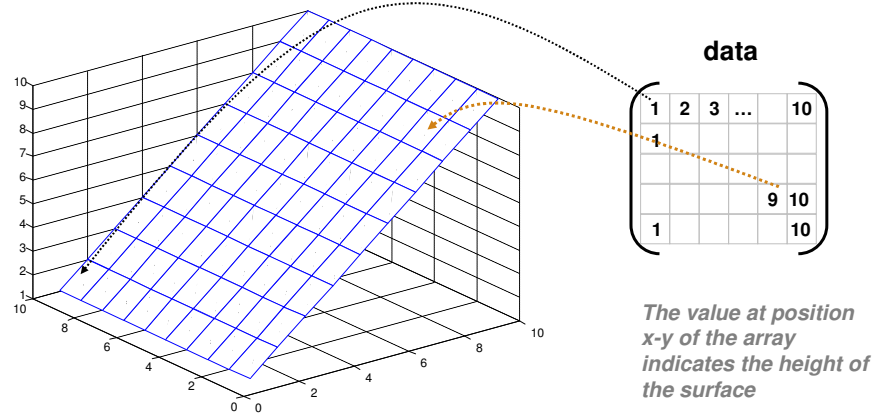


data			colormap		
10	8	7	0	0	0
9	6	5	0.0198	0.0124	0.0079
8	6	4	0.0397	0.0248	0.0158
6	5	4	0.0595	0.0372	0.0237
6	3	2	0.0794	0.0496	0.0316
3	2	1	0.0992	0.0620	0.0395
			...		
			1.0000	0.7440	0.4738
			1.0000	0.7564	0.4817
			1.0000	0.7688	0.4896
			1.0000	0.7812	0.4975

```
data = [ 10 8 7; 9 6 5; 8 6 4; 6 5 4; 6 3 2; 3 2 1];
bar3([1 2 3 5 6 7], data);

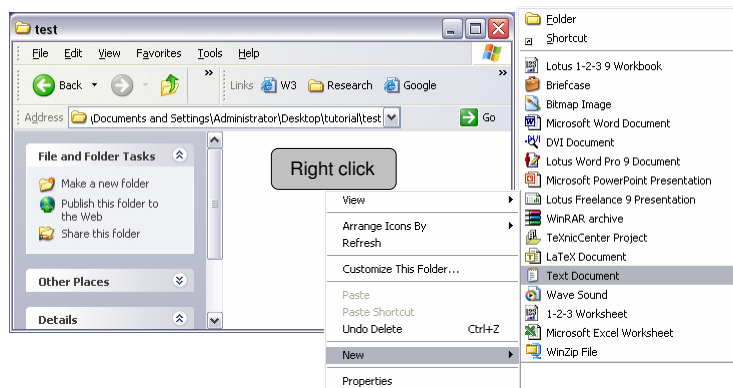
c = colormap(gray); % get colors of colormap
c = c(20:55, :); % get some colors
colormap(c); % new colormap
```

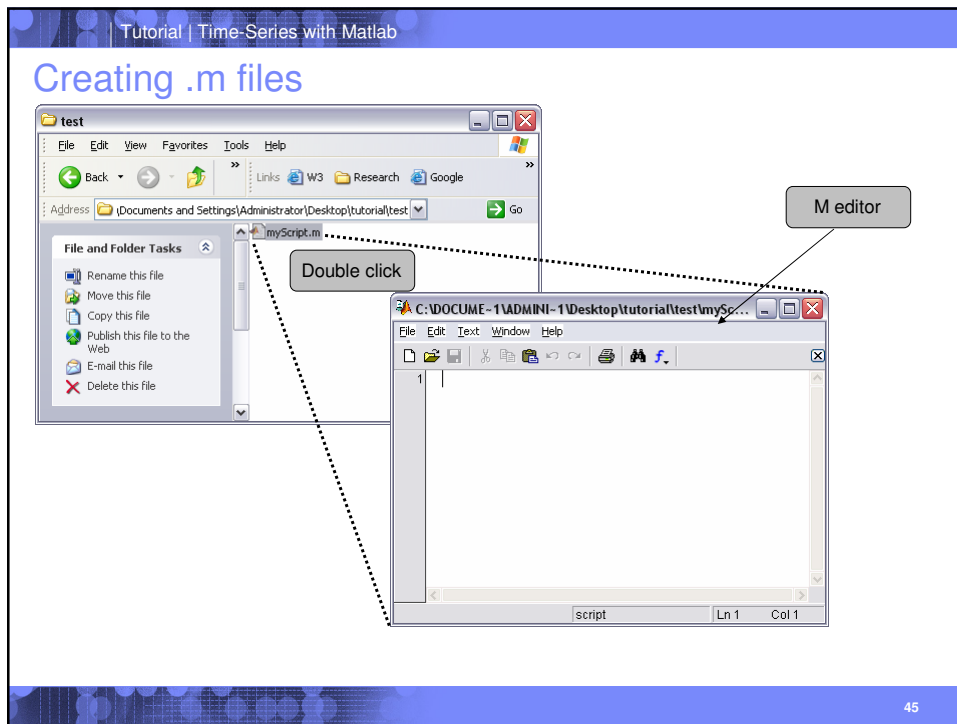
42



```
data = [1:10];
data = repmat(data,10,1); % create data
surface(data,'FaceColor',[1 1 1], 'Edgecolor',[0 0 1]); % plot data
view(3); grid on; % change viewpoint and put axis lines
```

- Script: A series of Matlab commands (no input/output arguments)
- Functions: Programs that accept input and return output





Tutorial | Time-Series with Matlab

cumsum, num2str, save

Creating .m files

- The following script will create:
 - An array with 10 random walk vectors
 - Will save them under text files: 1.dat, ..., 10.dat

myScript.m Sample Script

```

a = cumsum(randn(100,10)); % 10 random walk data of length 100
for i=1:size(a,2),          % number of columns
    data = a(:,i);
    fname = [num2str(i) '.dat']; % a string is a vector of characters!
    save(fname, 'data', '-ASCII'); % save each column in a text file
end

```

A **cumsum(A)**

1	1
2	3
3	6
4	10
5	15

A random walk time-series

Write this in the M editor...

...and execute by typing the name on the Matlab command line

46

Functions in .m scripts

- When we need to:
 - Organize our code
 - Frequently change parameters in our scripts

keyword output argument function name input argument

```
function dataN = zNorm(data)
% ZNORM zNormalization of vector
% subtract mean and divide by std

if (nargin<1), % check parameters
    error('Not enough arguments');
end
data = data - mean(data); % subtract mean
data = data/std(data); % divide by std
dataN = data;
```

Help Text
(help function_name)

Function Body

```
function [a,b] = myFunc(data, x, y) % pass & return more arguments
```

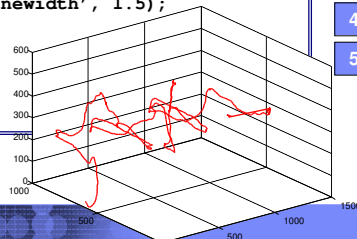
See also: varargin, varargout

47

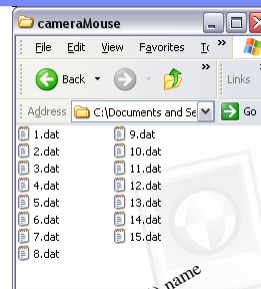
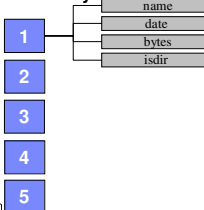
Cell Arrays

- Cells that hold other Matlab arrays
 - Let's read the files of a directory

```
>> f = dir('*.dat') % read file contents
f =
15x1 struct array with fields:
    name
    date
    bytes
    isdir
for i=1:length(f),
    a{i} = load(f(i).name);
    N = length(a{i});
    plot3([1:N], a{i}(:,1), a{i}(:,2), ...
        'r-', 'Linewidth', 1.5);
    grid on;
    pause;
    cla;
end
```



Struct Array



48

Reading/Writing Files

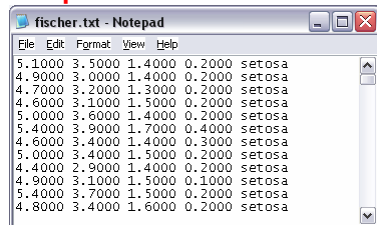
- **Load/Save are faster than C style I/O operations**

- But fscanf, fprintf can be useful for file formatting or reading non-Matlab files

```
fid = fopen('fischer.txt', 'wt');

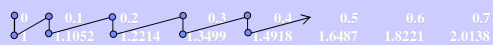
for i=1:length(species),
    fprintf(fid, '%6.4f %6.4f %6.4f %6.4f %s\n', meas(i,:), species{i});
end
fclose(fid);
```

Output file:



- **Elements are accessed column-wise (again...)**

```
x = 0:.1:1; y = [x; exp(x)];
fid = fopen('exp.txt', 'w');
fprintf(fid, '%6.2f %12.8f\n', y);
fclose(fid);
```



49

Flow Control/Loops

- **if (else/elseif) , switch**

- Check logical conditions

- **while**

- Execute statements infinite number of times

- **for**

- Execute statements a fixed number of times

- **break, continue**

- **return**

- Return execution to the invoking function



Tutorial | Time-Series with Matlab
tic, toc, clear all

For-Loop or vectorization?

```
clear all;
tic;
for i=1:50000
    a(i) = sin(i);
end
toc

clear all;
a = zeros(1,50000);
tic;
for i=1:50000
    a(i) = sin(i);
end
toc

clear all;
tic;
i = [1:50000];
a = sin(i);
toc;
```

elapsed_time =
5.0070

elapsed_time =
0.1400

elapsed_time =
0.0200

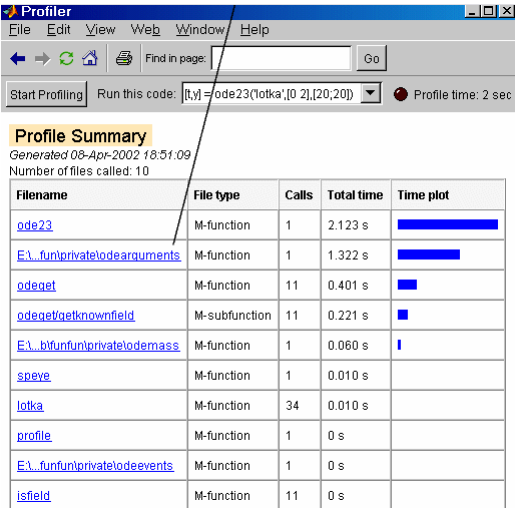
- **Pre-allocate arrays that store output results**
 - No need for Matlab to resize everytime
- **Functions are faster than scripts**
 - Compiled into pseudo-code
- **Load/Save faster than Matlab I/O functions**
- **After v. 6.5 of Matlab there is for-loop vectorization (interpreter)**
- **Vectorizations help, but not so obvious how to achieve many times**

Time not important...only life important. –The Fifth Element
51

Tutorial | Time-Series with Matlab

Matlab Profiler

- **Find which portions of code take up most of the execution time**
 - Identify bottlenecks
 - Vectorize offending code



Profile Summary
Generated 08-Apr-2002 18:51:09
Number of files called: 10

Filename	File type	Calls	Total time	Time plot
ode23	M-function	1	2.123 s	
E:_fun\private\odearguments	M-function	1	1.322 s	
odeget	M-function	11	0.401 s	
odeget\getknownfield	M-subfunction	11	0.221 s	
E:_bfunfun\private\odemass	M-function	1	0.060 s	
spevz	M-function	1	0.010 s	
lotka	M-function	34	0.010 s	
profile	M-function	1	0 s	
E:_funfun\private\odeevents	M-function	1	0 s	
isfield	M-function	11	0 s	

Time not important...only life important. –The Fifth Element
52

Hints & Tips

- **There is always an easier (and faster) way**
 - Typically there is a specialized function for what you want to achieve
- **Learn vectorization techniques, by ‘peaking’ at the actual Matlab files:**
 - edit [fname], eg
 - edit mean
 - edit princomp
- **Matlab Help contains many vectorization examples**

```

5 % the eigenvalues of the covariance matrix of X in L
6 % T-squared statistic for each data point in TSQUARE.
7
8 % Reference: J. Edward Jackson, A User's Guide to Pr:
9 % John Wiley & Sons, Inc. 1991 pp. 1-25.
10
11 % B. Jones 3-17-94
12 % Copyright 1993-2002 The MathWorks, Inc.
13 % $Revision: 2.9 $ $Date: 2002/01/17 21:31:45 $
14
15 [m,n] = size(X);
16 r = min(m-1,n); % max possible rank of x
17 avg = mean(X);
18 centerx = (X - avg(ones(m,1),:));
19
20 [U,latent,pc] = svd(centerx./sqrt(m-1),0);
21 score = centerx*pc;

```

53

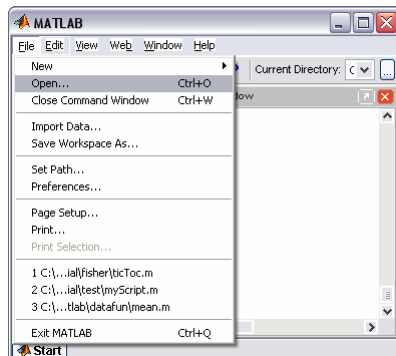
Debugging

Beware of bugs in the above code; I have only proved it correct, not tried it

-- R. Knuth

- **Not as frequently required as in C/C++**
 - Set breakpoints, step, step in, check variables values

Set breakpoints



```

64
65 %Y = fft(a,N); Y(1) = []; % sum of the data;
66 Y = fft(a); Y(1) = []; % sum of the data;
67 n = length(Y);
68 power = abs(Y(1:n/2)) .^2;
69 nyquist = 1/2;
70 freq = (1:n/2)/(n/2)*nyquist;
71 period = 1./freq;
72
73 % normalize so that sum(a.^2) = sum(abs(coefficients));
74 power = power/2; % we took only half of them
75 power = power/n; % scale so that it's no function of n
76
77 cla;
78 subplot(3,1,1);
79 plot(aOrig,'k'); axis tight;
80 title(['Query ' filename(1:end-4) '']);

```

54

Tutorial | Time-Series with Matlab

Debugging

- Full control over variables and execution path
 - F10: step, F11: step in (visit functions, as well)

A

B

C

Either this man is dead or my watch has stopped. –Groucho Marx

55

Tutorial | Time-Series with Matlab

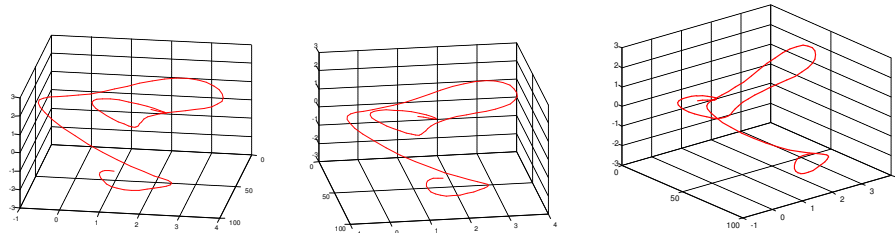
Advanced Features – 3D modeling/Volume Rendering

- Very easy volume manipulation and rendering

56

Advanced Features – Making Animations (Example)

- Create animation by changing the camera viewpoint



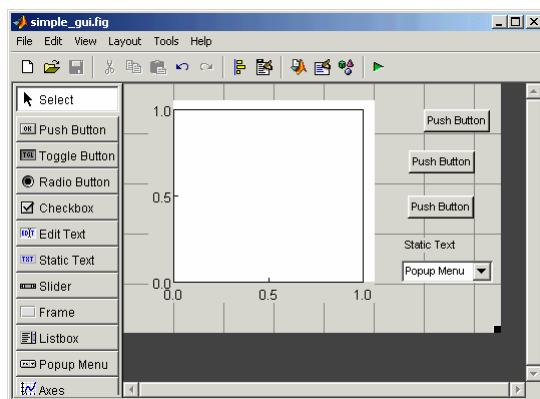
```
azimuth = [50:100 99:-1:50]; % azimuth range of values
for k = 1:length(azimuth),
    plot3(1:length(a), a(:,1), a(:,2), 'r', 'Linewidth',2);
    grid on;
    view(azimuth(k),30); % change new
    M(k) = getframe; % save the frame
end
movie(M,20); % play movie 20 times
```

See also: movie2avi

57

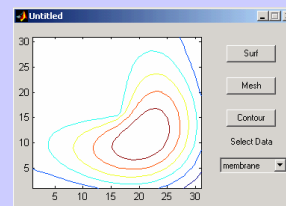
Advanced Features – GUI's

- Built-in Development Environment
 - Buttons, figures, Menus, sliders, etc



- Several Examples in Help

- Directory listing
- Address book reader
- GUI with multiple axis



58

Advanced Features – Using Java

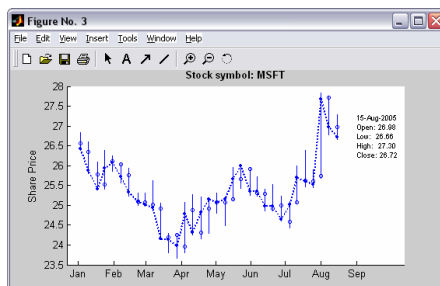
- Matlab is shipped with Java Virtual Machine (JVM)
- Access Java API (eg I/O or networking)
- Import Java classes and construct objects
- Pass data between Java objects and Matlab variables



59

Advanced Features – Using Java (Example)

- **Stock Quote Query**
 - Connect to Yahoo server
 - <http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=4069&objectType=file>



```
disp('Contacting YAHOO server using ...');
disp(['url = java.net.URL(' urlString ')']);
end;
url = java.net.URL(urlString);

try
    stream = openStream(url);
    ireader = java.io.InputStreamReader(stream);
    breader = java.io.BufferedReader(ireader);
    connect_query_data= 1; %connect made;
catch
    connect_query_data= -1; %could not connect
case;
    disp(['URL: ' urlString]);
    error(['Could not connect to server. It may
be unavailable. Try again later.']);
    stockdata={};
    return;
end
```

60

Matlab Toolboxes

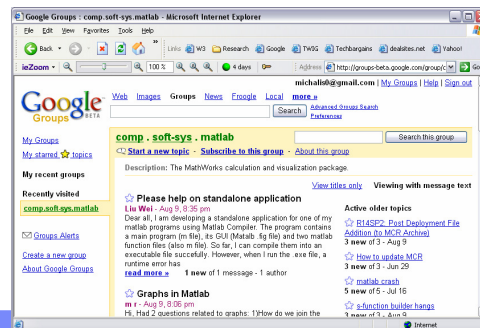
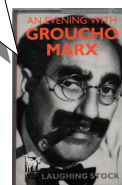
- **You can buy many specialized toolboxes from Mathworks**
 - Image Processing, Statistics, Bio-Informatics, etc
- **There are many equivalent *free* toolboxes too:**
 - SVM toolbox
 - <http://theoal.sys.uea.ac.uk/~gcc/svm/toolbox/>
 - Wavelets
 - <http://www.math.rutgers.edu/~ojanen/wavekit/>
 - Speech Processing
 - <http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html>
 - Bayesian Networks
 - <http://www.cs.ubc.ca/~murphyk/Software/BNT/bnt.html>

61

In case I get stuck...

- **help [command] (on the command line)**
eg. `help fft`
- **Menu: help -> matlab help**
 - Excellent introduction on various topics
- **Matlab webinars**
 - http://www.mathworks.com/company/events/archived_webinars.html?fp
- **Google groups**
 - comp.soft-sys.matlab
 - You can find ***anything*** here
 - Someone else had the same problem before you!

I've had a wonderful evening. But this wasn't it...



PART II: Time Series Analysis

Eight percent of
success is showing
up.

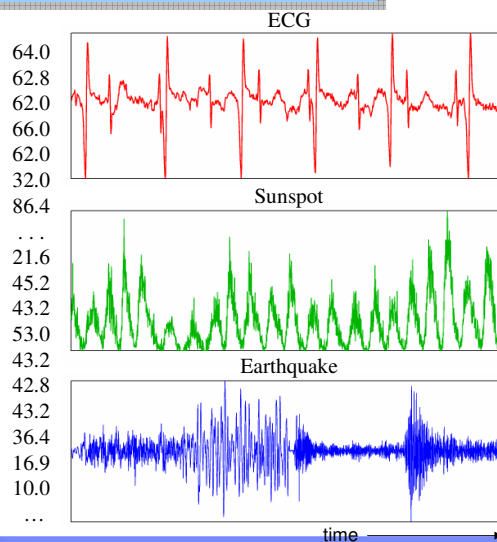


63

What is a time-series

Definition: *A sequence of measurements over time*

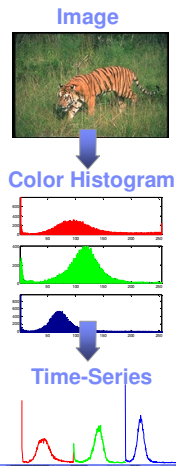
- **Medicine**
- **Stock Market**
- **Meteorology**
- **Geology**
- **Astronomy**
- **Chemistry**
- **Biometrics**
- **Robotics**



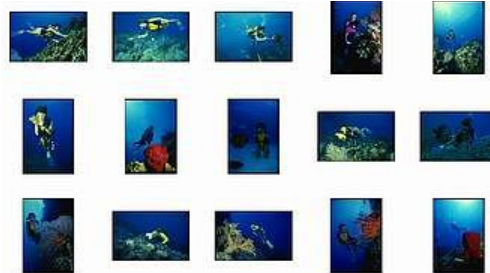
64

Applications (Image Matching)

Many types of data can be converted to time-series



Cluster 1



Cluster 2



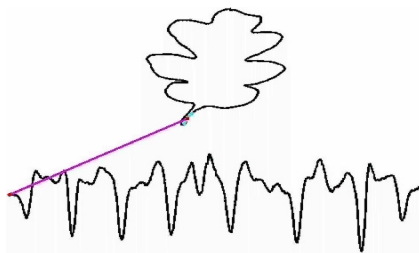
65

Applications (Shapes)

Recognize type of leaf based on its shape



Convert perimeter into a sequence of values



Special thanks to A. Ratanamahatana & E. Keogh for the leaf video.

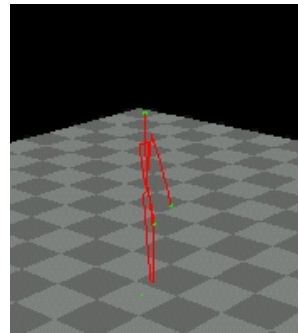
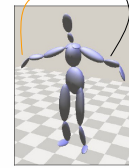
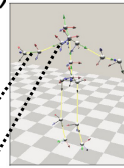
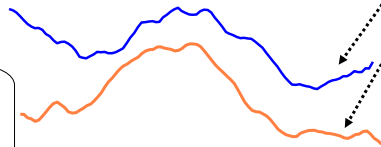
66

Applications (Motion Capture)

Motion-Capture (MOCAP) Data (Movies, Games)

- Track position of several joints over time
- 3×17 joints = **51 parameters per frame**

MOCAP data...
...my precious...



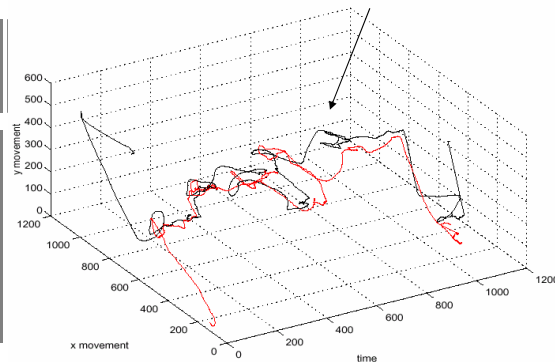
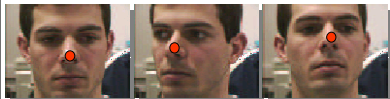
67

Applications (Video)

Video-tracking / Surveillance

- Visual tracking of body features (2D time-series)
- Sign Language recognition (3D time-series)

Video Tracking of body feature
over time (Athens1, Athens2)



68

Time-Series and Matlab

Time-series can be represented as vectors or arrays

- Fast vector manipulation
 - Most linear operations (eg [euclidean distance](#), [correlation](#)) can be trivially vectorized
- Easy visualization
- Many built-in functions
- Specialized Toolboxes

69

•PART II: Time Series Matching Introduction

Becoming sufficiently familiar with something is a substitute for understanding it.

70

Basic Data-Mining problem

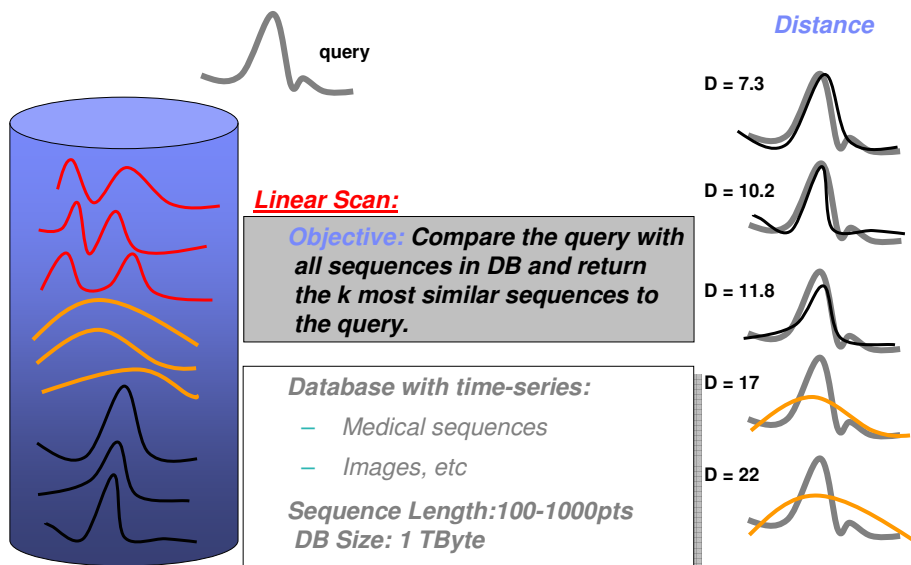
Today's databases are becoming too large. Search is difficult.
How can we overcome this obstacle?

Basic structure of data-mining solution:

- Represent data in a new format
- Search few data in the new representation
- Examine even fewer original data
- Provide guarantees about the search results
- Provide some type of data/result visualization

71

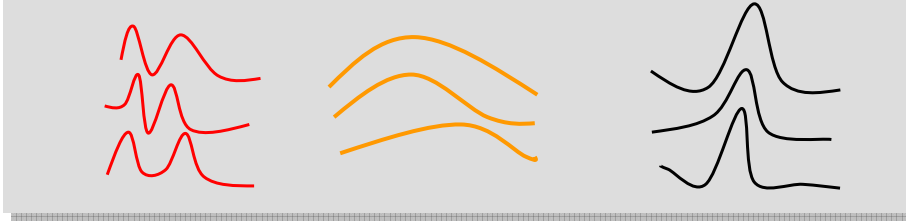
Basic Time-Series Matching Problem



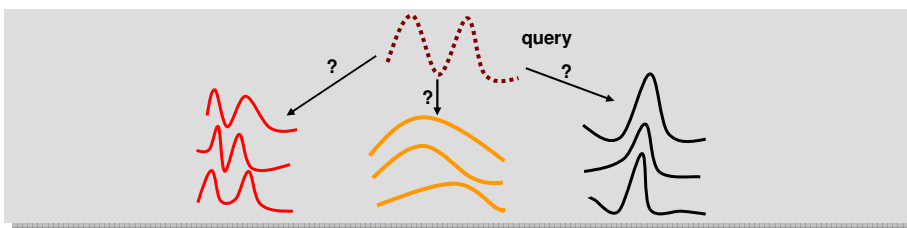
72

What other problems can we solve?

Clustering: “Place time-series into ‘similar’ groups”



Classification: “To which group is a time-series most ‘similar’ to?”

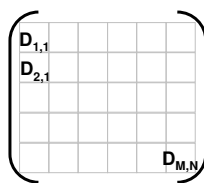


73

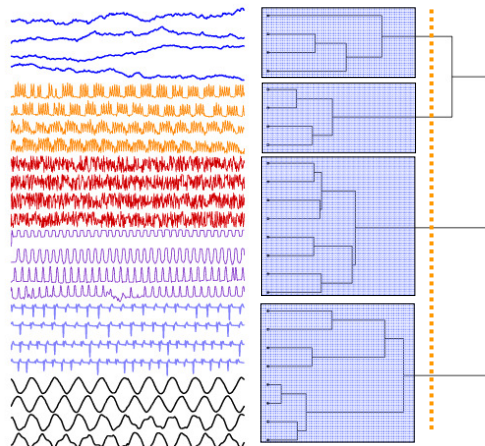
Hierarchical Clustering

- *Very generic & powerful tool*
- *Provides visual data grouping*

Pairwise
distances



1. Merge objects with smallest distance
2. Reevaluate distances
3. Repeat process



```
Z = linkage(D);  
H = dendrogram(Z);
```

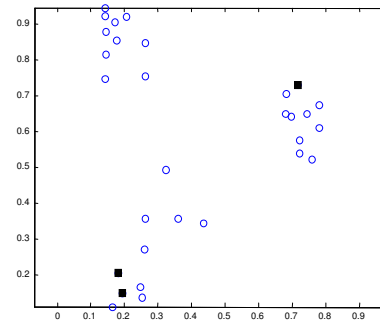
74

Partitional Clustering

- *Faster than hierarchical clustering*
- *Typically provides suboptimal solutions (local minima)*
- *Not good performance for high dimensions*

K-Means Algorithm:

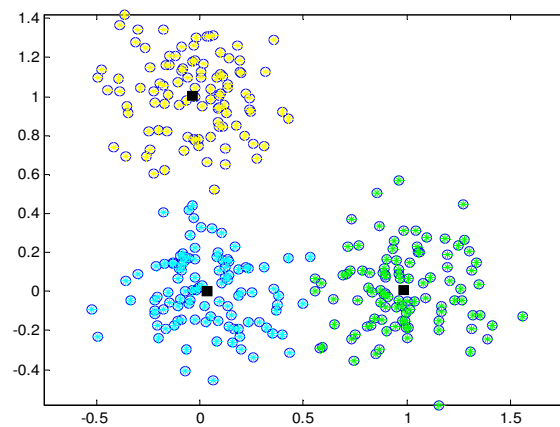
1. **Initialize k clusters (k specified by user) randomly.**
2. **Repeat until convergence**
 1. Assign each object to the nearest cluster center.
 2. Re-estimate cluster centers.



See: kmeans

75

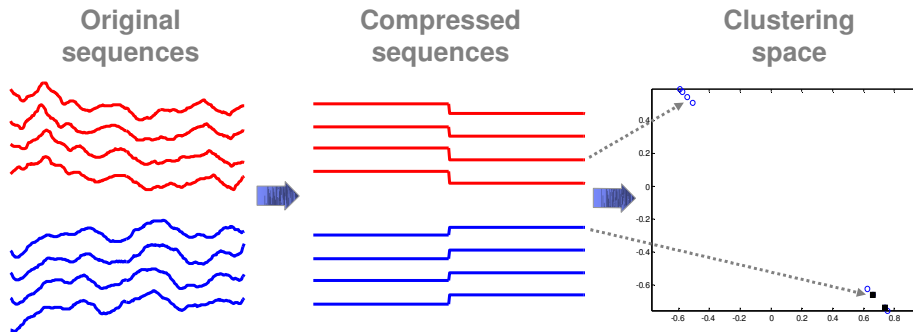
K-Means Demo



76

K-Means Clustering for Time-Series

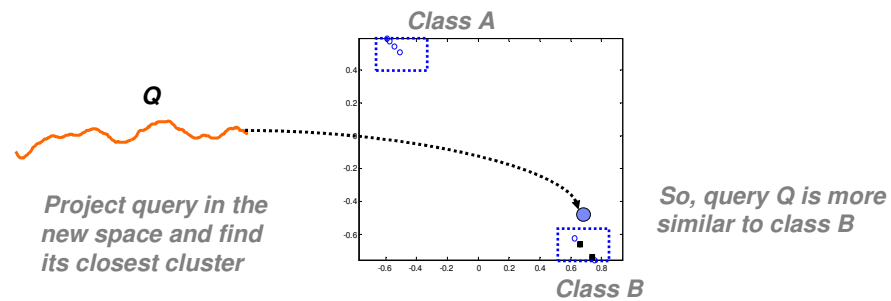
- So how is *kMeans* applied for Time-Series that are high-dimensional?
- Perform *kMeans* on a compressed dimensionality



77

Classification

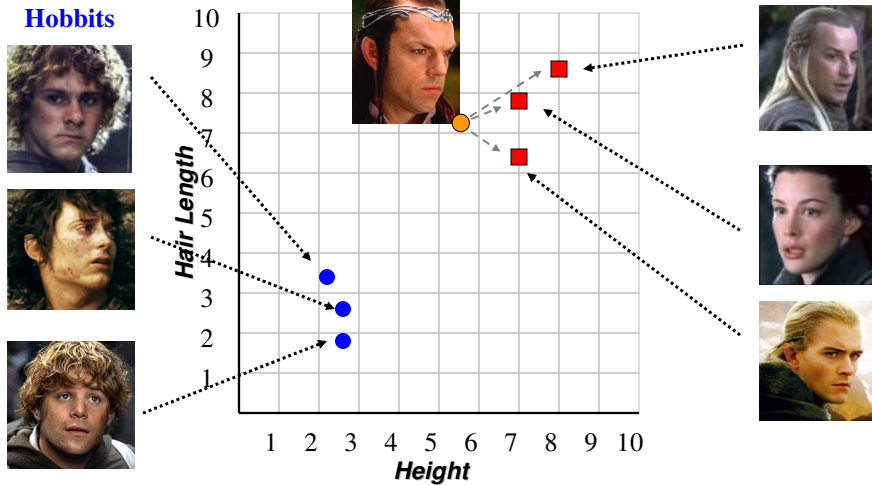
Typically classification can be made easier if we have clustered the objects



78

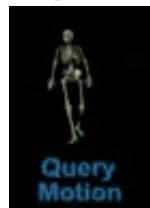
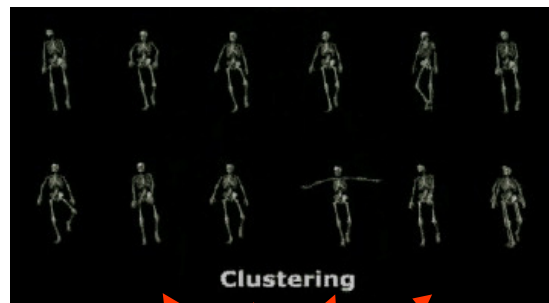
Nearest Neighbor Classification

We need not perform clustering before classification. We can classify an object based on the class majority of its nearest neighbors/matches.



79

Example



What do we need?

1. Define Similarity
2. Search fast
 - Dimensionality Reduction (compress data)

80

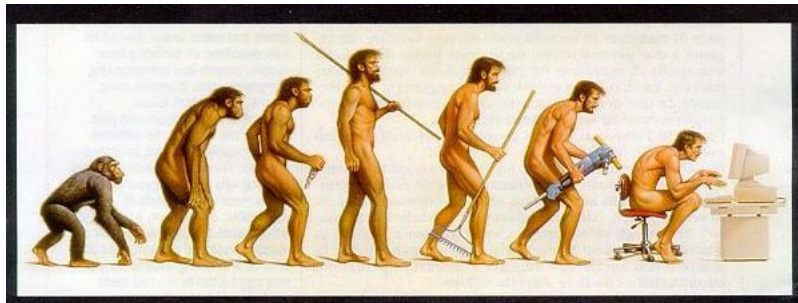
All models are wrong,
but some are useful...

•PART II: Time Series Matching Similarity/Distance functions

81

Notion of Similarity I

- *Solution to any time-series problem, boils down to a proper definition of *similarity**

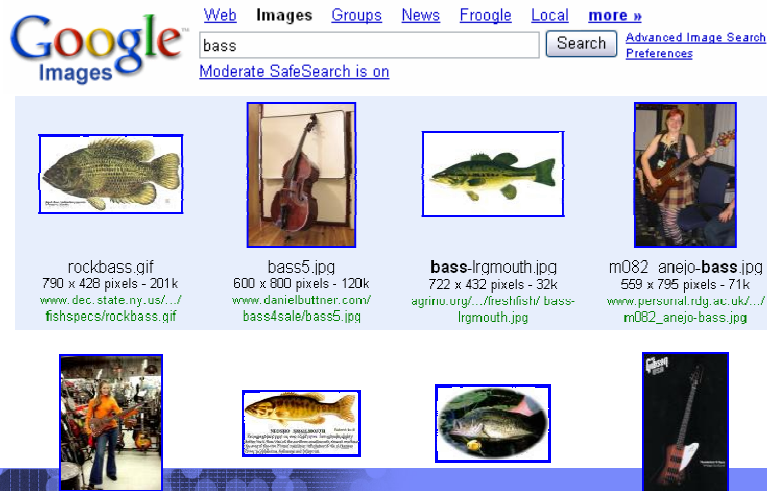


Similarity is always **subjective**.
(i.e. it depends on the application)

82

Notion of Similarity II

Similarity depends on the **features** we consider
(i.e. how we will describe or compress the sequences)



83

Metric and Non-metric Distance Functions

Distance functions

Metric

- Euclidean Distance
- Correlation

Non-Metric

- Time Warping
- LCSS

Properties

Positivity: $d(x,y) \geq 0$ and $d(x,y)=0$, if $x=y$

Symmetry: $d(x,y) = d(y,x)$

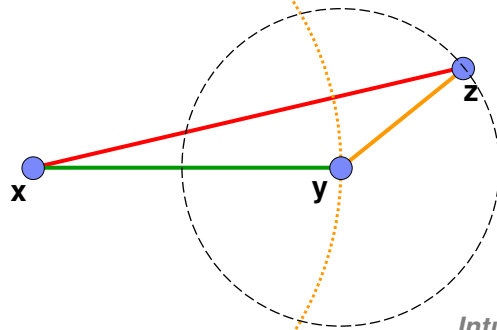
Triangle Inequality: $d(x,z) \leq d(x,y) + d(y,z)$

If **any** of these is not obeyed then the distance is a **non-metric**

84

Triangle Inequality

$$\text{Triangle Inequality: } d(x,z) \leq d(x,y) + d(y,z)$$



Metric distance functions can exploit the triangle inequality to speed-up search

Intuitively, if:

- *x is similar to y and,*
- *y is similar to z, then,*
- *x is similar to z too.*

85

Triangle Inequality (Importance)

$$\text{Triangle Inequality: } d(x,z) \leq d(x,y) + d(y,z)$$



Assume: $d(Q, \text{bestMatch}) = 20$
and $d(Q, B) = 150$

Then, since $d(A, B) = 20$

$$d(Q, A) \geq d(Q, B) - d(B, A)$$

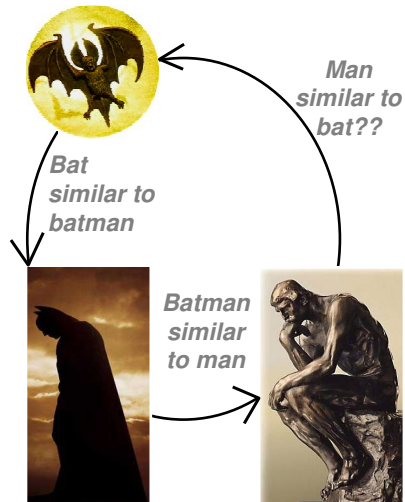
$$d(Q, A) \geq 150 - 20 = 130$$

So we don't have to retrieve A from disk

	A	B	C
A	0	20	110
B	20	0	90
C	110	90	0

86

Non-Metric Distance Functions



- **Matching flexibility**
- **Robustness to outliers**
- **Stretching in time/space**
- **Support for different sizes/lengths**



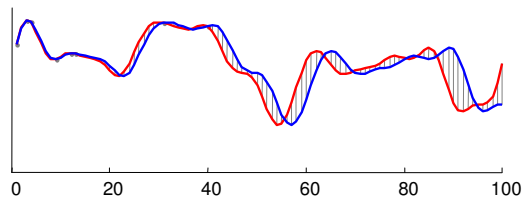
- **Speeding-up search can be difficult**

87

Euclidean Distance

- **Most widely used distance measure**

- **Definition:** $L_2 = \sqrt{\sum_{i=1}^n (a[i] - b[i])^2}$



```
L2 = sqrt(sum((a-b).^2)); % return Euclidean distance
```

88

Euclidean Distance (Vectorization)

Question: If I want to compare many sequences to each other do I have to use a for-loop?

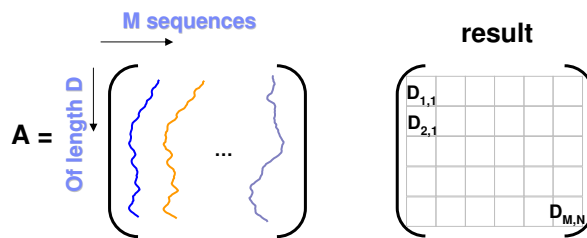
Answer: No, one can use the following equation to perform matrix computations only...

$$\|A-B\| = \sqrt{\|A\|^2 + \|B\|^2 - 2 \cdot A \cdot B}$$

A: $D \times M$ matrix

B: $D \times N$ matrix

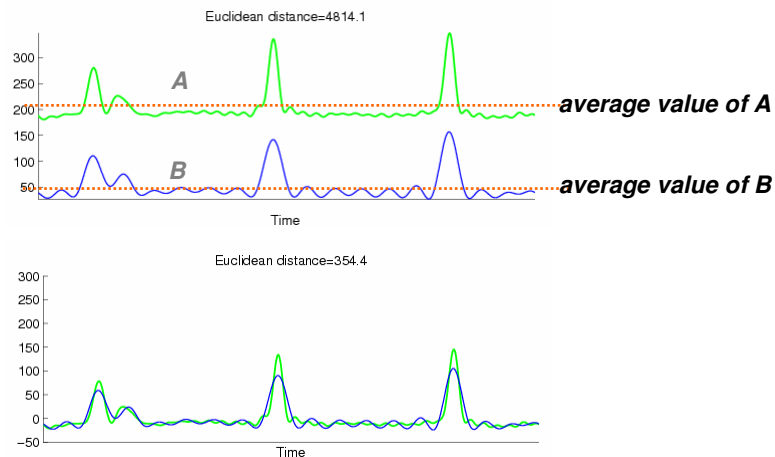
Result is $M \times N$ matrix



```
aa=sum(a.*a); bb=sum(b.*b); ab=a'*b;
d = sqrt(repmat(aa',[1 size(bb,2)]) + repmat(bb,[size(aa,2) 1]) - 2*ab);
```

89

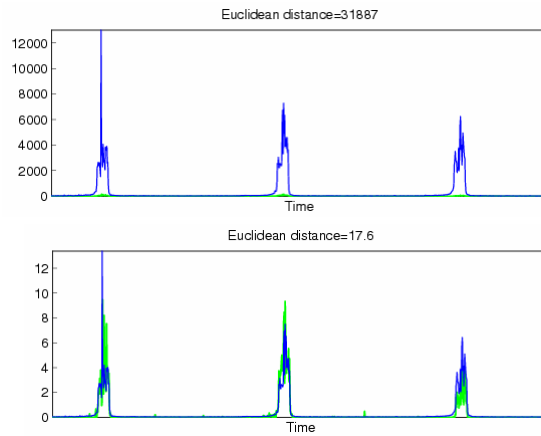
Data Preprocessing (Baseline Removal)



```
a = a - mean(a);
```

90

Data Preprocessing (Rescaling)

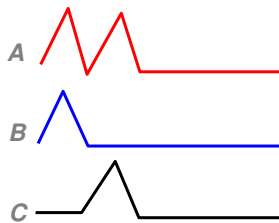


```
a = a ./ std(a);
```

91

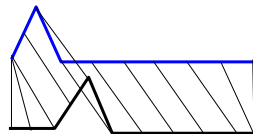
Dynamic Time-Warping (Motivation)

Euclidean distance or warping cannot compensate for small distortions in time axis.



*According to Euclidean distance
B is more similar to A than to C*

Solution: Allow for compression & decompression in time



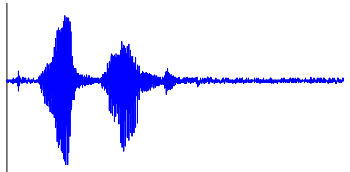
92

Dynamic Time-Warping

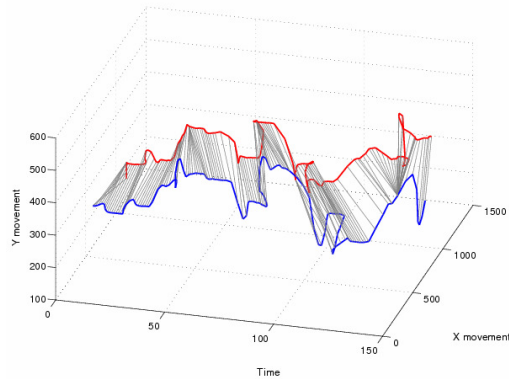
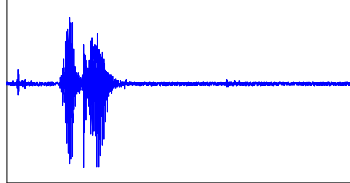
First used in speech recognition for recognizing words spoken at different speeds

Same idea can work equally well for generic time-series data

---Maat--llaabb-----



---Mat-lab-----



93

Dynamic Time-Warping (how does it work?)

The intuition is that we copy an element multiple times so as to achieve a better matching

Euclidean distance

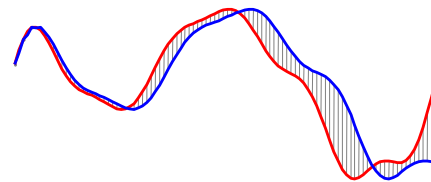
T1 = [1, 1, 2, 2]

T2 = [1, 2, 2, 2] $d = 1$

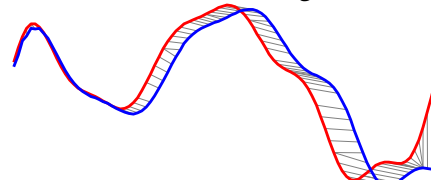
Warping distance

T1 = [1, 1, 2, 2]

T2 = [1, 2, 2, 2] $d = 0$



One-to-one linear alignment

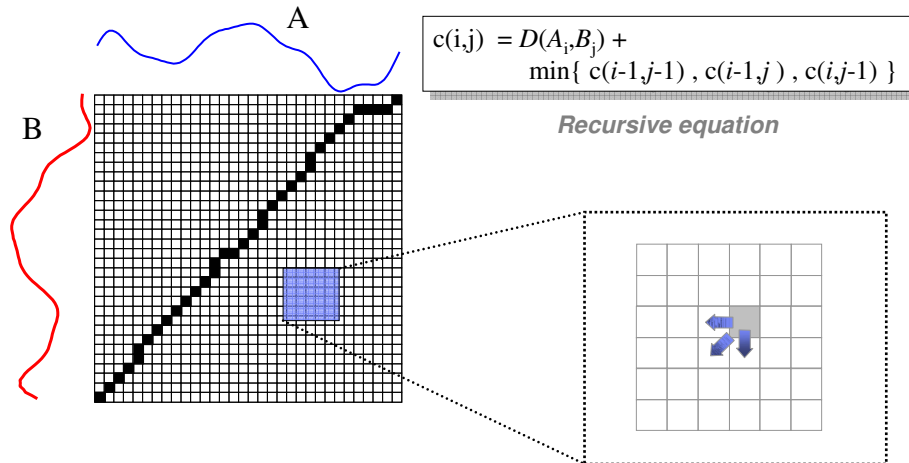


One-to-many non-linear alignment

94

Dynamic Time-Warping (implementation)

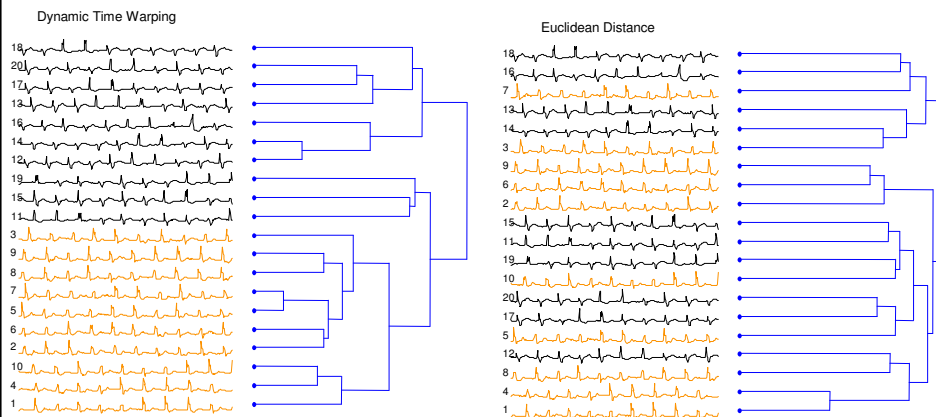
It is implemented using dynamic programming. Create an array that stores all solutions for all possible subsequences.



95

Dynamic Time-Warping (Examples)

So does it work better than Euclidean? Well yes! After all it is more costly..

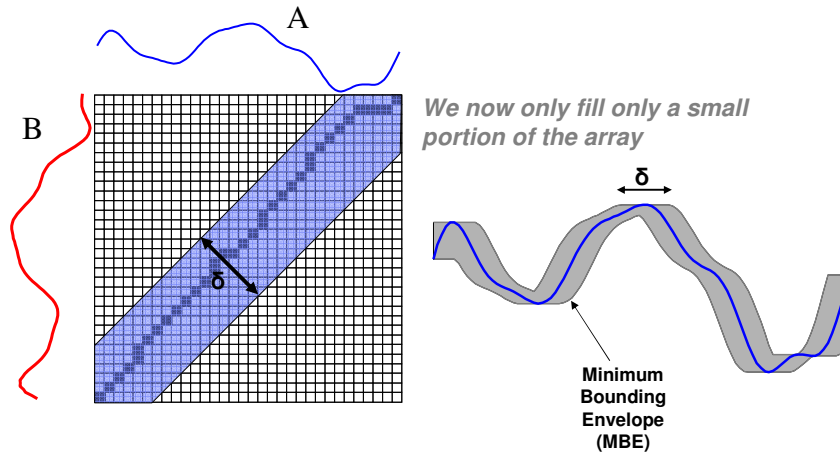


MIT arrhythmia database

96

Dynamic Time-Warping (Can we speed it up?)

Complexity is $O(n^2)$. We can reduce it to $O(\delta n)$ simply by restricting the warping path.



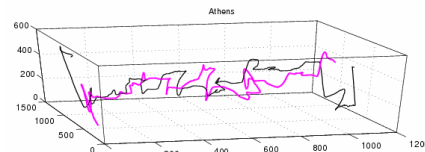
97

Dynamic Time-Warping (restricted warping)

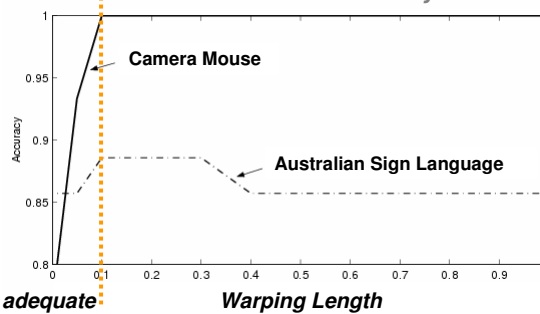
The restriction of the warping path helps:

- A. Speed-up execution
- B. Avoid extreme (degenerate) matchings
- C. Improve clustering/classification accuracy

Camera-Mouse dataset



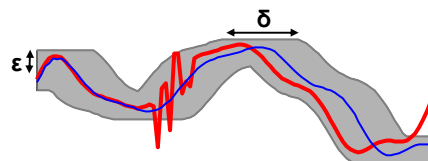
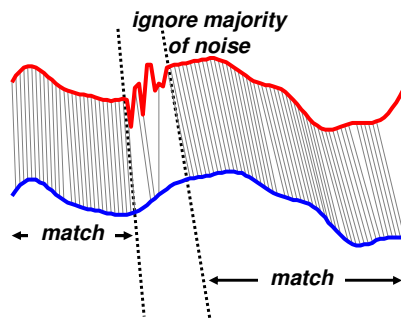
Classification Accuracy



98

Longest Common Subsequence (LCSS)

With Time Warping extreme values (outliers) can destroy the distance estimates. The LCSS model can offer more resilience to noise and impose spatial constraints too.



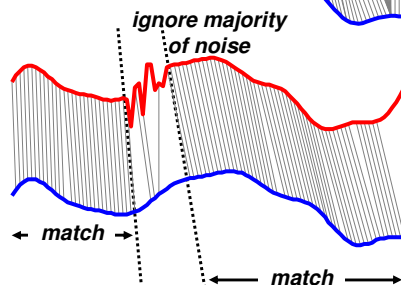
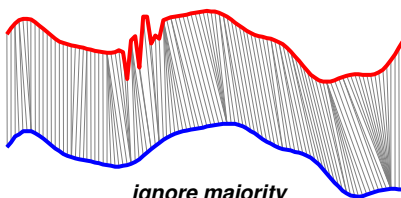
Matching within δ time and ϵ in space

Everything that is outside the bounding envelope can never be matched

99

Longest Common Subsequence (LCSS)

LCSS is more resilient to noise than DTW.



Disadvantages of DTW:

- A. All points are matched
- B. Outliers can distort distance
- C. One-to-many mapping

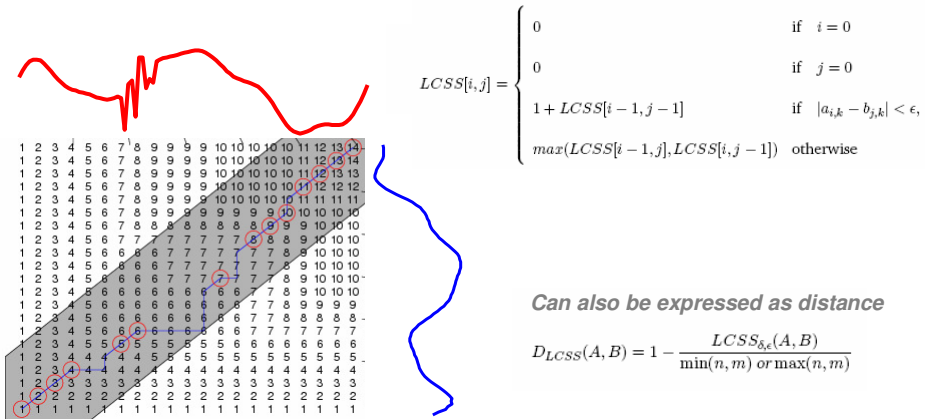
Advantages of LCSS:

- A. Outlying values not matched
- B. Distance/Similarity distorted less
- C. Constraints in time & space

100

Longest Common Subsequence (Implementation)

Similar dynamic programming solution as DTW, but now we measure similarity not distance.



101

Distance Measure Comparison

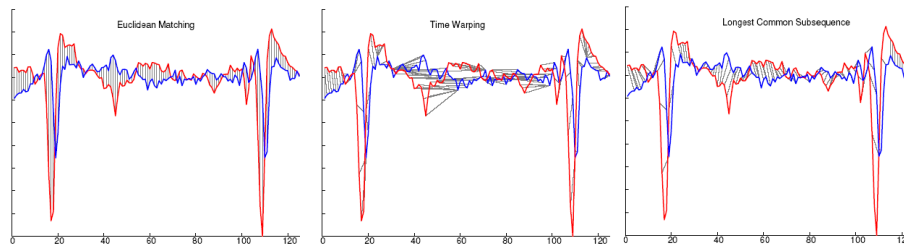
Dataset	Method	Time (sec)	Accuracy
Camera-Mouse	Euclidean	34 🍷	20%
	DTW	237	80%
	LCSS	210	100% 🍷
ASL	Euclidean	2.2 🍷	33%
	DTW	9.1	44%
	LCSS	8.2	46% 🍷
ASL+noise	Euclidean	2.1 🍷	11%
	DTW	9.3	15%
	LCSS	8.3	31% 🍷

LCSS offers enhanced robustness under noisy conditions

102

Distance Measure Comparison (Overview)

Method	Complexity	Elastic Matching	One-to-one Matching	Noise Robustness
<i>Euclidean</i>	$O(n)$	✗	✓	✗
<i>DTW</i>	$O(n*\delta)$	✓	✗	✗
<i>LCSS</i>	$O(n*\delta)$	✓	✓	✓



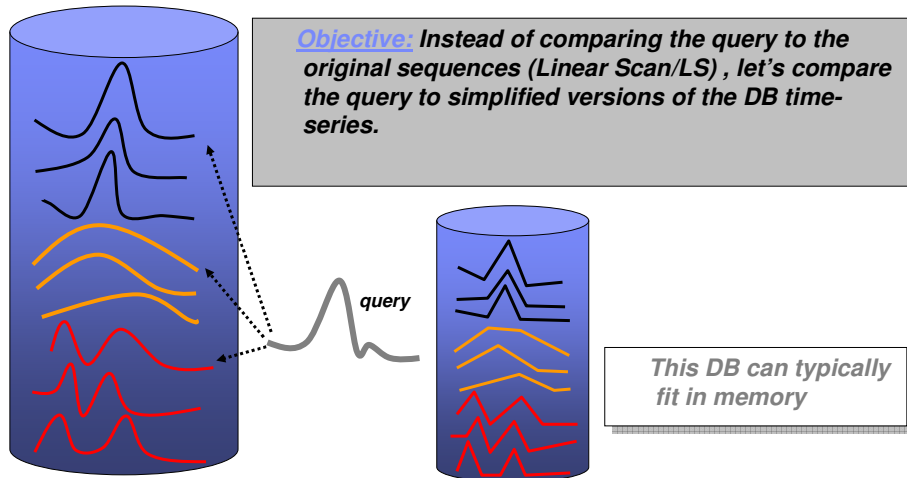
103

•PART II: Time Series Matching

Lower Bounding

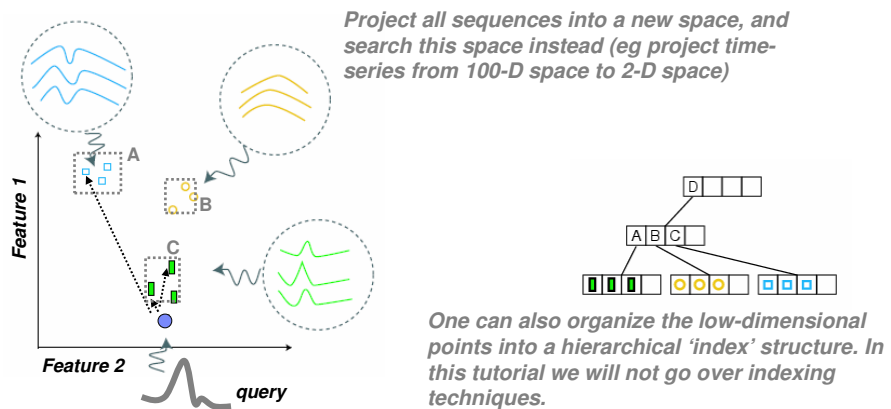
104

Basic Time-Series Problem Revisited



105

Compression – Dimensionality Reduction



Question: When searching the original space it is guaranteed that we will find the best match. Does this hold (or under which circumstances) in the new compressed space?

106

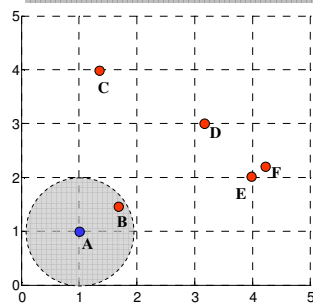
Concept of Lower Bounding

- You can guarantee similar results to Linear Scan in the original dimensionality, as long as you provide a *Lower Bounding (LB) function* (in low dim) to the original distance (high dim.)

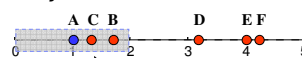
GEMINI, GEneric Multimedia INDEXing

- So, for projection from high dim. (N) to low dim. (n): $A \rightarrow a, B \rightarrow b$ etc

$$D_{LB}(a,b) \leq D_{true}(A,B)$$

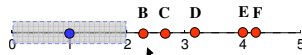


Projection onto X-axis



False alarm (not a problem)

Projection on some other axis

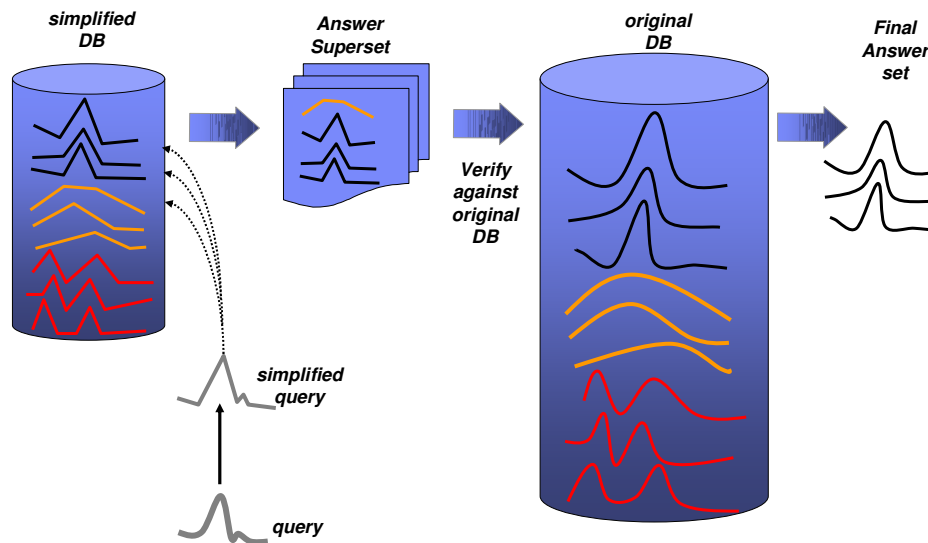


False dismissal (bad!)

“Find everything within range of 1 from A”

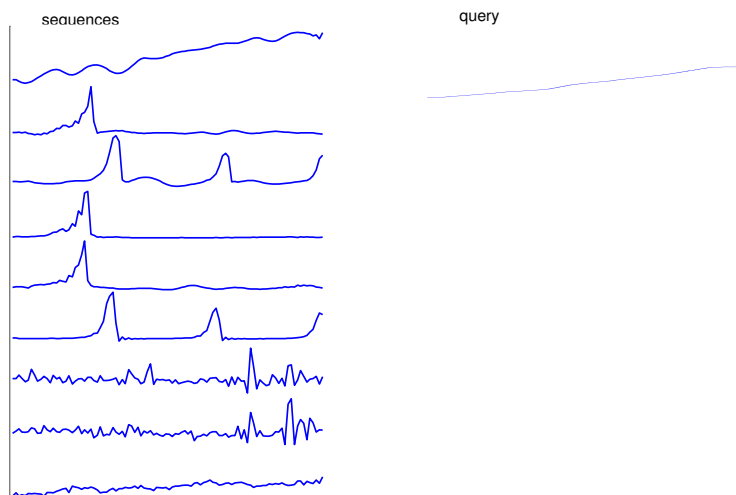
107

Generic Search using Lower Bounding



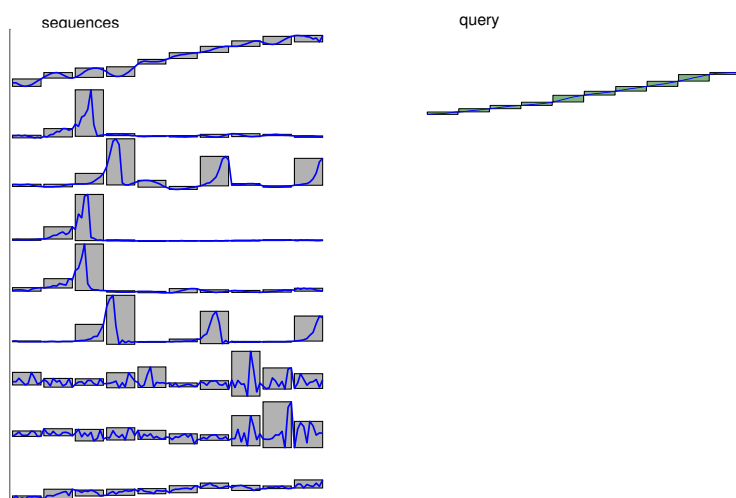
108

Lower Bounding Example



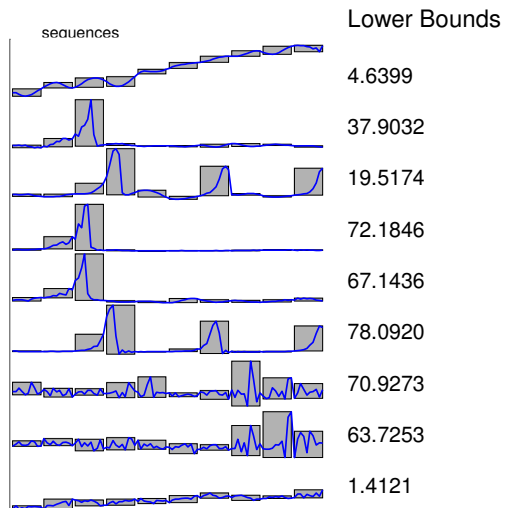
109

Lower Bounding Example



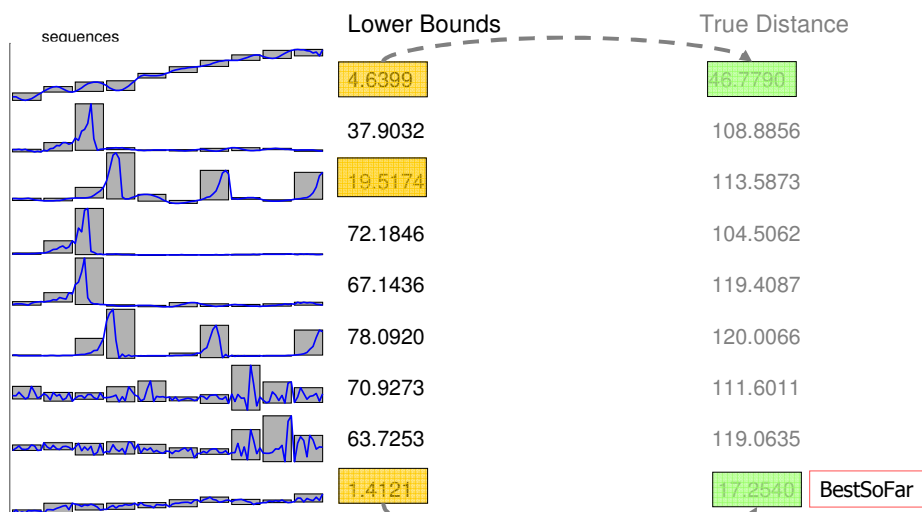
110

Lower Bounding Example



111

Lower Bounding Example



112

Lower Bounding the Euclidean distance

There are many dimensionality reduction (compression) techniques for time-series data. The following ones can be used to lower bound the Euclidean distance.

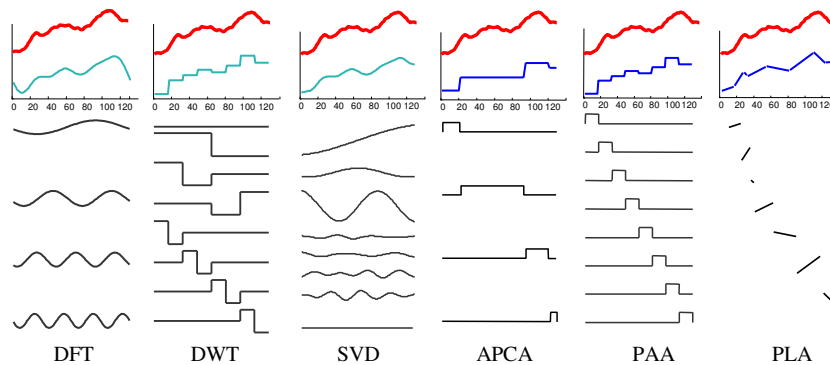


Figure by Eamonn Keogh, 'Time-Series Tutorial'

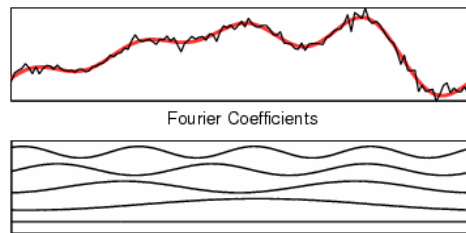
113

Fourier Decomposition

Decompose a time-series into sum of sine waves

$$\text{DFT: } X(f_{k/N}) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) e^{-j \frac{2\pi k n}{N}}, \quad k = 0, 1, \dots, N-1$$

$$\text{IDFT: } x(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X(f_{k/N}) e^{j \frac{2\pi k n}{N}}, \quad k = 0, 1, \dots, N-1$$



"Every signal can be represented as a superposition of sines and cosines" (...alas nobody believes me...)



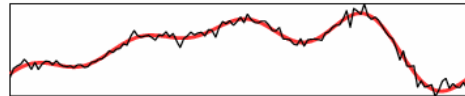
114

Fourier Decomposition

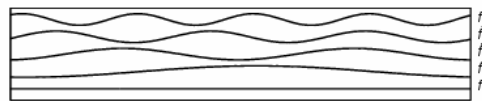
Decompose a time-series into sum of sine waves

$$\text{DFT: } X(f_{k/N}) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) e^{-j \frac{2\pi k n}{N}}, \quad k = 0, 1, \dots, N-1$$

$$\text{IDFT: } x(n) = \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} X(f_{k/N}) e^{j \frac{2\pi k n}{N}}, \quad k = 0, 1, \dots, N-1$$



Fourier Coefficients



```
fa = fft(a); % Fourier decomposition
fa(5:end) = 0; % keep first 5 coefficients (low frequencies)
reconstr = real(ifft(fa)); % reconstruct signal
```

$X(f)$ $x(n)$

-0.3633	-0.4446
-0.6280 + 0.2709i	-0.9864
-0.4929 + 0.0399i	-0.3254
-1.0143 + 0.9520i	-0.6938
0.7200 - 1.0571i	-0.1086
-0.0411 + 0.1674i	-0.3470
-0.5120 - 0.3572i	0.5849
0.9860 + 0.8043i	1.5927
-0.3680 - 0.1296i	-0.9430
-0.0517 - 0.0830i	-0.3037
-0.9158 + 0.4481i	-0.7805
1.1212 - 0.6795i	-0.1953
0.2667 + 0.1100i	-0.3037
0.2667 - 0.1100i	0.2381
1.1212 + 0.6795i	2.8389
-0.9158 - 0.4481i	-0.7046
-0.0517 + 0.0830i	-0.5529
-0.3680 + 0.1296i	-0.6721
0.9860 - 0.8043i	0.1189
-0.5120 + 0.3572i	0.2706
-0.0411 - 0.1674i	-0.0003
0.7200 + 1.0571i	1.3976
-1.0143 - 0.9520i	-0.4987
-0.4929 - 0.0399i	-0.2387
-0.6280 - 0.2709i	-0.7588

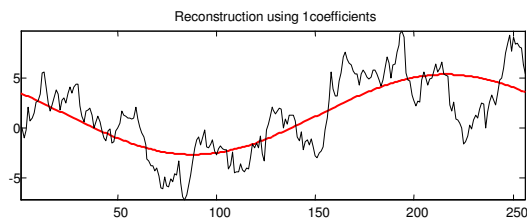


Life is complex, it has both real and imaginary parts.

115

Fourier Decomposition

How much space we gain by compressing random walk data?

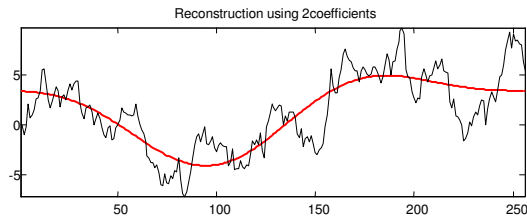


- 1 coeff > 60% of energy
- 10 coeff > 90% of energy

116

Fourier Decomposition

How much space we gain by compressing random walk data?

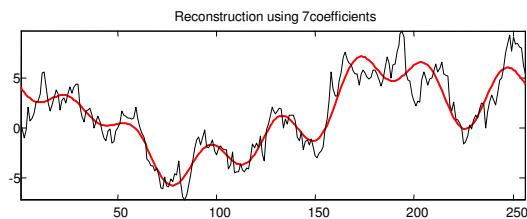


- 1 coeff > 60% of energy
- 10 coeff > 90% of energy

117

Fourier Decomposition

How much space we gain by compressing random walk data?

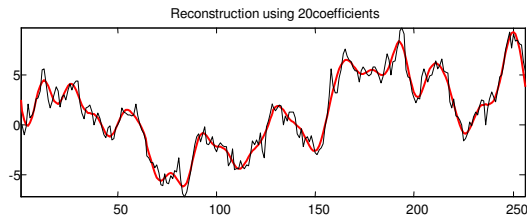


- 1 coeff > 60% of energy
- 10 coeff > 90% of energy

118

Fourier Decomposition

How much space we gain by compressing random walk data?

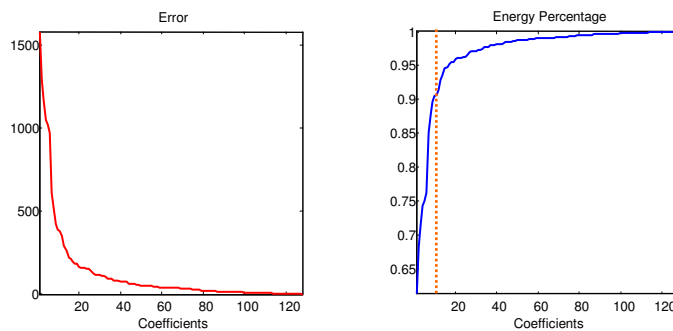


- 1 coeff > 60% of energy
- 10 coeff > 90% of energy

119

Fourier Decomposition

How much space we gain by compressing random walk data?



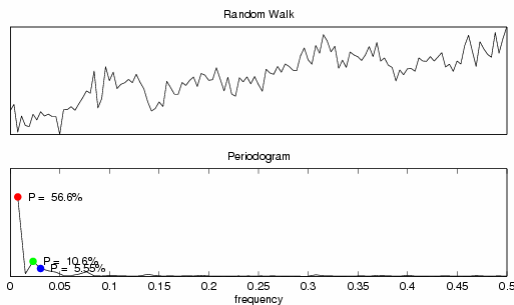
- 1 coeff > 60% of energy
- 10 coeff > 90% of energy

120

Fourier Decomposition

Which coefficients are important?

- We can measure the 'energy' of each coefficient
- $\text{Energy} = \text{Real}(X(f_k))^2 + \text{Imag}(X(f_k))^2$



```
fa = fft(a); % Fourier decomposition
N = length(a); % how many?
fa = fa(1:ceil(N/2)); % keep first half only
mag = 2*abs(fa).^2; % calculate energy
```

Most of data-mining research uses first k coefficients:

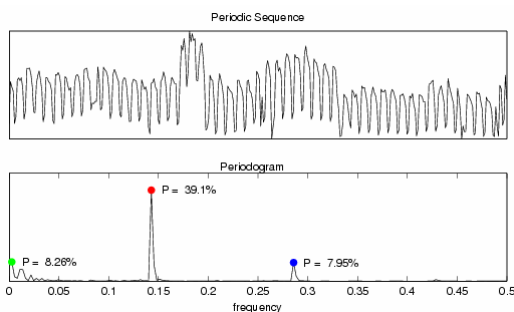
- Good for random walk signals (eg stock market)
- Easy to 'index'
- Not good for general signals

121

Fourier Decomposition

Which coefficients are important?

- We can measure the 'energy' of each coefficient
- $\text{Energy} = \text{Real}(X(f_k))^2 + \text{Imag}(X(f_k))^2$



Usage of the coefficients with highest energy:

- Good for all types of signals
- Believed to be difficult to index
- CAN be indexed using *metric trees*

122

Code for Reconstructed Sequence

```

a = load('randomWalk.dat');
a = a-mean(a)/std(a);           % z-normalization

fa = fft(a);

maxInd = ceil(length(a)/2);     % until the middle
N = length(a);

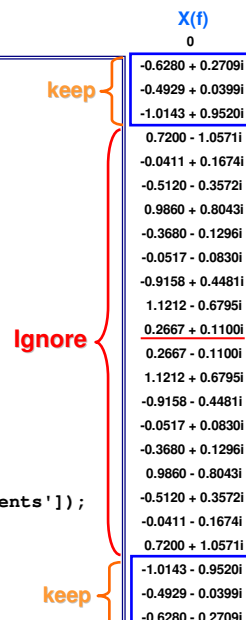
energy = zeros(maxInd-1, 1);
E = sum(a.^2);                  % energy of a

for ind=2:maxInd,

    fa_N = fa;                  % copy fourier
    fa_N(ind+1:N-ind+1) = 0;    % zero out unused
    r = real(ifft(fa_N));        % reconstruction

    plot(r, 'r', 'LineWidth', 2); hold on;
    plot(a, 'k');
    title(['Reconstruction using ' num2str(ind-1) 'coefficients']);
    set(gca, 'plotboxaspectratio', [3 1 1]);
    axis tight
    pause;                      % wait for key
    cla;                        % clear axis
end

```



123

Code for Plotting the Error

```

a = load('randomWalk.dat');
a = a-mean(a)/std(a);           % z-normalization
fa = fft(a);
maxInd = ceil(length(a)/2);     % until the middle
N = length(a);
energy = zeros(maxInd-1, 1);
E = sum(a.^2);                  % energy of a

for ind=2:maxInd,

    fa_N = fa;                  % copy fourier
    fa_N(ind+1:N-ind+1) = 0;    % zero out unused
    r = real(ifft(fa_N));        % reconstruction

    energy(ind-1) = sum(r.^2);   % energy of reconstruction
    error(ind-1) = sum(abs(r-a).^2); % error

end

E = ones(maxInd-1, 1)*E;
error = E - energy;
ratio = energy ./ E;

subplot(1,2,1);                 % left plot
plot([1:maxInd-1], error, 'r', 'LineWidth', 1.5);
subplot(1,2,2);                 % right plot
plot([1:maxInd-1], ratio, 'b', 'LineWidth', 1.5);

```

This is the same

124

Lower Bounding using Fourier coefficients

Parseval's Theorem states that energy in the frequency domain equals the energy in the time domain:

$$\sum_{t=0}^{N-1} \|x(t)\|^2 = \sum_{k=0}^{N-1} \|X(f_{k/N})\|^2$$

or, that

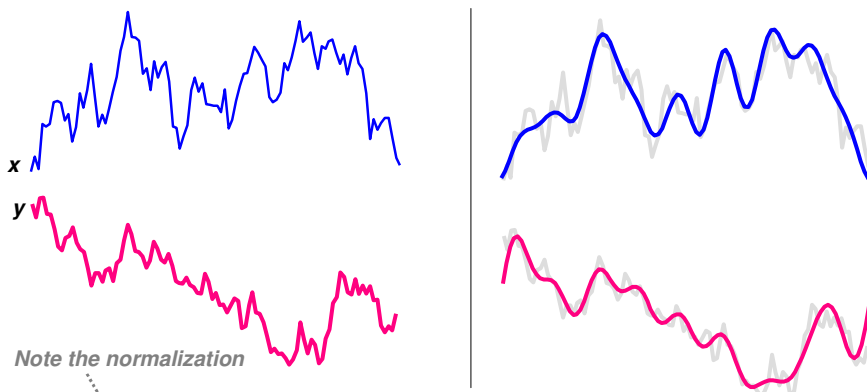
$$\sum_{t=0}^{N-1} \|x(t) - y(t)\|^2 = \sum_{k=0}^{N-1} \|X(f_{k/N}) - Y(f_{k/N})\|^2 \quad \text{Euclidean distance}$$

If we just keep some of the coefficients, their sum of squares always underestimates (ie lower bounds) the Euclidean distance:

$$\sum_{k=0}^m \|X(f_{k/N}) - Y(f_{k/N})\|^2 \leq \sum_{k=0}^{N-1} \|x(t) - y(t)\|^2, \quad m \leq N-1$$

125

Lower Bounding using Fourier coefficients -Example



```
x = cumsum(randn(100,1));
y = cumsum(randn(100,1));
euclid_Time = sqrt(sum((x-y).^2));
fx = fft(x)/sqrt(length(x));
fy = fft(y)/sqrt(length(x));
euclid_Freq = sqrt(sum(abs(fx - fy).^2));
```

120.9051

120.9051

Keeping 10 coefficients
the distance is:
115.5556 < 120.9051

126

Fourier Decomposition



- $O(n \log n)$ complexity
- Tried and tested
- Hardware implementations
- Many applications:
 - compression
 - smoothing
 - periodicity detection

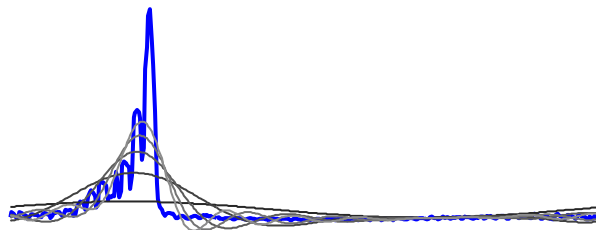


- Not good approximation for *bursty* signals
- Not good approximation for signals with flat and busy sections
(requires many coefficients)

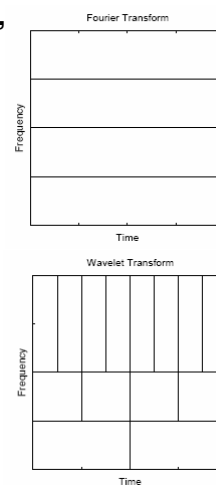
127

Wavelets – Why exist?

- Similar concept with Fourier decomposition
- Fourier coefficients represent global contributions, wavelets are localized



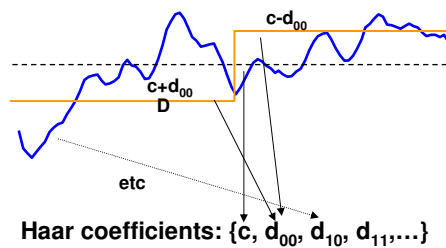
*Fourier is good for smooth, random walk data,
but not for **bursty** data or **flat** data*



128

Wavelets (Haar) - Intuition

- Wavelet coefficients, still represent an inner product (projection) of the signal with some basis functions.
- These functions have lengths that are powers of two (full sequence length, half, quarter etc)



An arithmetic example

$X = [9, 7, 3, 5]$

$\text{Haar} = [6, 2, 1, -1]$

$$c = 6 = (9+7+3+5)/4$$

$$c + d_{00} = 6+2 = 8 = (9+7)/2$$

$$c - d_{00} = 6-2 = 4 = (3+5)/2$$

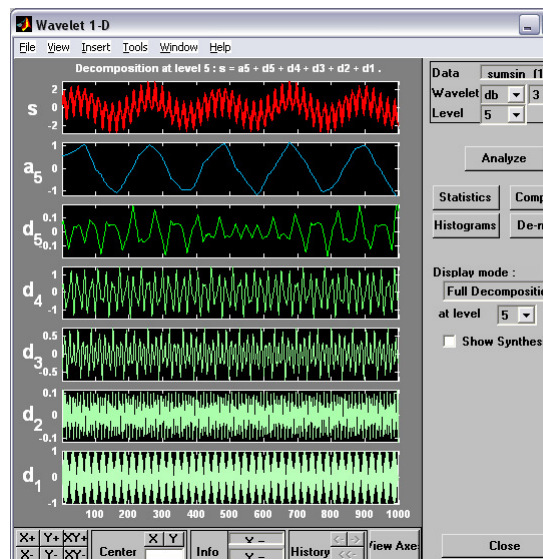
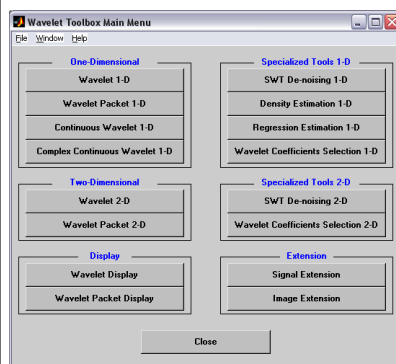
etc

See also:wavemenu

129

Wavelets in Matlab

*Specialized Matlab interface
for wavelets*

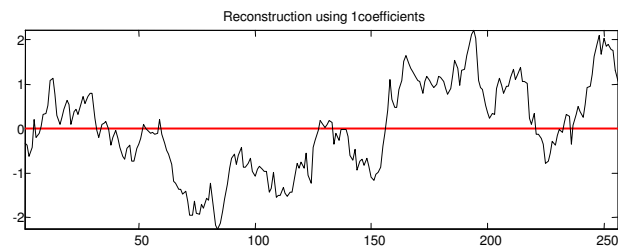


See also:wavemenu

130

PAA (Piecewise Aggregate Approximation) also featured as Piecewise Constant Approximation

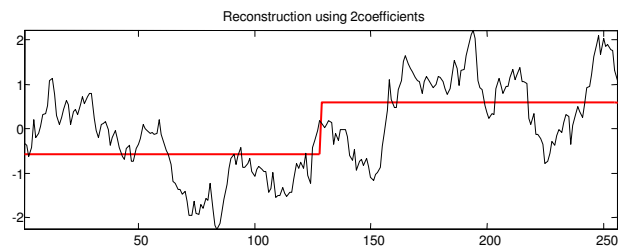
- Represent time-series as a sequence of segments
- Essentially a projection of the Haar coefficients in time



131

PAA (Piecewise Aggregate Approximation) also featured as Piecewise Constant Approximation

- Represent time-series as a sequence of segments
- Essentially a projection of the Haar coefficients in time

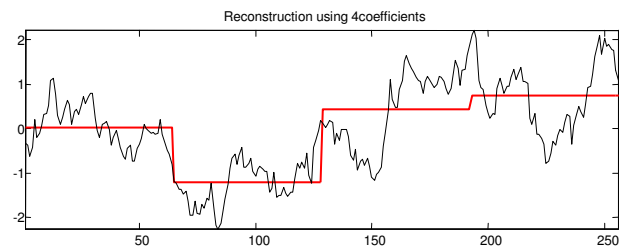


132

PAA (Piecewise Aggregate Approximation)

also featured as **Piecewise Constant Approximation**

- Represent time-series as a sequence of segments
- Essentially a projection of the Haar coefficients in time

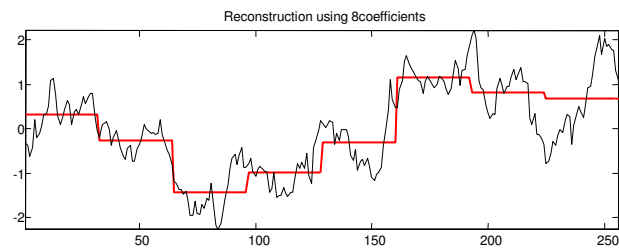


133

PAA (Piecewise Aggregate Approximation)

also featured as **Piecewise Constant Approximation**

- Represent time-series as a sequence of segments
- Essentially a projection of the Haar coefficients in time

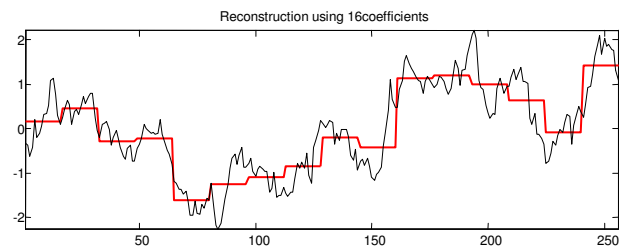


134

PAA (Piecewise Aggregate Approximation)

also featured as **Piecewise Constant Approximation**

- Represent time-series as a sequence of segments
- Essentially a projection of the Haar coefficients in time

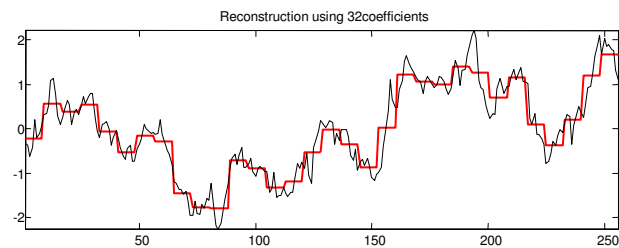


135

PAA (Piecewise Aggregate Approximation)

also featured as **Piecewise Constant Approximation**

- Represent time-series as a sequence of segments
- Essentially a projection of the Haar coefficients in time



136

PAA Matlab Code

```
function data = paa(s, numCoeff)
% PAA(s, numcoeff)
% s: sequence vector (Nx1 or Nx1)
% numCoeff: number of PAA segments
% data: PAA sequence (Nx1)

N = length(s);           % length of sequence
segLen = N/numCoeff;      % assume it's integer

sN = reshape(s, segLen, numCoeff); % break in segments
avg = mean(sN);           % average segments
data = repmat(avg, segLen, 1);    % expand segments
data = data(:);           % make column
```

s 1 2 3 4 5 6 7 8 **numCoeff** 4

137

PAA Matlab Code

```
function data = paa(s, numCoeff)
% PAA(s, numcoeff)
% s: sequence vector (Nx1 or Nx1)
% numCoeff: number of PAA segments
% data: PAA sequence (Nx1)

N = length(s);           % length of sequence
segLen = N/numCoeff;      % assume it's integer

sN = reshape(s, segLen, numCoeff); % break in segments
avg = mean(sN);           % average segments
data = repmat(avg, segLen, 1);    % expand segments
data = data(:);           % make column
```

N=8
segLen = 2

s 1 2 3 4 5 6 7 8 **numCoeff** 4

138

PAA Matlab Code

```
function data = paa(s, numCoeff)
% PAA(s, numcoeff)
% s: sequence vector (Nx1 or Nx1)
% numCoeff: number of PAA segments
% data: PAA sequence (Nx1)

N = length(s);           % length of sequence
segLen = N/numCoeff;      % assume it's integer
sN = reshape(s, segLen, numCoeff); % break in segments
avg = mean(sN);           % average segments
data = repmat(avg, segLen, 1); % expand segments
data = data(:);           % make column
```

N=8
segLen = 2

s 1 2 3 4 5 6 7 8 numCoeff 4

sN 1 3 5 7
2 4 6 8

139

PAA Matlab Code

```
function data = paa(s, numCoeff)
% PAA(s, numcoeff)
% s: sequence vector (Nx1 or Nx1)
% numCoeff: number of PAA segments
% data: PAA sequence (Nx1)

N = length(s);           % length of sequence
segLen = N/numCoeff;      % assume it's integer
sN = reshape(s, segLen, numCoeff); % break in segments
avg = mean(sN);           % average segments
data = repmat(avg, segLen, 1); % expand segments
data = data(:);           % make column
```

N=8
segLen = 2

s 1 2 3 4 5 6 7 8 numCoeff 4

sN 1 3 5 7
2 4 6 8

avg 1.5 3.5 5.5 7.5

140

PAA Matlab Code

```
function data = paa(s, numCoeff)
% PAA(s, numcoeff)
% s: sequence vector (1xN)
% numCoeff: number of PAA segments
% data: PAA sequence (1xN)

N = length(s);           % length of sequence
segLen = N/numCoeff;      % assume it's integer

sN = reshape(s, segLen, numCoeff); % break in segments
avg = mean(sN);           % average segments
data = repmat(avg, segLen, 1);    % expand segments
data = data(:)';          % make row
```

N=8
segLen = 2

s 1 2 3 4 5 6 7 8 numCoeff 4

sN 1 3 5 7
 2 4 6 8

data 1.5 3.5 5.5 7.5
 1.5 3.5 5.5 7.5

avg 1.5 3.5 5.5 7.5

141

PAA Matlab Code

```
function data = paa(s, numCoeff)
% PAA(s, numcoeff)
% s: sequence vector (1xN)
% numCoeff: number of PAA segments
% data: PAA sequence (1xN)

N = length(s);           % length of sequence
segLen = N/numCoeff;      % assume it's integer

sN = reshape(s, segLen, numCoeff); % break in segments
avg = mean(sN);           % average segments
data = repmat(avg, segLen, 1);    % expand segments
data = data(:)';          % make row
```

N=8
segLen = 2

s 1 2 3 4 5 6 7 8 numCoeff 4

sN 1 3 5 7
 2 4 6 8

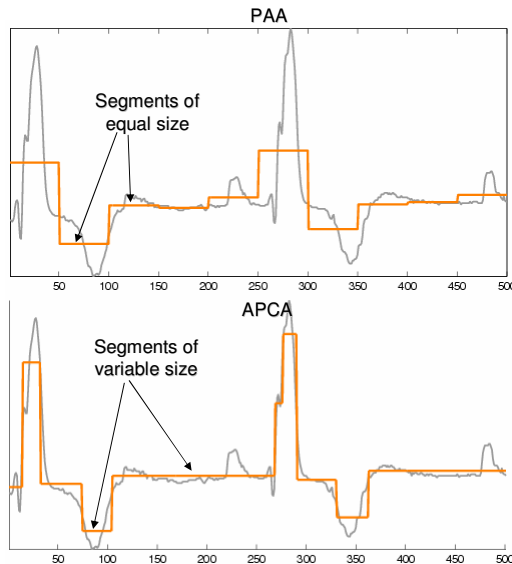
data 1.5 3.5 5.5 7.5
 1.5 3.5 5.5 7.5

avg 1.5 3.5 5.5 7.5

data 1.5 1.5 3.5 3.5 5.5 5.5 7.5 7.5

142

APCA (Adaptive Piecewise Constant Approximation)



- Not all haar/PAA coefficients are equally important
- Intuition: Keep ones with the highest energy
- Segments of variable length

- APCA is good for bursty signals
- PAA requires **1 number** per segment, APCA requires **2: [value, length]**

E.g. 10 bits for a sequence of 1024 points

143

Wavelet Decomposition



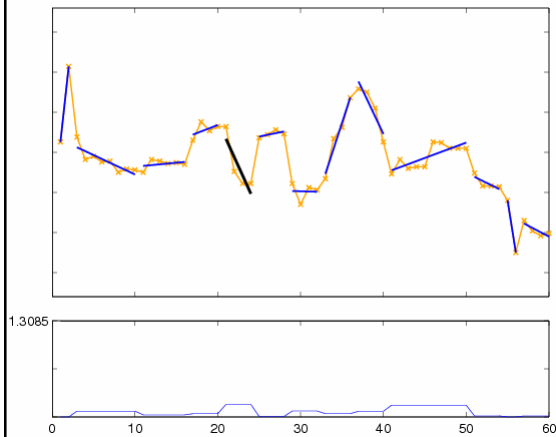
- $O(n)$ complexity
- Hierarchical structure
- Progressive transmission
- Better localization
- Good for bursty signals
- Many applications:
 - compression
 - periodicity detection



- Most data-mining research still utilizes Haar wavelets because of their simplicity.

144

Piecewise Linear Approximation (PLA)



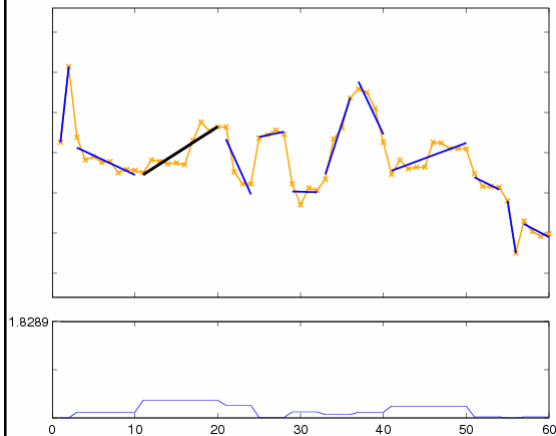
- Approximate a sequence with multiple linear segments
- First such algorithms appeared in *cartography* for map approximation
- Many implementations
 - Optimal
 - Greedy Bottom-Up
 - Greedy Top-down
 - Genetic, etc

- You can find a **bottom-up** implementation here:

— http://www.cs.ucr.edu/~eamonn/TSDMA/time_series_toolbox/

145

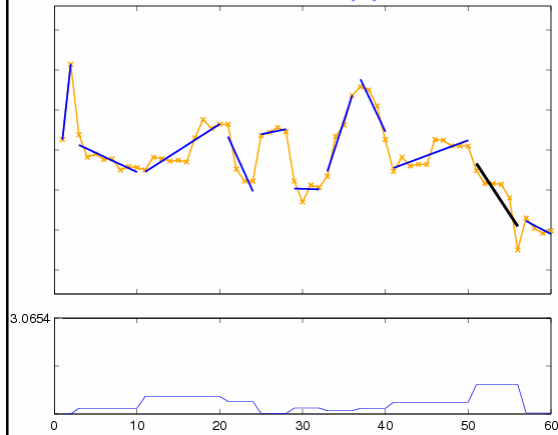
Piecewise Linear Approximation (PLA)



- Approximate a sequence with multiple linear segments
- First such algorithms appeared in *cartography* for map approximation

146

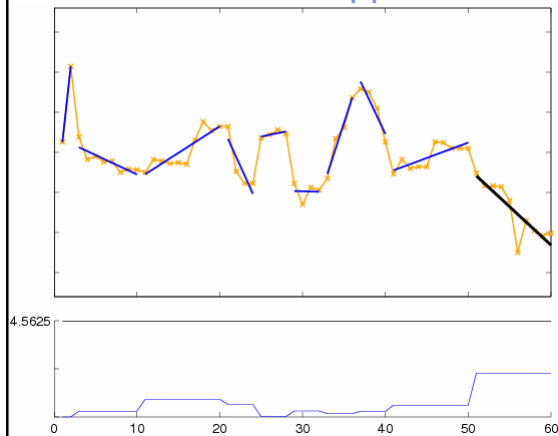
Piecewise Linear Approximation (PLA)



- Approximate a sequence with multiple linear segments
- First such algorithms appeared in *cartography* for map approximation

147

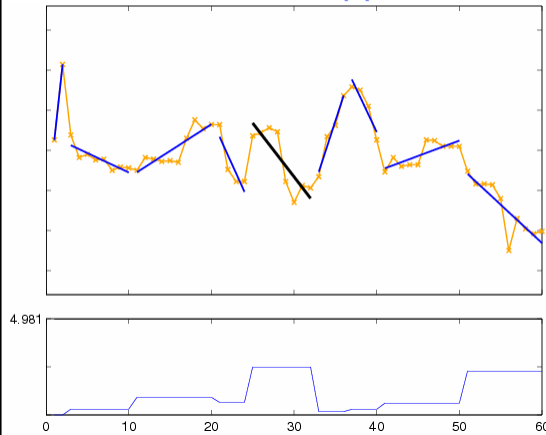
Piecewise Linear Approximation (PLA)



- Approximate a sequence with multiple linear segments
- First such algorithms appeared in *cartography* for map approximation

148

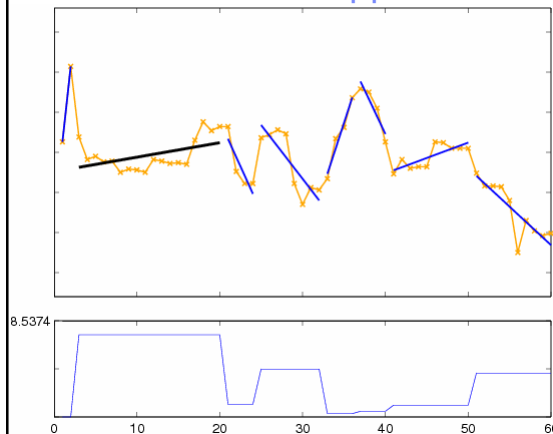
Piecewise Linear Approximation (PLA)



- Approximate a sequence with multiple linear segments
- First such algorithms appeared in *cartography* for map approximation

149

Piecewise Linear Approximation (PLA)



- Approximate a sequence with multiple linear segments
- First such algorithms appeared in *cartography* for map approximation

150

Piecewise Linear Approximation (PLA)



- $O(n \log n)$ complexity for “bottom up” algorithm
- Incremental computation possible
- Provable error bounds
- Applications for:
 - Image / signal simplification
 - Trend detection

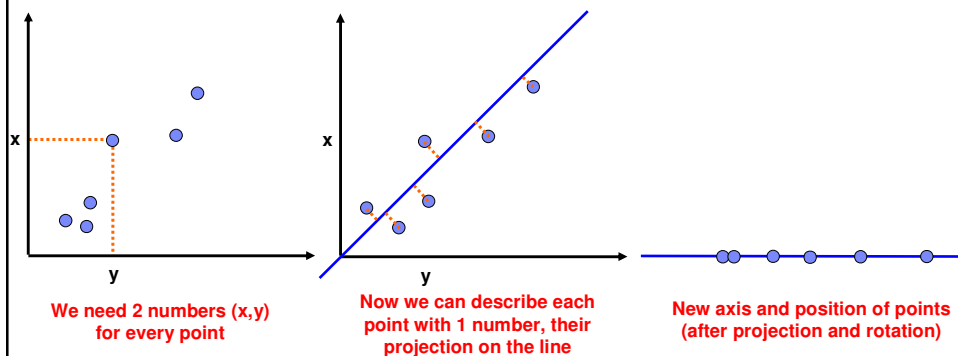


- Visually not very smooth or pleasing.

151

Singular Value Decomposition (SVD)

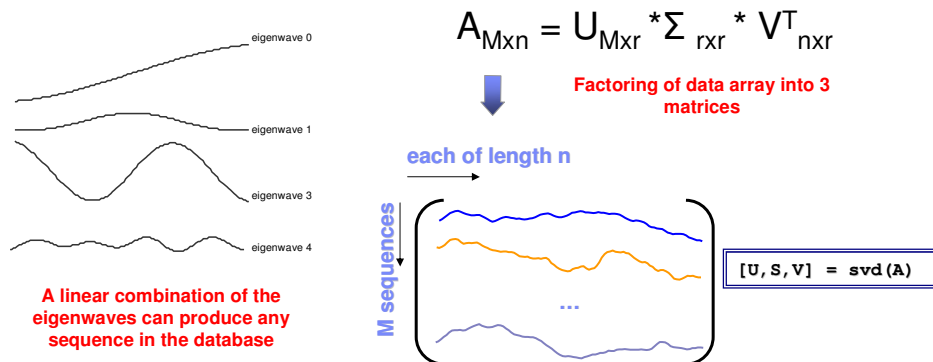
- SVD attempts to find the ‘optimal’ basis for describing a set of multidimensional points
- Objective: Find the axis (‘directions’) that describe better the data variance



152

Singular Value Decomposition (SVD)

- Each time-series is essentially a multidimensional point
- Objective: Find the 'eigenwaves' (basis) whose linear combination describes best the sequences. Eigenwaves are data-dependent.



153

Singular Value Decomposition



- Optimal dimensionality reduction in Euclidean distance sense
- SVD is a very powerful tool in many domains:
 - Websearch (PageRank)

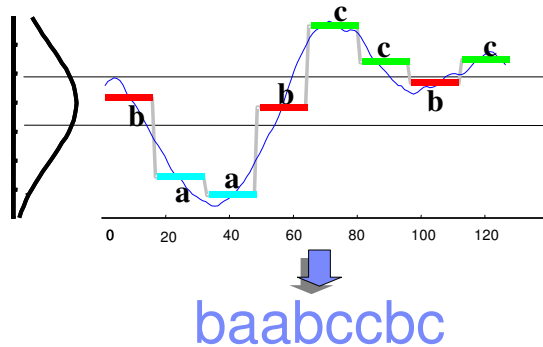


- Cannot be applied for just one sequence. A set of sequences is required.
- Addition of a sequence in database requires recomputation
- Very costly to compute.
Time: $\min\{O(M^2n), O(Mn^2)\}$
Space: $O(Mn)$
M sequences of length n

154

Symbolic Approximation

- Assign a different symbol based on range of values
- Find ranges either from data histogram or uniformly



- You can find an implementation here:
 - <http://www.ise.gmu.edu/~jessica/sax.htm>

155

Symbolic Approximations



- Linear complexity
- After 'symbolization' many tools from bioinformatics can be used
 - Markov models
 - Suffix-Trees, etc

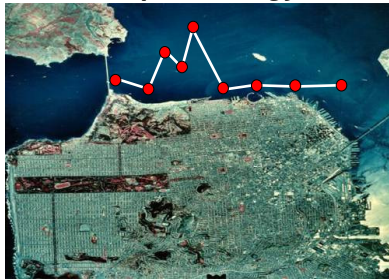


- Number of regions (alphabet length) can affect the quality of result

156

Multidimensional Time-Series

- Catching momentum lately
- Applications for *mobile trajectories, sensor networks, epidemiology, etc*



- Let's see how to approximate 2D trajectories with Minimum Bounding Rectangles

Ari, are you sure the world is not 1D?

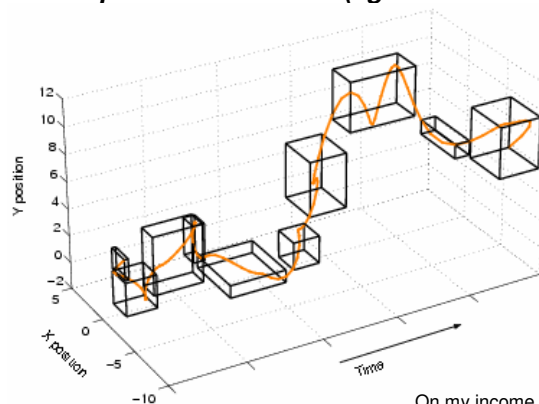


Aristotle

157

Multidimensional MBRs

Find Bounding rectangles that completely contain a trajectory given some optimization criteria (eg minimize volume)

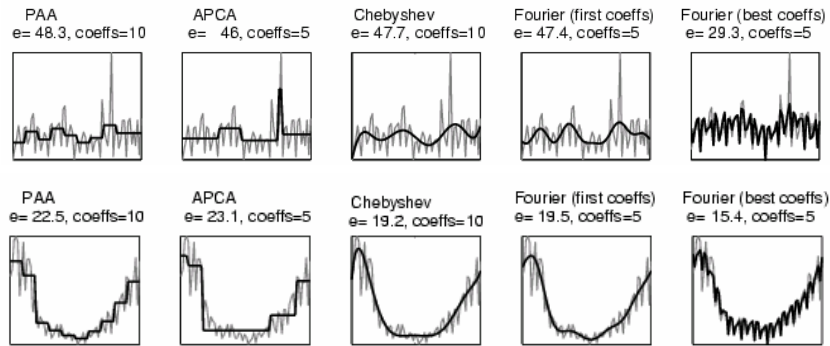


On my income tax 1040 it says "Check this **box** if you are blind." I wanted to put a check mark about three inches away.

- Tom Lehrer

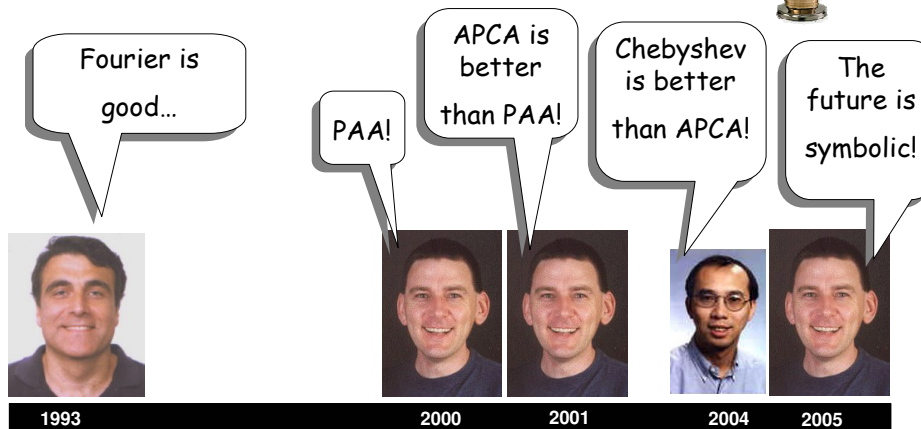
158

Comparison of different Dim. Reduction Techniques



159

So which dimensionality reduction is the best?



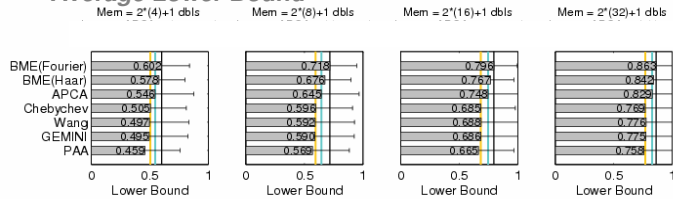
Absence of proof is no proof of absence.
- Michael Crichton

160

Comparisons

Lets see how tight the lower bounds are for a variety on 65 datasets

Average Lower Bound

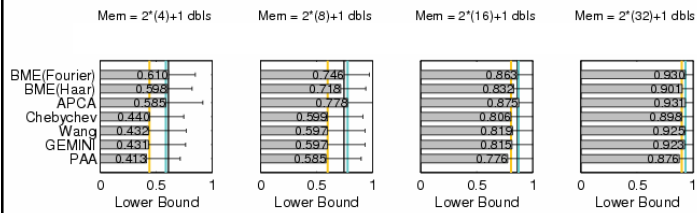


A. No approach is better on all datasets

B. Best coeff. techniques can offer tighter bounds

C. Choice of compression depends on application

Median Lower Bound



Note: similar results also reported by Keogh in SIGKDD02

161

•PART II: Time Series Matching

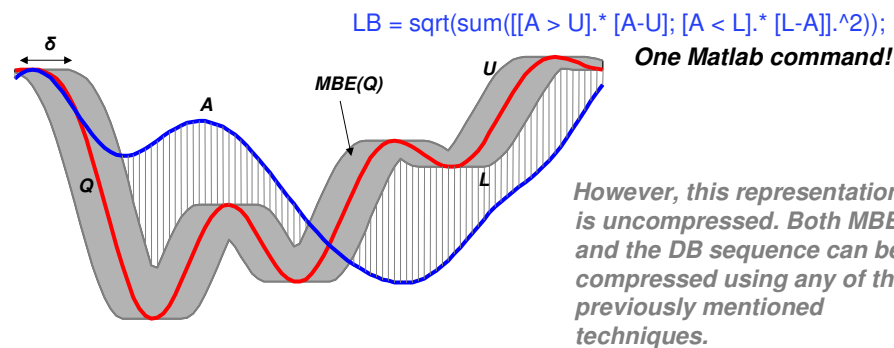
Lower Bounding the DTW and LCSS

162

Lower Bounding the Dynamic Time Warping

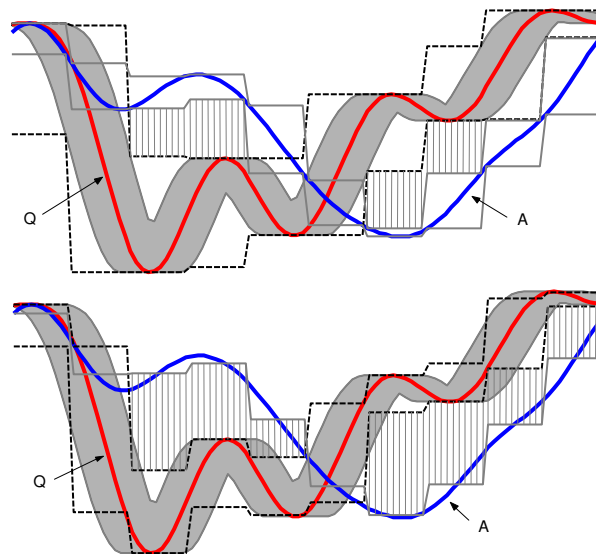
Recent approaches use the Minimum Bounding Envelope for bounding the DTW

- Create Minimum Bounding Envelope (MBE) of query Q
- Calculate distance between MBE of Q and any sequence A
- One can show that: $D(MBE(Q), A) < DTW(Q, A)$



163

Lower Bounding the Dynamic Time Warping



LB by Keogh
 approximate MBE and
 sequence using MBRs

LB = 13.84

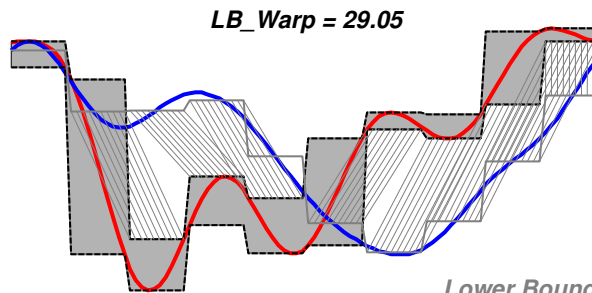
LB by Zhu and Shasha
 approximate MBE and
 sequence using PAA

LB = 25.41

164

Lower Bounding the Dynamic Time Warping

An even tighter lower bound can be achieved by 'warping' the MBE approximation against any other compressed signal.



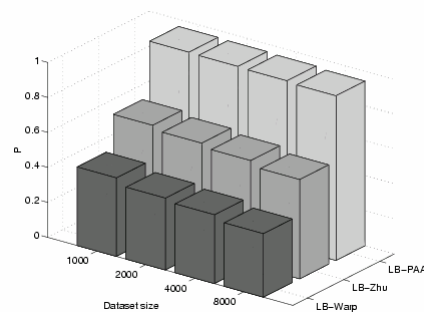
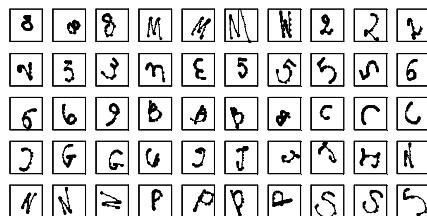
Lower Bounding approaches for DTW, will typically yield at least an order of magnitude speed improvement compared to the naïve approach.

Let's compare the 3 LB approaches: ➡

165

Time Comparisons

We will use DTW (and the corresponding LBs) for recognition of hand-written digits/shapes.



Accuracy: Using DTW we can achieve recognition above 90%.

Running Time: $runTime\ LB_Warp < runTime\ LB_Zhu < runTime\ LB\text{-}Keogh$

Pruning Power: For some queries LB_Warp can examine up to 65 time fewer sequences

166

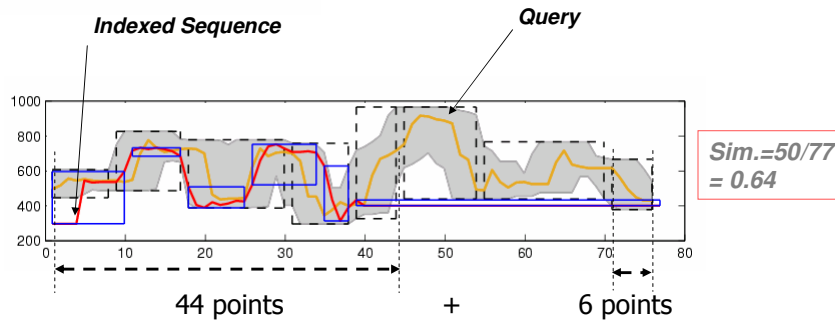
Upper Bounding the LCSS

Since LCSS measures similarity and similarity is the inverse of distance, to speed up LCSS we need to upper bound it.

$$\begin{cases} EnvHigh[i] = \max(Q[i-\delta:i+\delta]) + \epsilon \\ EnvLow[i] = \min(Q[i-\delta:i+\delta]) + \epsilon \end{cases}$$

$$LCSS(MBE_Q, A) = \sum_{i=1}^n \begin{cases} 1 & \text{if } A[i] \text{ within envelope} \\ 0 & \text{otherwise} \end{cases}$$

$$LCSS(MBE_Q, A) \geq LCSS(Q, A)$$



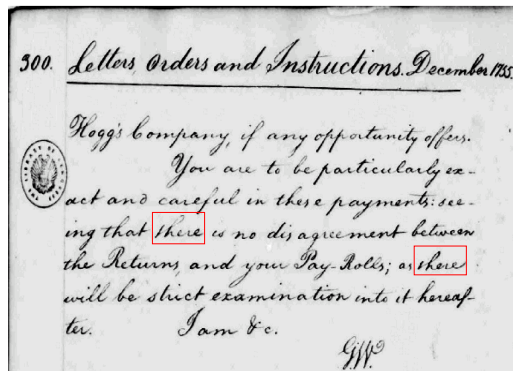
167

LCSS Application – Image Handwriting

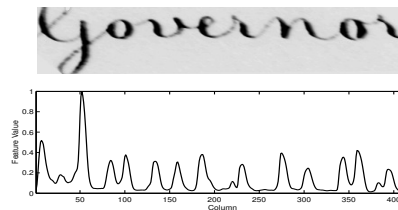
- Library of Congress has 54 million manuscripts (20TB of text)
- Increasing interest for automatic transcribing

Word annotation:

1. Extract words from document
2. Extract image features
3. Annotate a subset of words
4. Classify remaining words



George Washington Manuscript



Features:

- Black pixels / column
- Ink-paper transitions/ col, etc

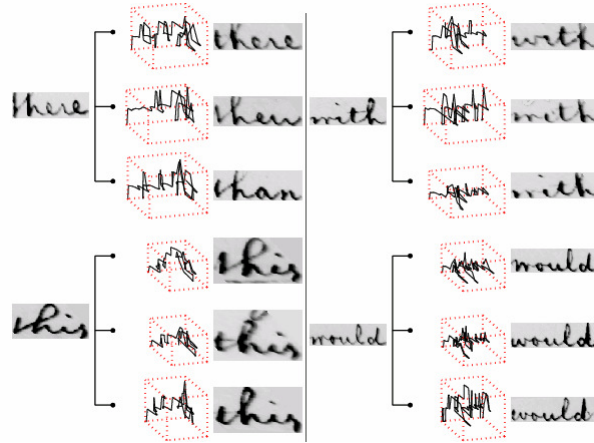
168

LCSS Application – Image Handwriting

Utilized 2D time-series (2 features)

Returned 3-Nearest Neighbors of following words

Classification accuracy > 70%

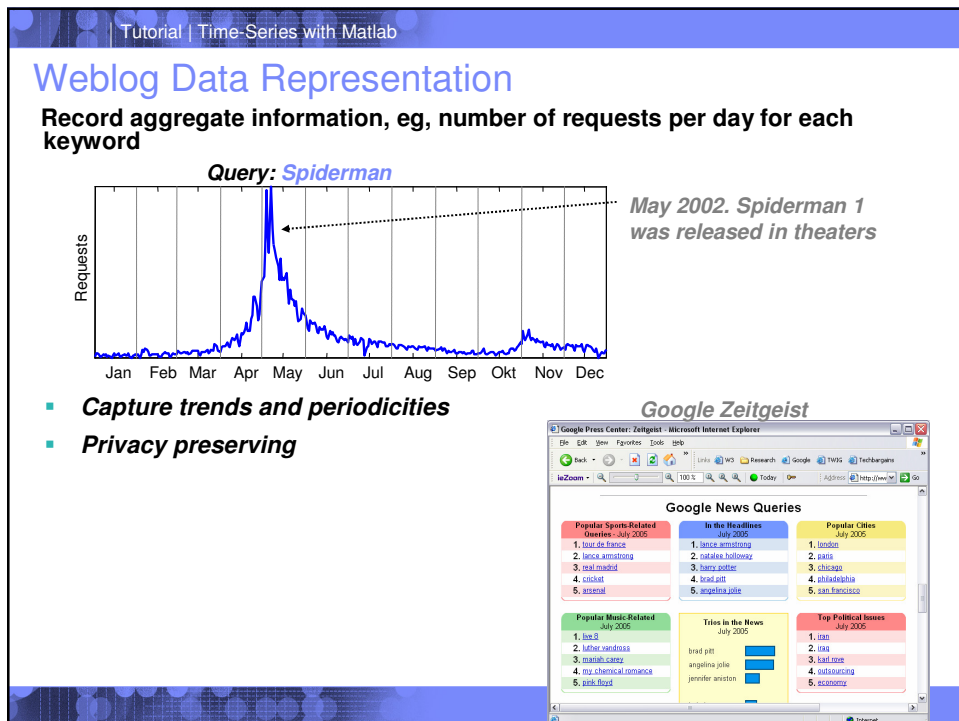
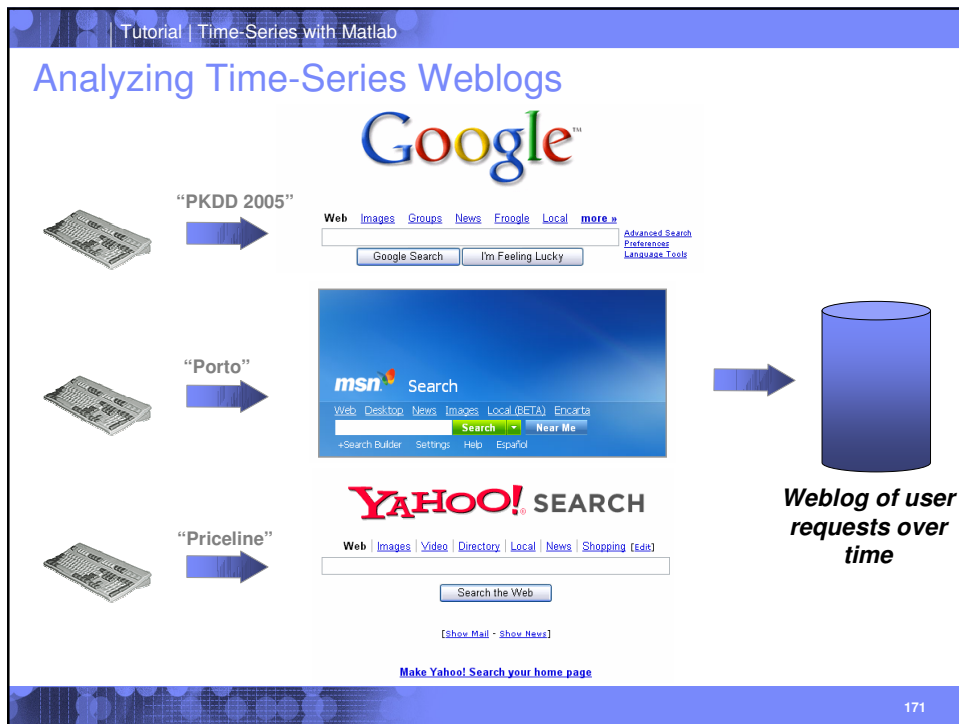


169

•PART II: Time Series Analysis

Test Case and Structural Similarity Measures

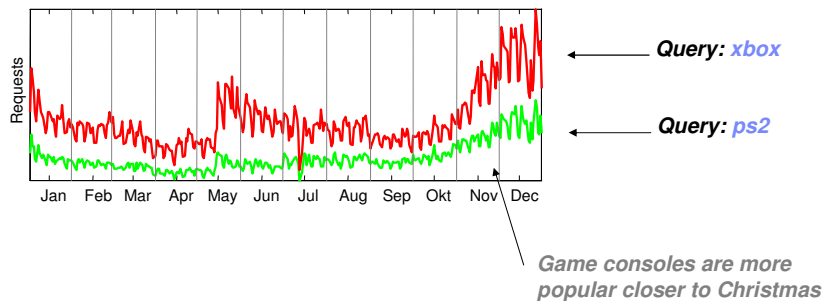
170



Finding similar patterns in query logs

We can find useful patterns and correlation in the user demand patterns which can be useful for:

- **Search engine optimization**
- **Recommendations**
- **Advertisement pricing (e.g. keyword more expensive at the popular months)**

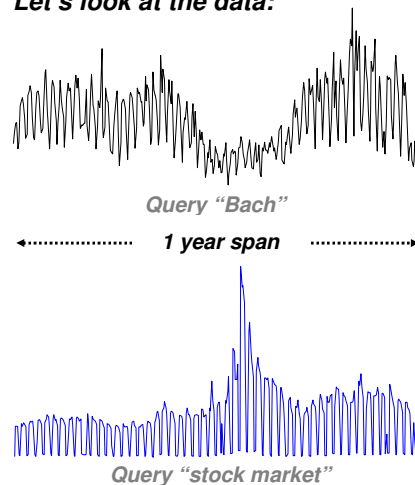


173

Matching of Weblog data

Use Euclidean distance to match time-series. But which dimensionality reduction technique to use?

Let's look at the data:



The data is smooth and highly periodic, so we can use Fourier decomposition.

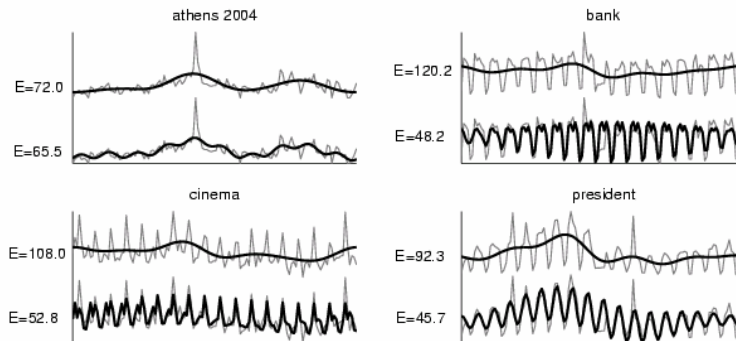
Instead of using the first Fourier coefficients we can use the best ones instead.

Let's see how the approximation will look:



174

First Fourier Coefficients vs Best Fourier Coefficients

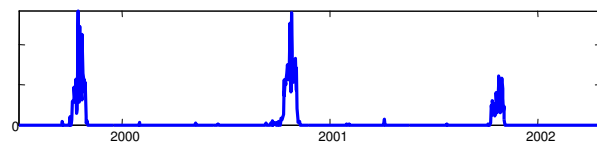
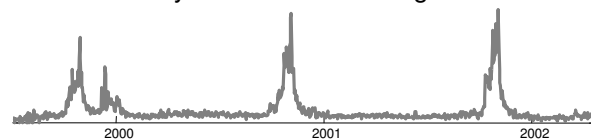


Using the best coefficients, provides a very high quality approximation of the original time-series

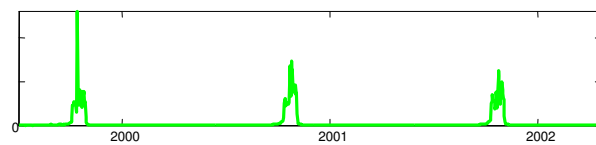
175

Matching results I

Query = "Lance Armstrong"



LeTour

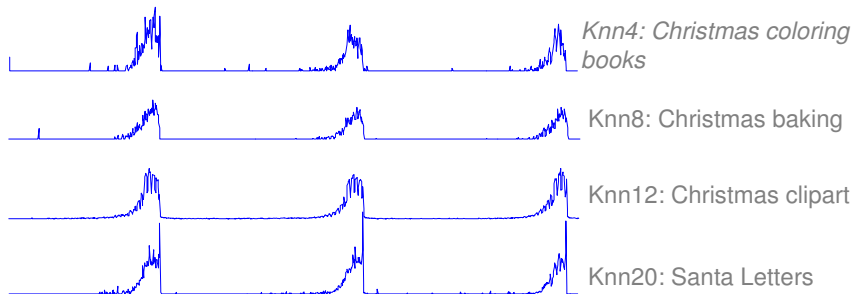
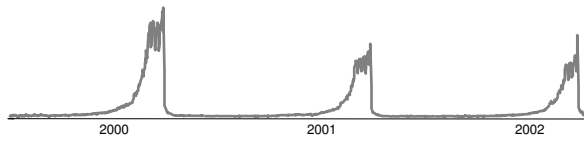


Tour De France

176

Matching results II

Query = "Christmas"

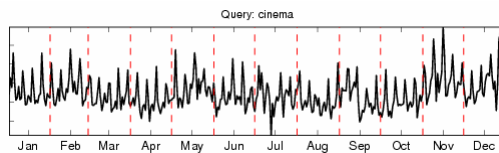


177

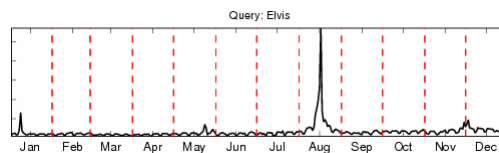
Finding Structural Matches

The Euclidean distance cannot distill all the potentially useful information in the weblog data.

- Some data are **periodic**, while other are **bursty**. We will attempt to provide similarity measures that are based on periodicity and burstiness.

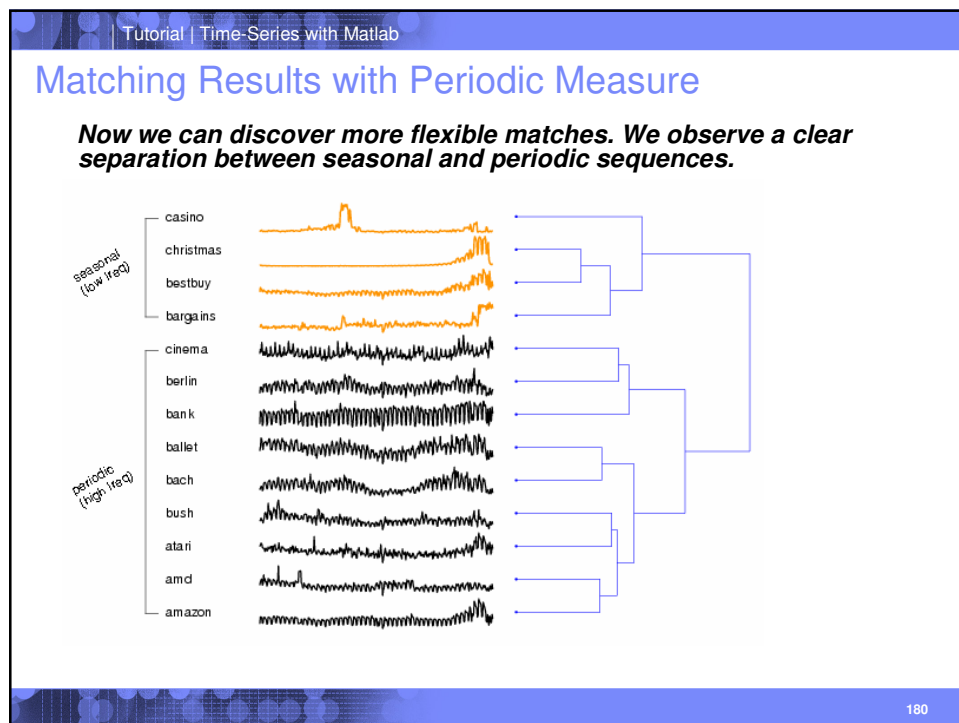
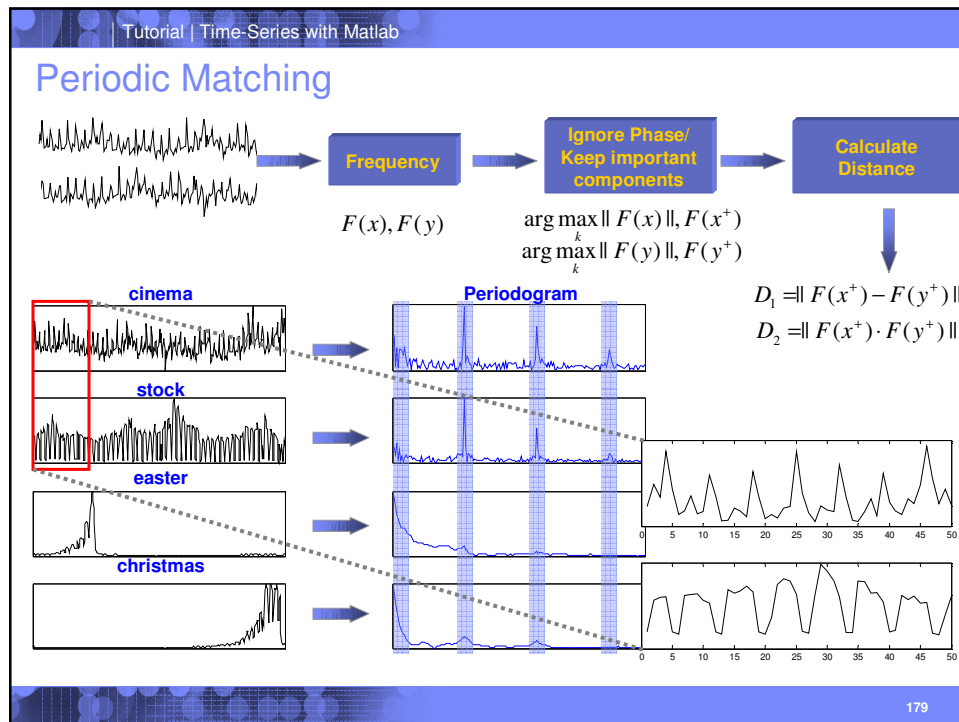


Query "cinema". Weakly periodicity. Peak of period every Friday.



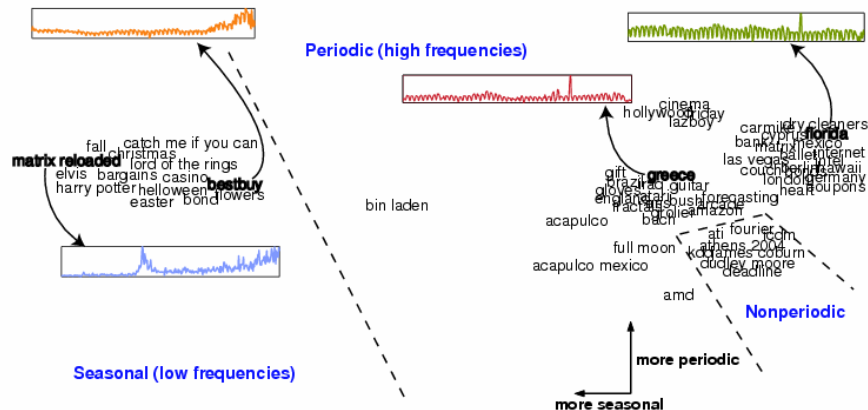
Query "Elvis". Burst in demand on 16th August. Death anniversary of Elvis Presley

178



Matching Results with Periodic Measure

Compute *pairwise periodic distances* and do a mapping of the sequences on 2D using *Multi-dimensional scaling (MDS)*.



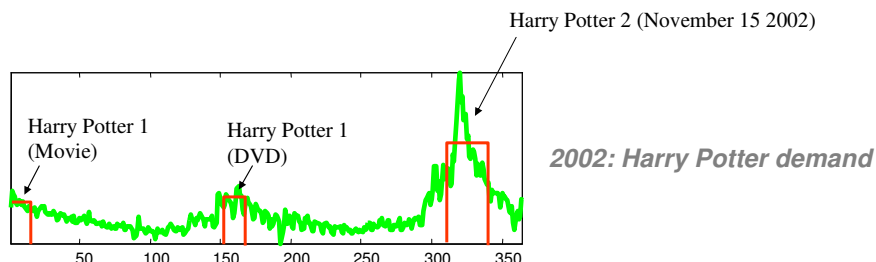
181

Matching Based on Bursts

Another method of performing structural matching can be achieved using *burst features* of sequences.

Burst feature detection can be useful for:

- Identification of important events
- 'Query-by-burst'

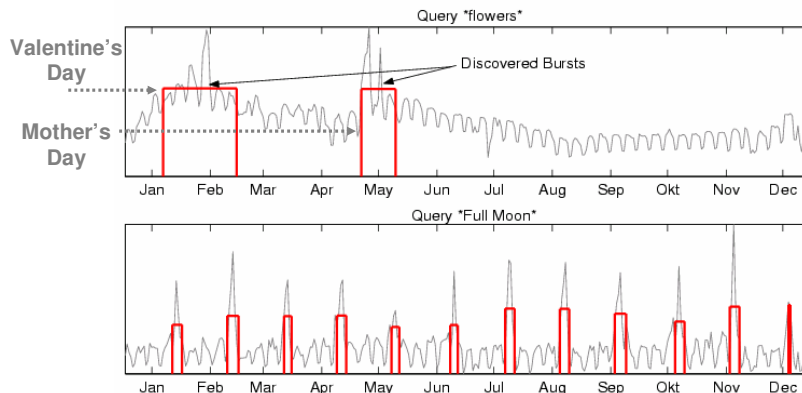


182

Burst Detection

Burst detection is similar to anomaly detection.

- Create distribution of values (eg gaussian model)
- Any value that deviates from the observed distribution (eg more than 3 std) can be considered as burst.

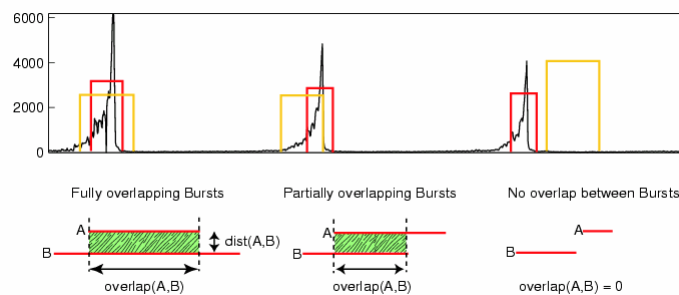


183

Query-by-burst

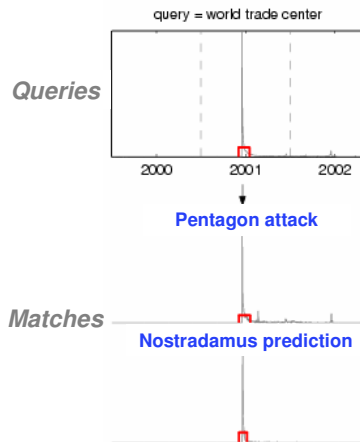
To perform 'query-by-burst' we can perform the following steps:

1. Find burst regions in given query
2. Represent query bursts as time segments
3. Find which sequences in DB have **overlapping burst regions**.



184

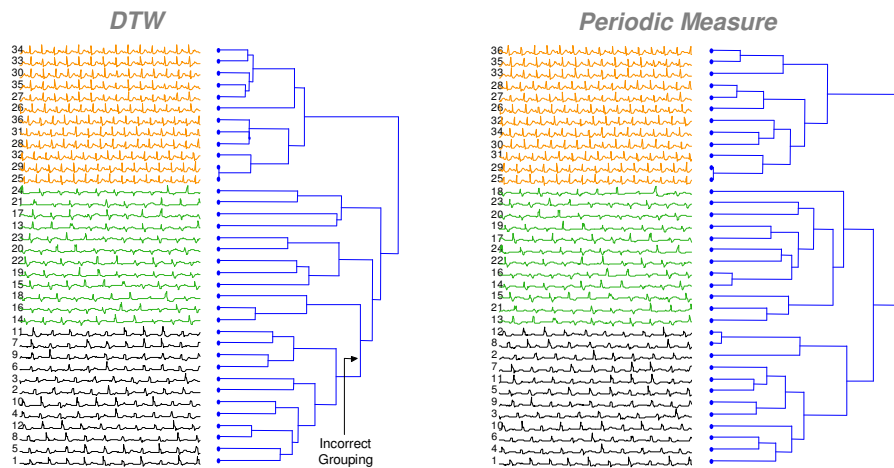
Query-by-burst Results



185

Structural Similarity Measures

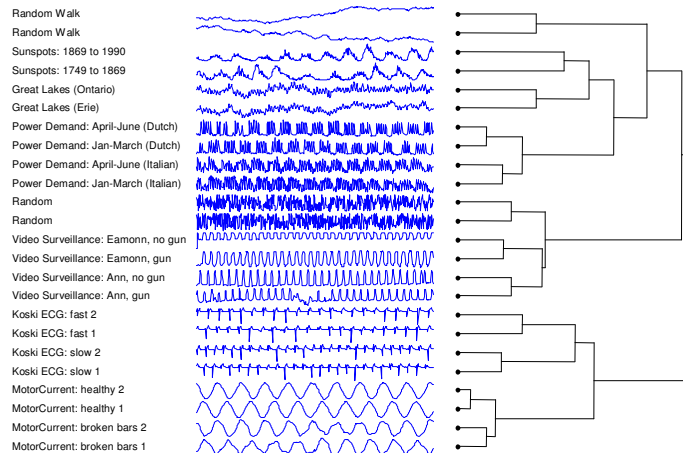
Periodic similarity achieves high clustering/classification accuracy in ECG data



186

Structural Similarity Measures

Periodic similarity is a very powerful visualization tool.



187

Structural Similarity Measures

Burst correlation can provide useful insights for understanding which sequences are related/connected. Applications for:

- ***Gene Expression Data***
- ***Stock market data (identification of causal chains of events)***

Query: Which stocks exhibited trading bursts during 9/11 attacks?

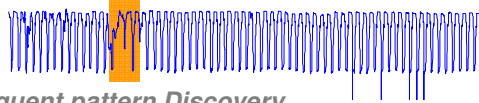
188

Conclusion

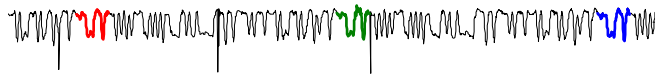
The traditional shape matching measures cannot address all time-series matching problems and applications.
Structural distance measures can provide more flexibility.

There are many other exciting time-series problems that haven't been covered in this tutorial:

- ***Anomaly Detection***



- ***Frequent pattern Discovery***



- ***Rule Discovery***

- ***etc***

I don't want to achieve immortality through my work...I want to achieve it through not dying.

