ECML PKDD >> 2005

16th european conference on machine learning

9th european conference on principles and practice of knowledge discovery in databases

A practical Time-Series Tutorial with MATLAB

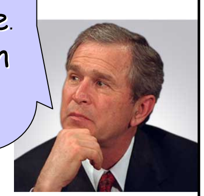
Michalis Vlachos
IBM T.J. Watson Research Center
Hawthorne, NY, 10532

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...



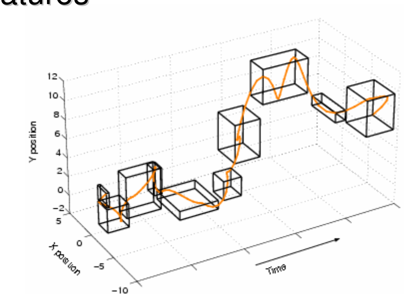
Disclaimer

- 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.**

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

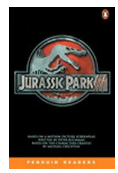


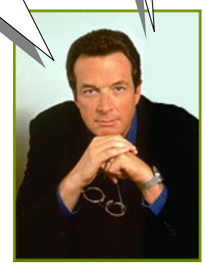
What this tutorial is NOT about

- 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!





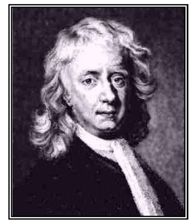
Michael Crichton

PART I: Matlab Introduction

Why does anyone need Matlab?

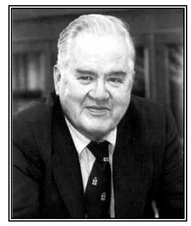
- **Matlab enables the efficient *Exploratory Data Analysis (EDA)***

"Science progresses through observation"
-- Isaac Newton



Isaac Newton

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

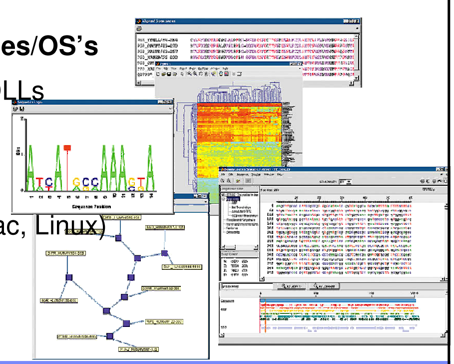


John Tukey

Matlab

The MathWorks
Accelerating the pace of engineering and science

- **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



History of Matlab (MATrix LABoratory)

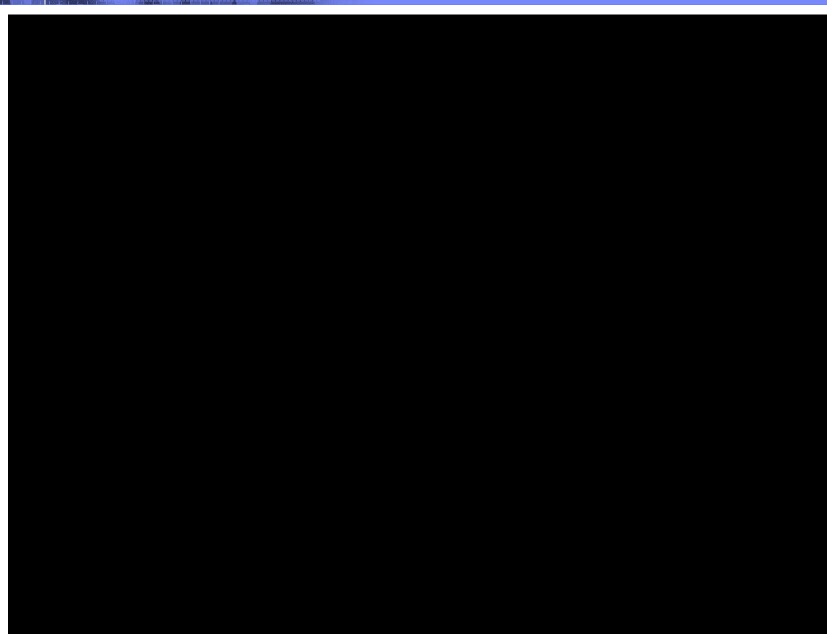
"The most important thing in the programming language is the name. I have recently invented a very good name and now I am looking for a suitable language". -- R. Knuth

Cleve Moler

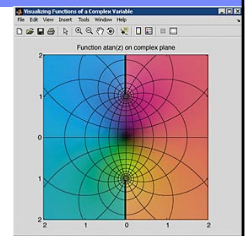
Programmed by Cleve Moler as an interface for EISPACK & LINPACK

- 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**

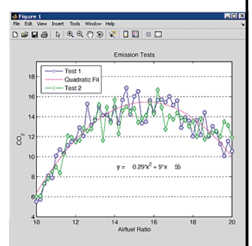
Video:http://www.mathworks.com/company/aboutus/founders/origins_of_matlab_wm.html



Current State of Matlab/Mathworks

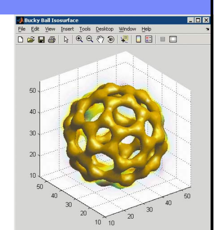


- **Matlab, Simulink, Stateflow**
- **Matlab version 7, service pack 2**
- **Used in variety of industries**
  – Aerospace, defense, computers, communication, biotech
- **Mathworks still is privately owned**
- **Used in >3,500 Universities, with >500,000 users worldwide**
- **2004 Revenue: 300 M.**
-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

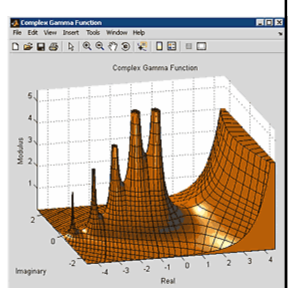


Money is better than poverty, if only for financial reasons. –Woody Allen

Who needs Matlab?



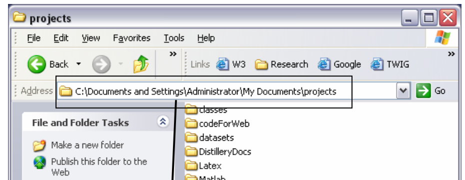
- **R&D companies for easy application deployment**
- **Professors**
  – Lab assignments
  – Matlab allows focus on *algorithms* not on language features
- **Students**
  – Batch processing of files
    - No more incomprehensible perl code!
  – Great environment for testing ideas
    - Quick coding of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

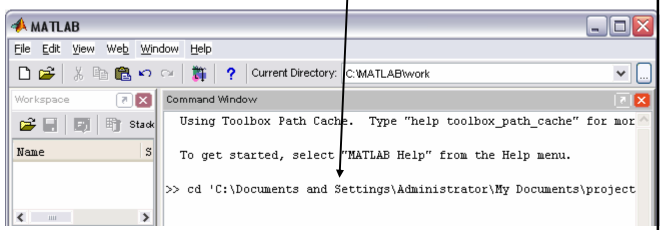


Starting up Matlab

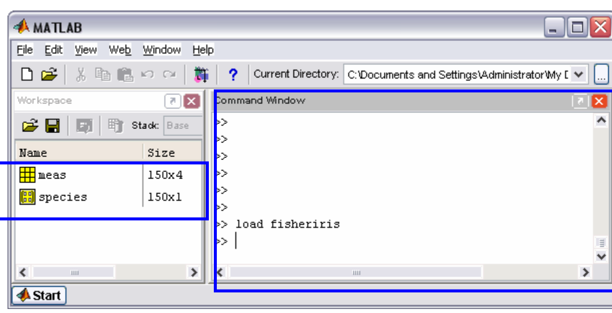
Personally I'm always ready to learn, although I do not always like being taught.
Sir Winston Churchill

- **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\'





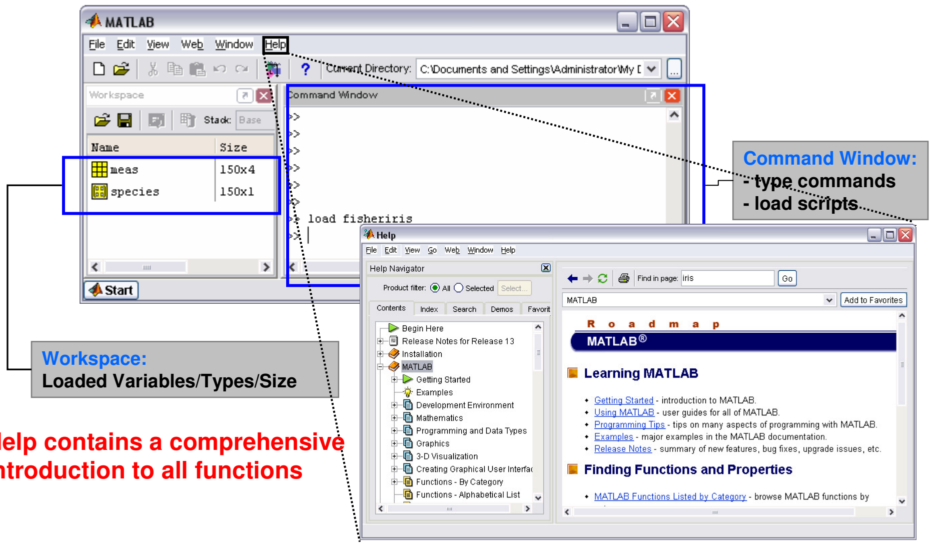
Matlab Environment



Command Window:
- type commands
- load scripts

Workspace:
Loaded Variables/Types/Size

Matlab Environment

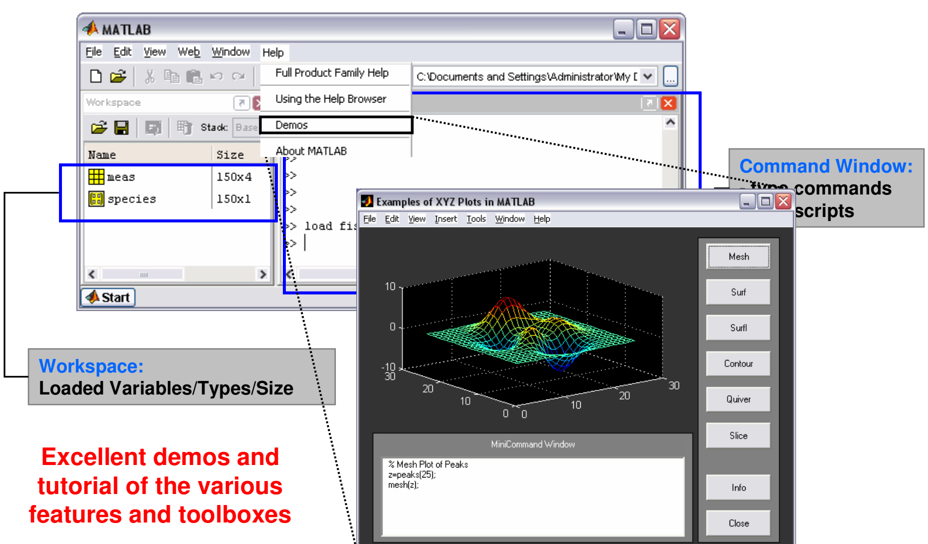


Command Window:
- type commands
- load scripts

Workspace:
Loaded Variables/Types/Size

Help contains a comprehensive introduction to all functions

Matlab Environment



Command Window:
- type commands
- load scripts

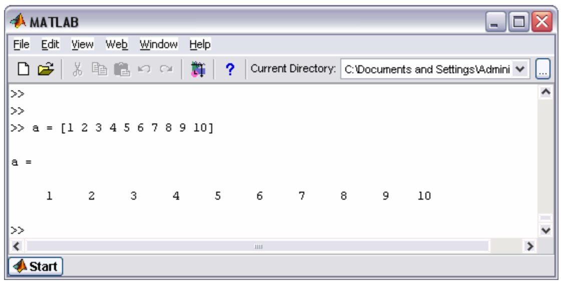
Workspace:
Loaded Variables/Types/Size

Excellent demos and tutorial of the various features and toolboxes

Starting with Matlab

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

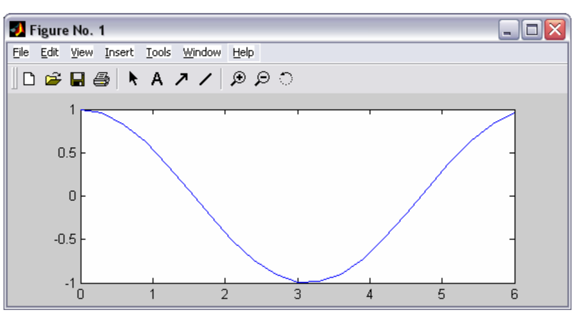
```
a = [1 2 3 4 5 6 7 9 10] % define an array
```



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
```

Array Access

- **Access array elements**

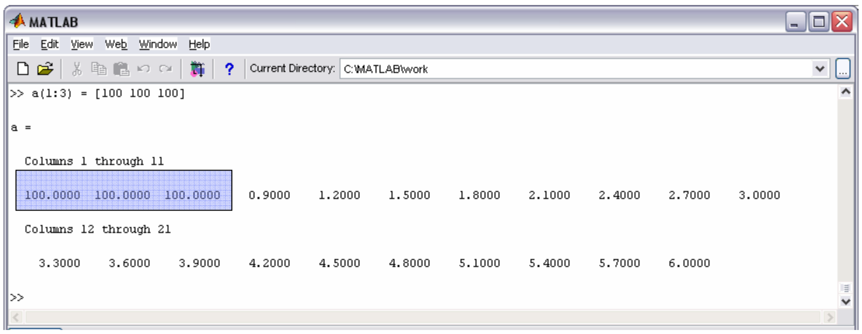
```
>> a(1)

ans =

     0
```

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

- **Set array elements**

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

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



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

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
```


Concatenating arrays

- **Column-wise or row-wise**

Row next to row

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

c =

     1     2     3     4     5     6
```

Column next to column

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

     1     3
     2     4
```

Row below to row

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

c =

     1     2     3
     4     5     6
```

Column below column

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

c =

     1
     2
     3
     4
```

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
```


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

Useful Array functions

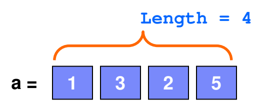
- **Last element of array [`end`]**

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

ans =

     5
```

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

ans =

     2
```

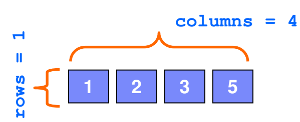
- **Length of array [`length`]**

```
>> length(a)

ans =

     4
```



- **Dimensions of array [`size`]**

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

columns = 4
```



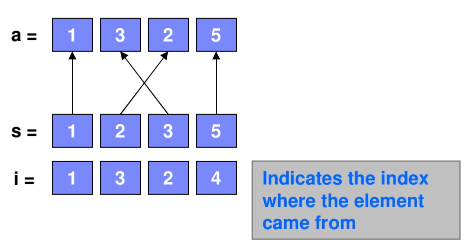
Useful Array functions

- **Find a specific element [`find`] ****

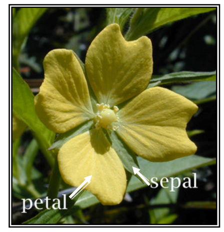
```
>> a = [1 3 2 5 10 5 2 3];
>> b = find(a==2)

b =

     3     7
```

- **Sorting [`sort`] *****

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

s =
     1     2     3     5

i =
     1     3     2     4
```



Visualizing Data and Exporting Figures



- **Use Fisher's Iris dataset**

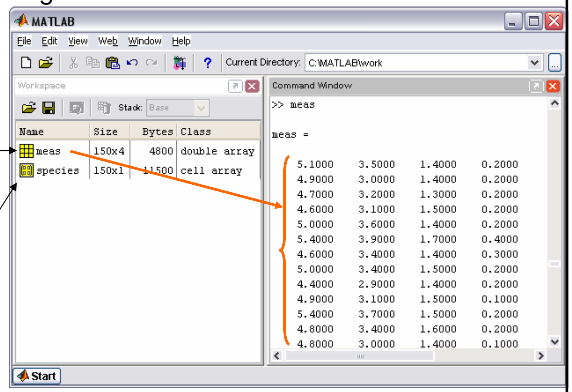
```
>> load fisheriris
```

  - 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'
'virginica'
'virginica'
'virginica'
'virginica'
. . .

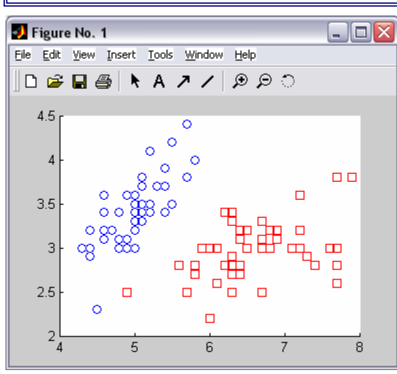
species (150x1 cell array):
Holds name of species for the specific measurement



strcmp, scatter, hold on

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

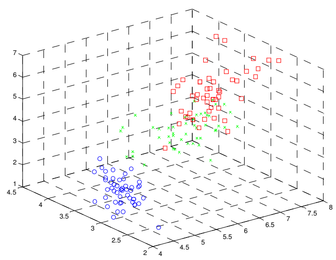
. . .
1
1
1
0
0
0
. . .

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

scatter3

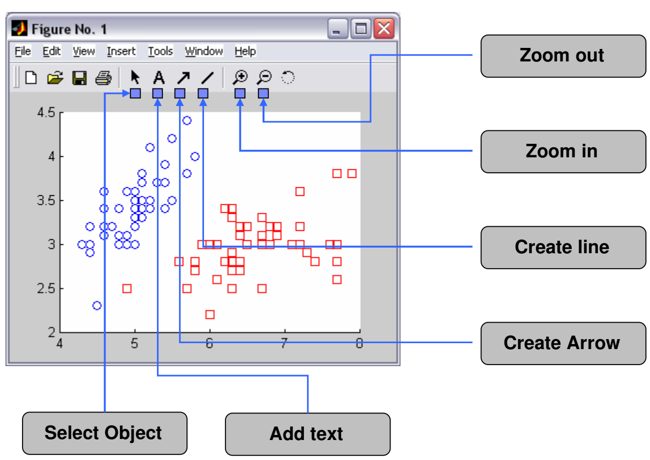
Visualizing Data (3D)

```
>> 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), virgin(:,2),versi(:,3), 'gx'); % green x's
```

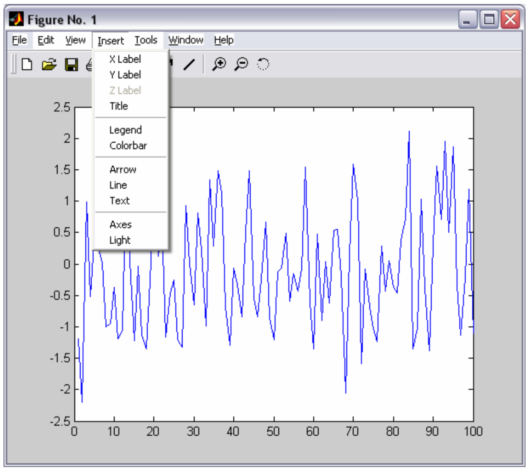


```
>> grid on; % show grid on axis
>> rotate3D on; % rotate with mouse
```


Changing Plots Visually

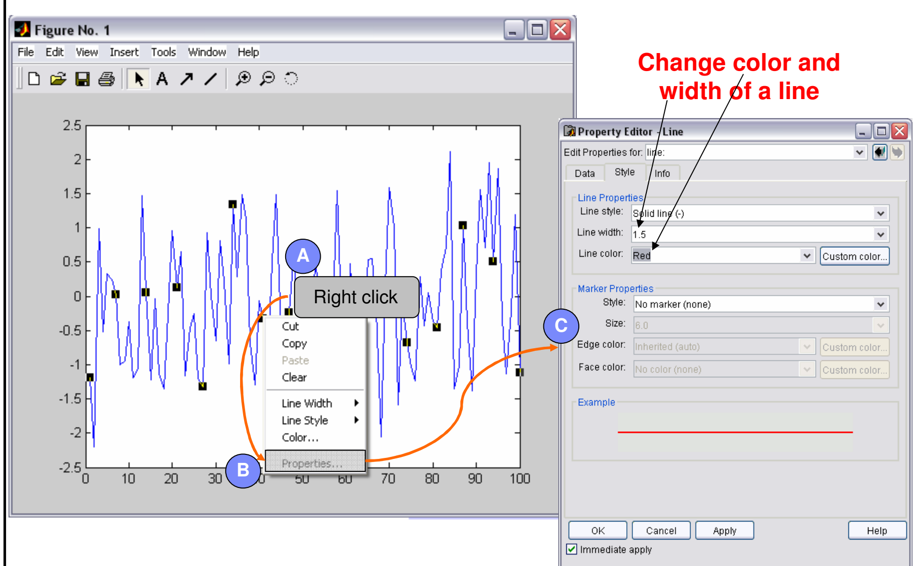


Changing Plots Visually

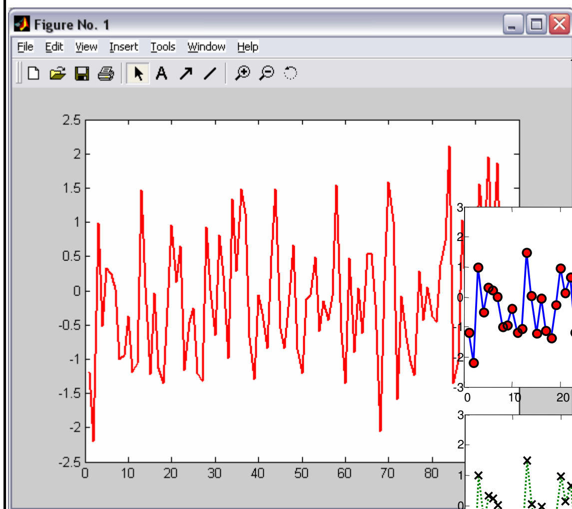


- *Add titles*
- *Add labels on axis*
- *Change tick labels*
- *Add grids to axis*
- *Change color of line*
- *Change thickness/ Linestyle*
- *etc*

Changing Plots Visually (Example)

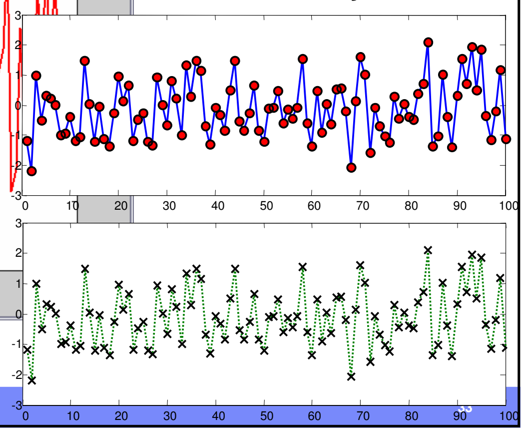


Changing Plots Visually (Example)



The **result** …

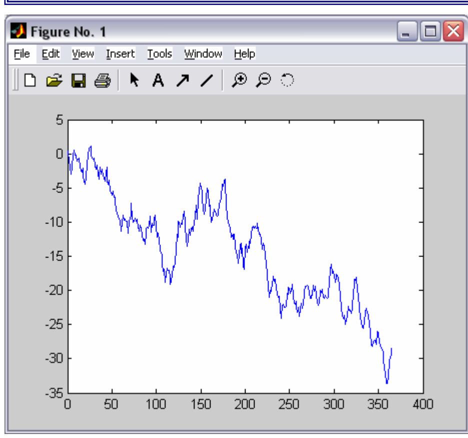
Other Styles:



Changing Figure Properties with Code

- **GUI's are easy, but sooner or later we realize that coding is faster**

```
>> a = cumsum(randn(365,1));  % random walk of 365 values
```



If this represents a year's worth of measurements of an imaginary quantity, we will change:

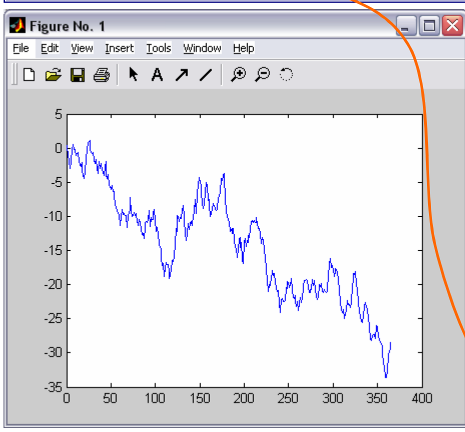
- **x-axis annotation to months**
- **Axis labels**
- **Put title in the figure**
- **Include some greek letters in the title *just for fun***

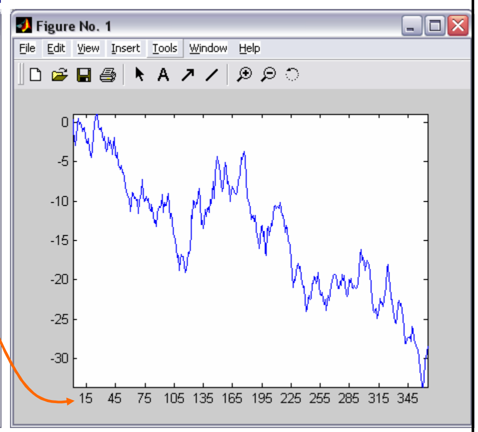
Changing Figure Properties with Code

- **Axis annotation to months**

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

The result …





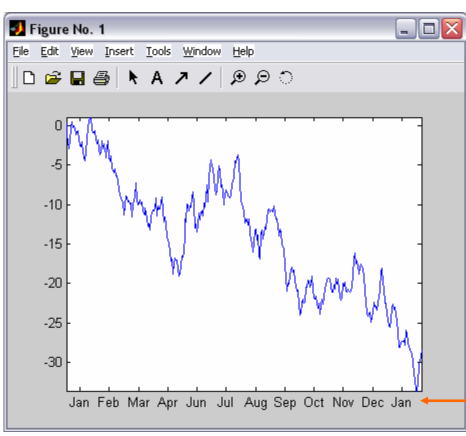
Tutorial | Time-Series with Matlab

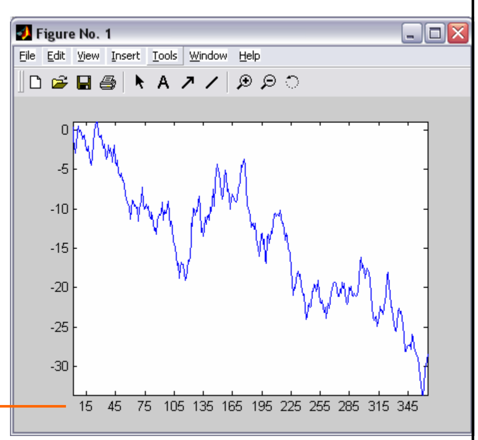
Changing Figure Properties with Code

- **Axis annotation to months**

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

The result …





Changing Figure Properties with Code

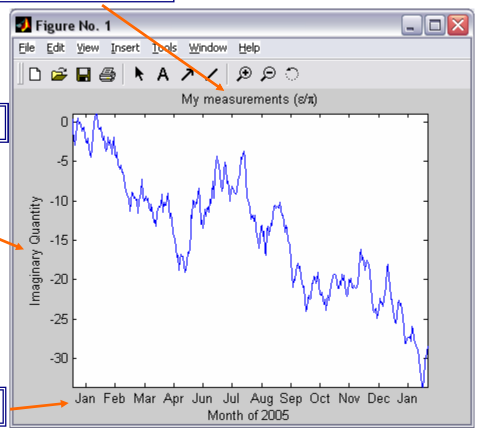
- **Axis labels and title**

Other latex examples:
\alpha, \beta, e^{-\alpha} etc

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

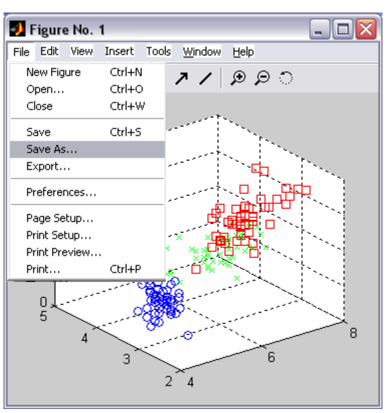
```
>> ylabel('Imaginary Quantity')
```

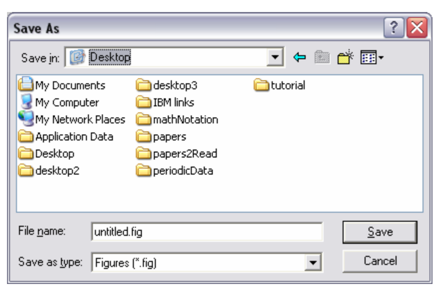
```
>> xlabel('Month of 2005')
```



Saving Figures

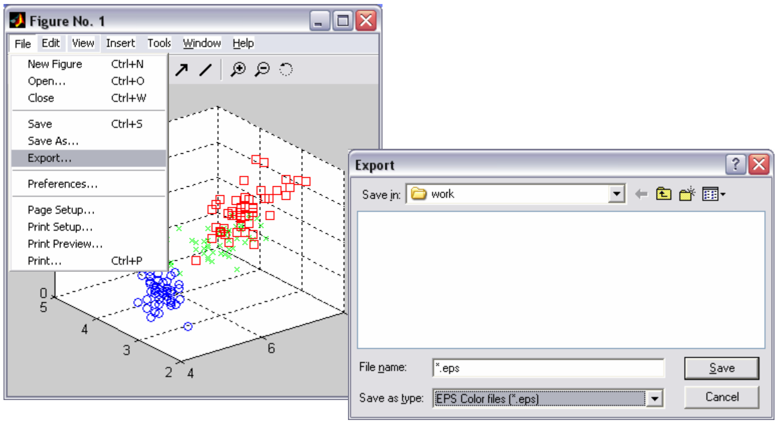
- **Matlab allows to save the figures (.fig) for later processing**





.fig can be later opened through Matlab

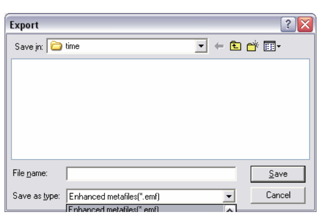
Exporting Figures

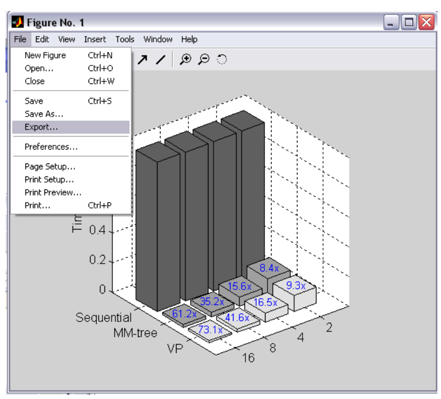


Export to:
emf, eps, jpg, etc

Exporting figures (code)

- **You can also achieve the same result with Matlab code**

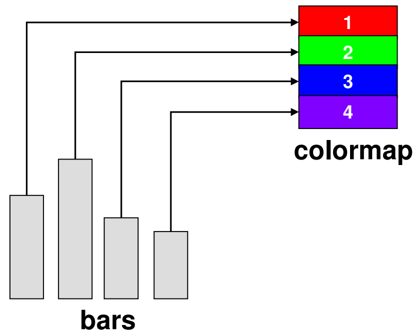




- **Matlab code:**

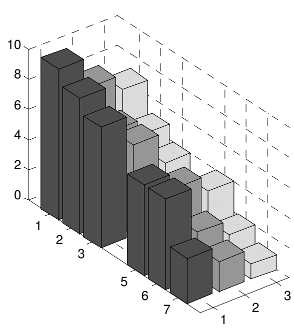
```
% extract to color eps
print -depsc myImage.eps;  % from command-line
print(gcf,'-depsc','myImage') % using variable as name
```

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);
```


Visualizing Data - 3D Bars



data

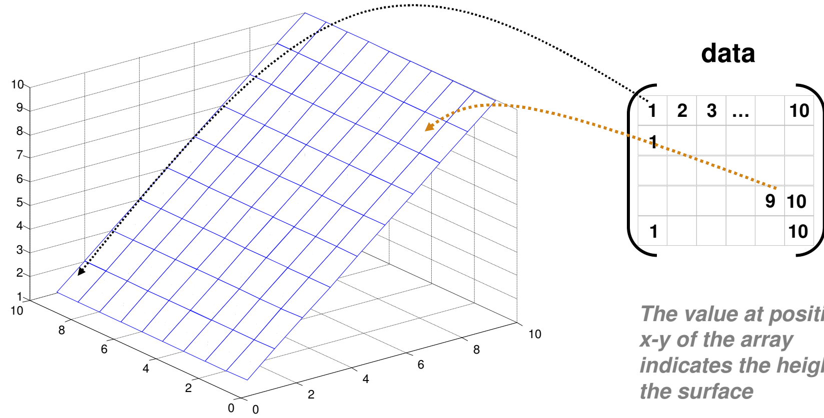
10	8	7
9	6	5
8	6	4
6	5	4
6	3	2
3	2	1

colormap

0	0	0
0.0198	0.0124	0.0079
0.0397	0.0248	0.0158
0.0595	0.0372	0.0237
0.0794	0.0496	0.0316
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
```

Visualizing Data - Surfaces

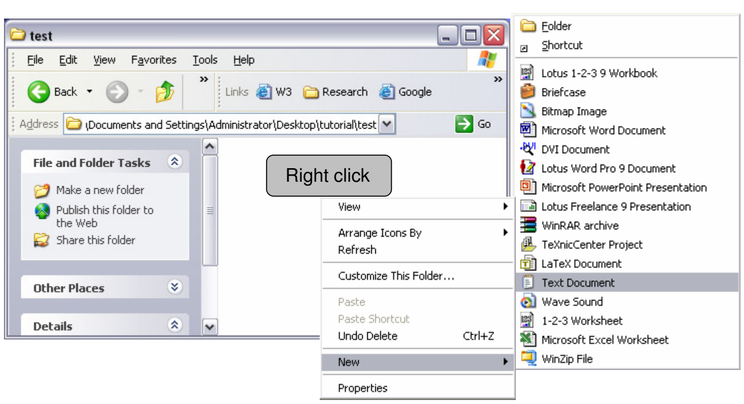


The value at position x-y of the array indicates the height of the surface

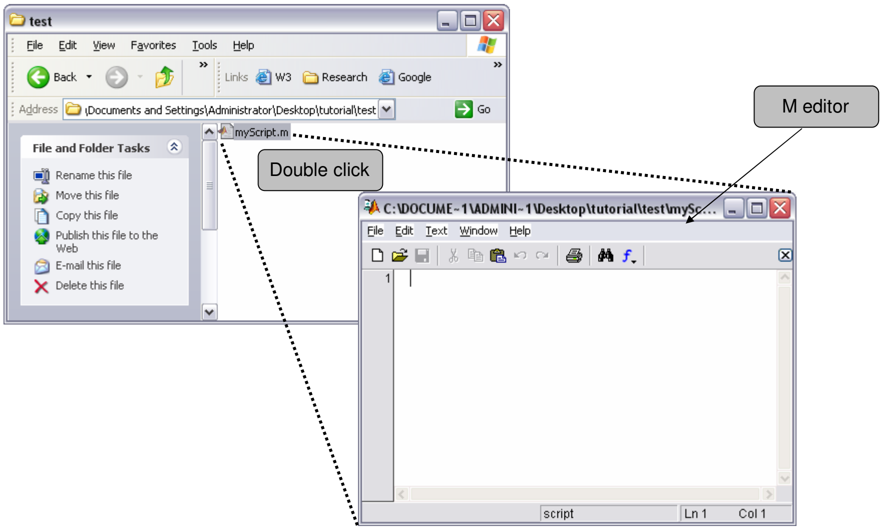
```
data = [1:10];
data = repmat(data,10,1);  % create data
surface(data,'FaceColor',[1 1 1], 'Edgecolor', [0 0 1]); % plot data
view(3); grid on; % change viewpoint and put axis lines
```


Creating .m files

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



Creating .m files



cumsum, num2str, save

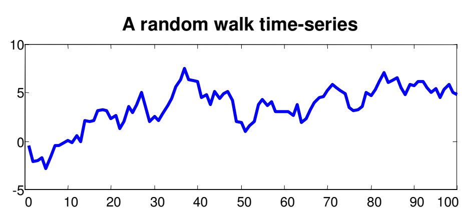
Creating .m files

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

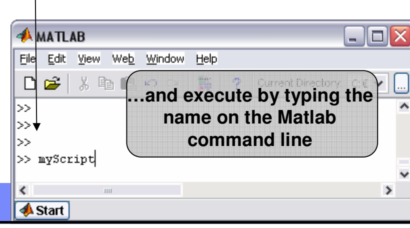
myScript.m **Sample Script**

```
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



Write this in the M editor…



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

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

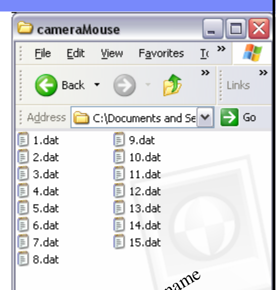
```
function [a,b] = myFunc(data, x, y) % pass & return more arguments
```

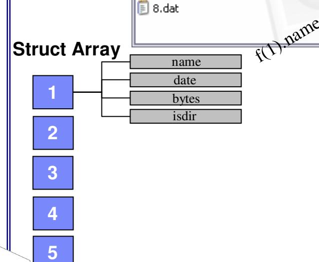
See also:varargin, varargout

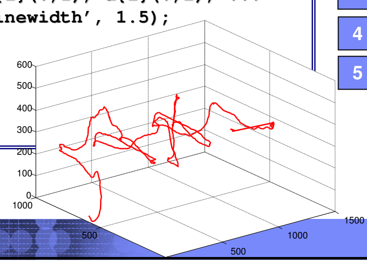
Cell Arrays

- **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
```





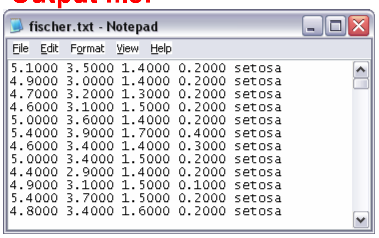


Reading/Writing Files

- **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:


fischer.txt - Notepad

File Edit Format View Help

```
5.1000 3.5000 1.4000 0.2000 setosa
4.9000 3.0000 1.4000 0.2000 setosa
4.7000 3.2000 1.3000 0.2000 setosa
4.6000 3.1000 1.5000 0.2000 setosa
5.0000 3.6000 1.4000 0.2000 setosa
5.4000 3.9000 1.7000 0.4000 setosa
4.6000 3.4000 1.4000 0.3000 setosa
5.0000 3.4000 1.5000 0.2000 setosa
4.4000 2.9000 1.4000 0.2000 setosa
4.9000 3.1000 1.5000 0.1000 setosa
5.4000 3.7000 1.5000 0.2000 setosa
4.8000 3.4000 1.6000 0.2000 setosa
```

- **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);
```

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

Life is pleasant. Death is peaceful. It's the transition that's troublesome. –Isaac Asimov tic, toc, clear all

For-Loop or vectorization?

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

elapsed_time =

5.0070

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

elapsed_time =

0.1400

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

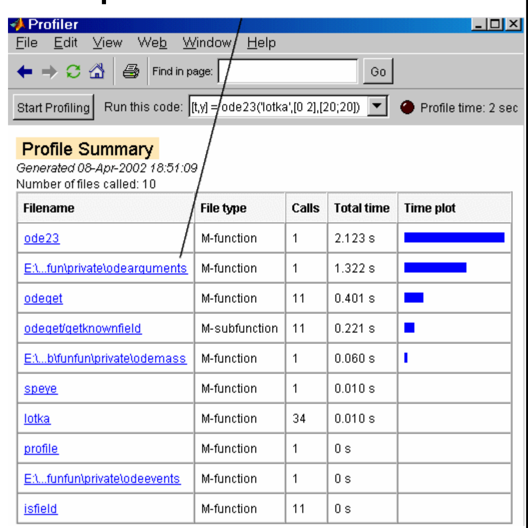
elapsed_time =

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**

Matlab Profiler

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



Profile Summary

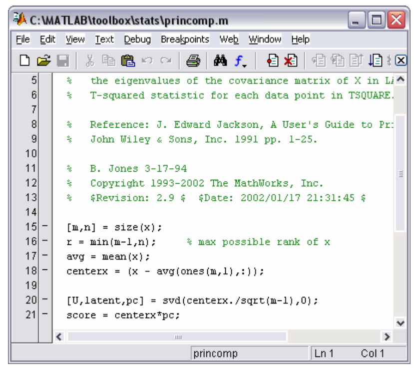
Generated 08-Apr-2002 18:51:09

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:\..b\funfun\private\odemass	M-function	1	0.060 s	
speye	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	

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**

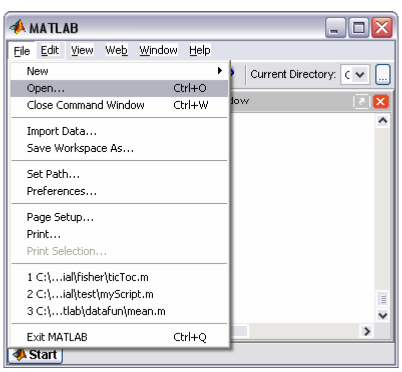


Debugging

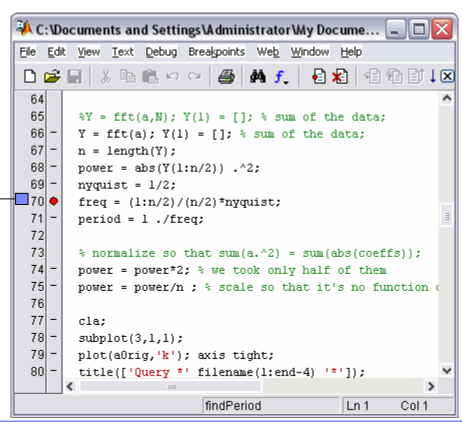
Beware of bugs in the above code; I have only proved it correct, not tried it

-- R. Knuth

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

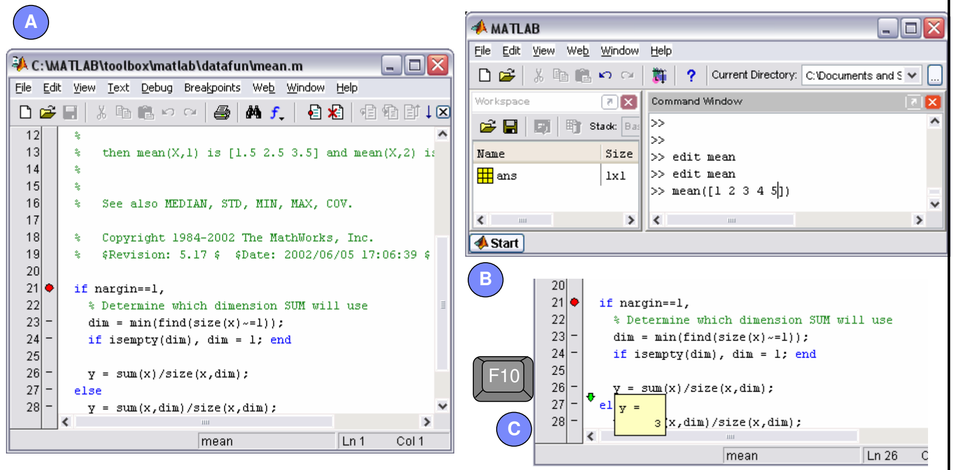


Set breakpoints



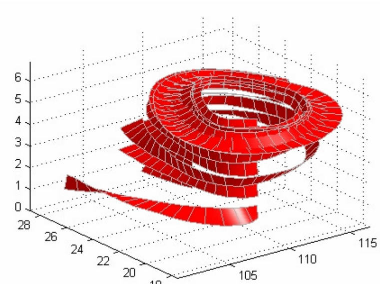
Debugging

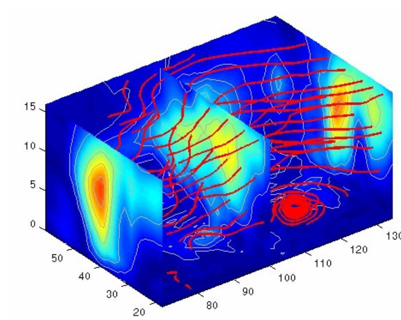
- **Full control over variables and execution path**
  - F10: step, F11: step in (visit functions, as well)



Advanced Features – 3D modeling/Volume Rendering

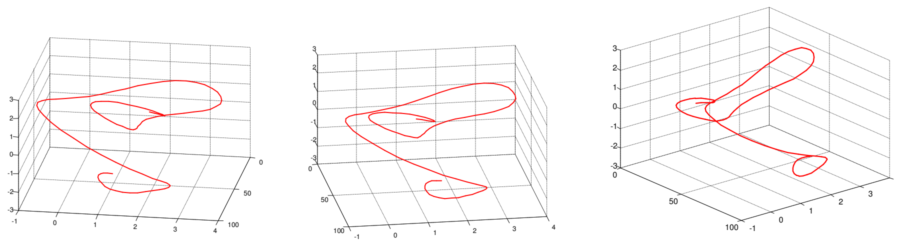
- **Very easy volume manipulation and rendering**





Advanced Features – Making Animations (Example)

- **Create animation by changing the camera viewpoint**

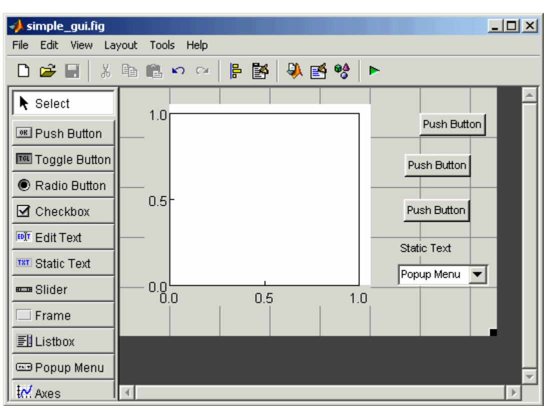


```
azimuth = [50:100 99:-1:50]; % azimuth range of values
for k = 1:length(azimuth),
    plot3(1:length(a), a(:,1), a(:,2), 'r', 'Linewidth',2);
    grid on;
    view(azimuth(k),30); % change new
    M(k) = getframe; % save the frame
end

movie(M,20); % play movie 20 times
```

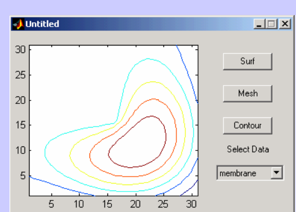
See also:movie2avi

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

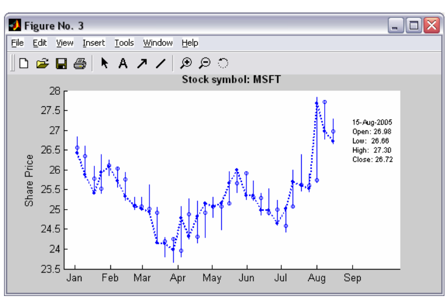


Advanced Features – Using Java

- **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**

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
```

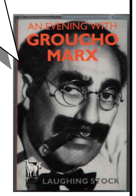
Matlab Toolboxes

- **You can buy many specialized toolboxes from Mathworks**
  - Image Processing, Statistics, Bio-Informatics, etc

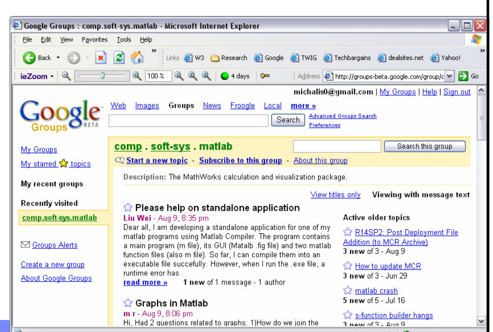
- **There are many equivalent *free* toolboxes too:**
  - SVM toolbox
    - http://theoval.sys.uea.ac.uk/~gcc/svm/toolbox/
  - Wavelets
    - http://www.math.rutgers.edu/~ojanen/wavekit/
  - Speech Processing
    - http://www.ee.ic.ac.uk/hp/staff/dmb/voicebox/voicebox.html
  - Bayesian Networks
    - http://www.cs.ubc.ca/~murphyk/Software/BNT/bnt.html

In case I get stuck…

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

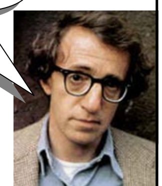


- **help [command] (on the command line)**
  eg. `help fft`
- **Menu: help -> matlab help**
  - Excellent introduction on various topics
- **Matlab webinars**
  - http://www.mathworks.com/company/events/archived_webinars.html?fp
- **Google groups**
  - **comp.soft-sys.matlab**
  - **You can find *anything*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



PART II: Time Series Analysis

Eight percent of success is showing up.

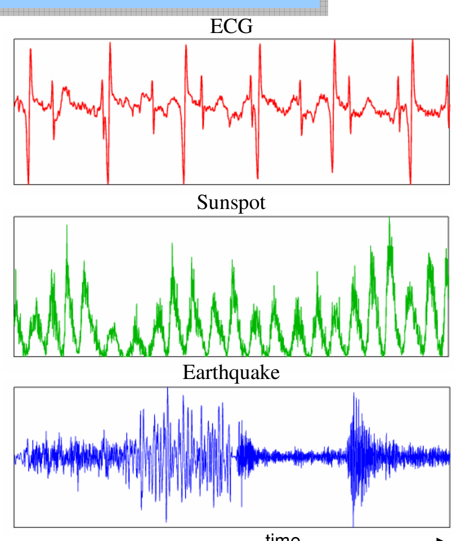


What is a time-series

Definition: *A sequence of measurements over time*

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

64.0
62.8
62.0
66.0
62.0
32.0
86.4
. . .
21.6
45.2
43.2
53.0
43.2
42.8
43.2
36.4
16.9
10.0
…

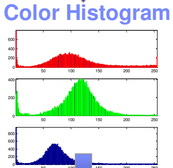


Applications (Image Matching)

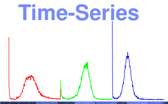
Many types of data can be converted to time-series

Image

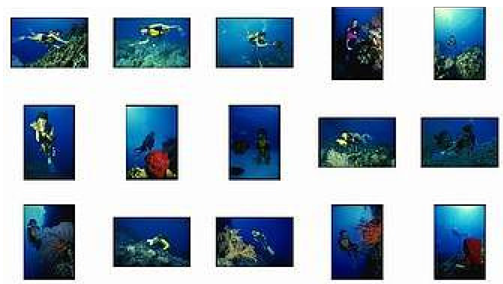
Color Histogram



Time-Series



Cluster 1

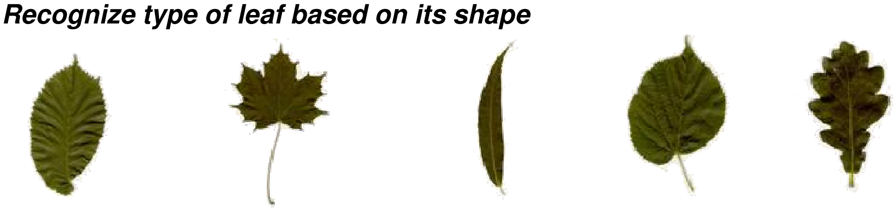


Cluster 2

Tutorial | Time-Series with Matlab

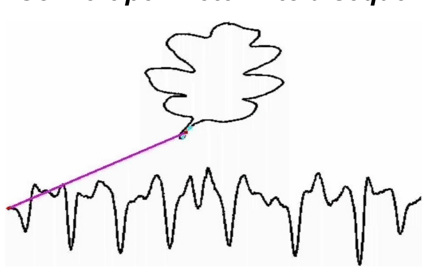
Applications (Shapes)

Recognize type of leaf based on its shape



Ulmus carpinifolia **Acer platanoides** **Salix fragilis** **Tilia** **Quercus robur**

Convert perimeter into a sequence of values

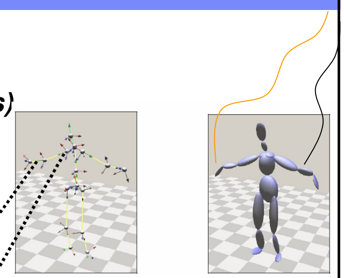


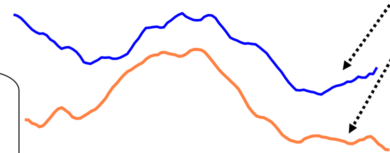
Special thanks to A. Ratanamahatana & E. Keogh for the leaf video.

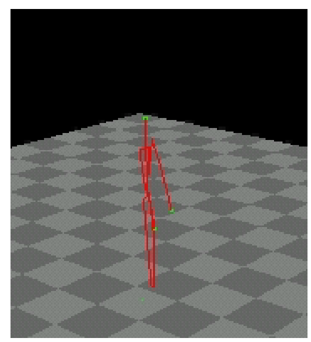
Applications (Motion Capture)

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

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





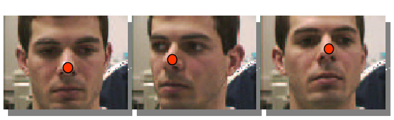


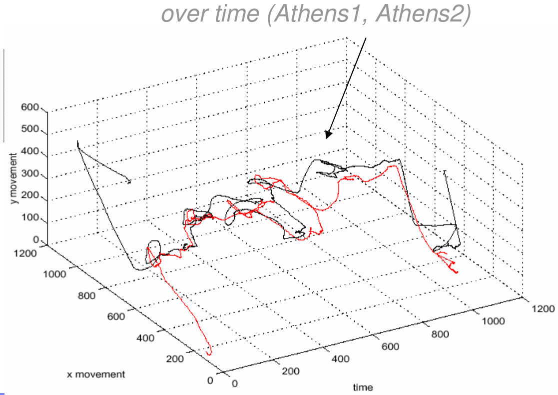
Applications (Video)

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)





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

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

•PART II: Time Series Matching
Introduction

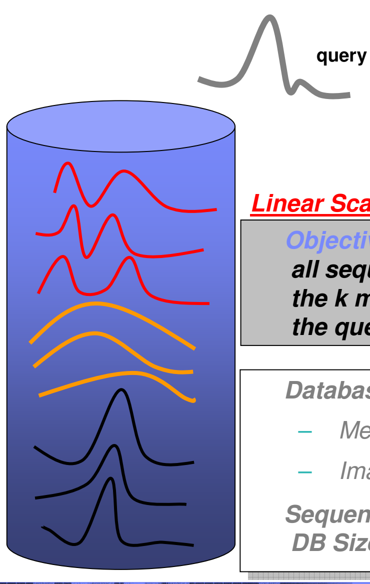
Basic Data-Mining problem

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

Basic Time-Series Matching Problem



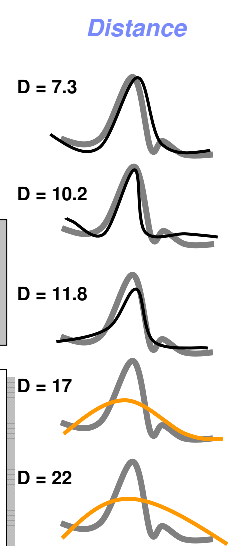
Linear Scan:

***Objective:* Compare the query with all sequences in DB and return the k most similar sequences to the query.**

Database with time-series:

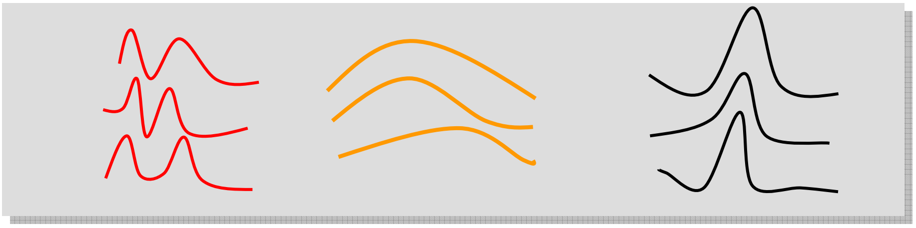
- *Medical sequences*
- *Images, etc*

Sequence Length:100-1000pts
DB Size: 1 TByte

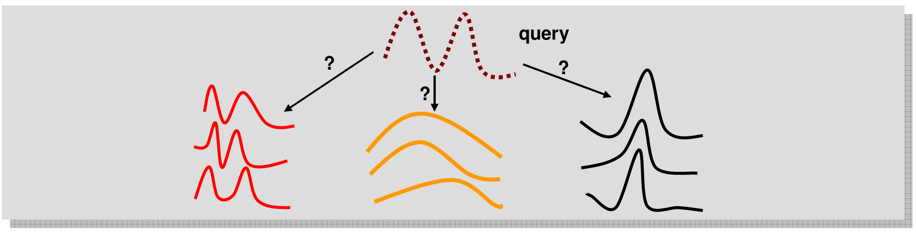


What other problems can we solve?

Clustering: **"Place time-series into '<u>similar</u>' groups"**

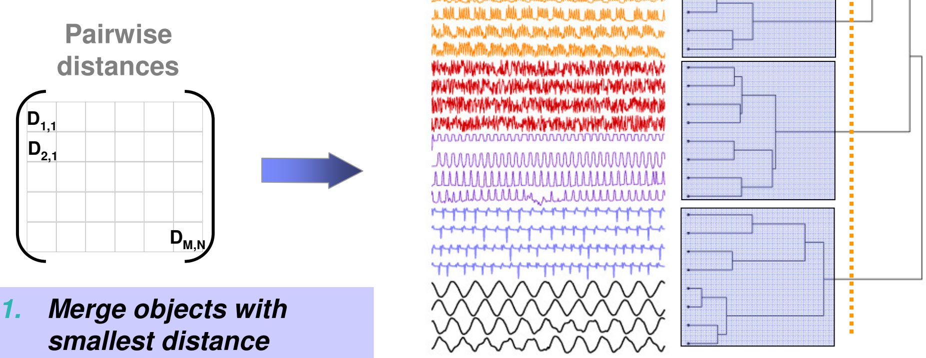


Classification: **"To which group is a time-series most '<u>similar</u>' to?"**



Hierarchical Clustering

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



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

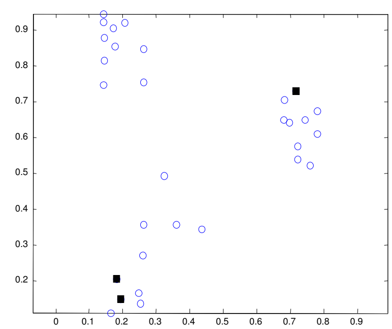
```
Z = linkage(D);
H = dendrogram(Z);
```

Partitional Clustering

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

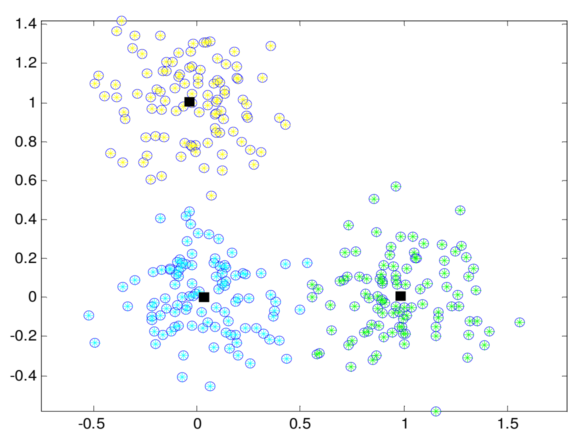
K-Means Algorithm:

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.*



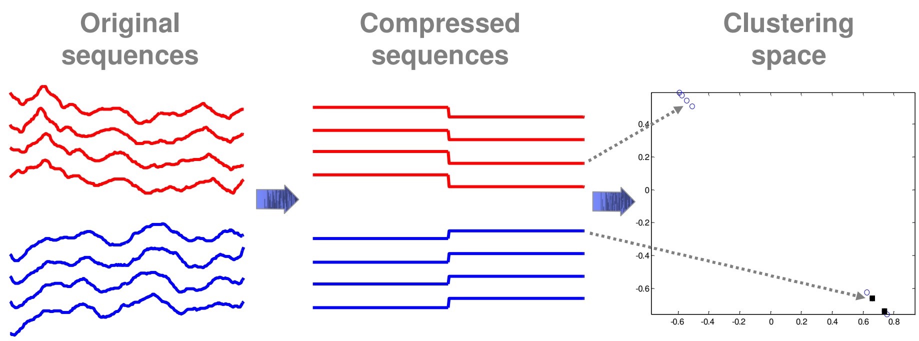
See: kmeans

K-Means Demo



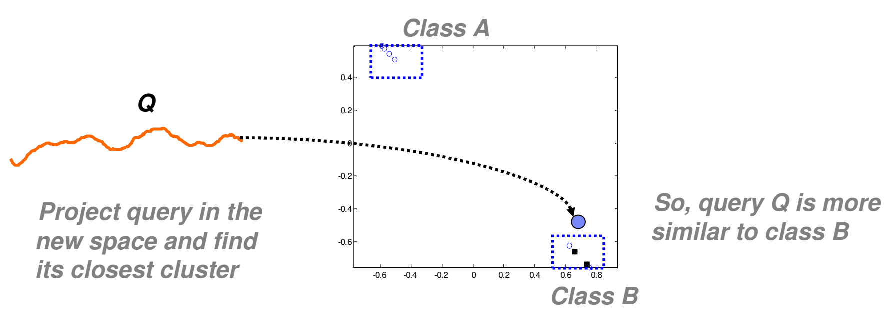
K-Means Clustering for Time-Series

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



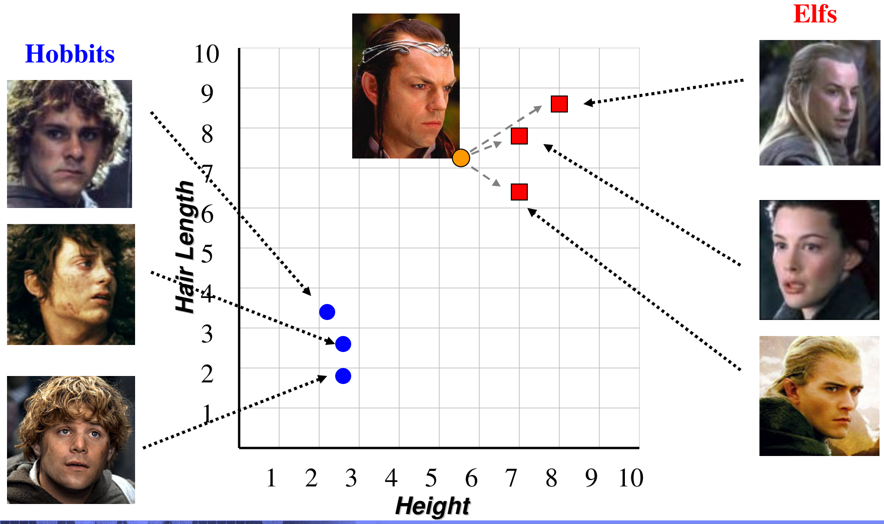
Classification

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

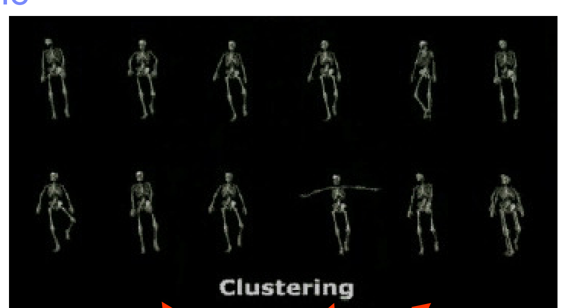


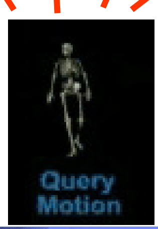
Nearest Neighbor Classification

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



Example





What do we need?

1. Define Similarity

2. Search fast

- Dimensionality Reduction (compress data)

Tutorial | Time-Series with Matlab

All models are wrong, but some are useful…

•PART II: Time Series Matching

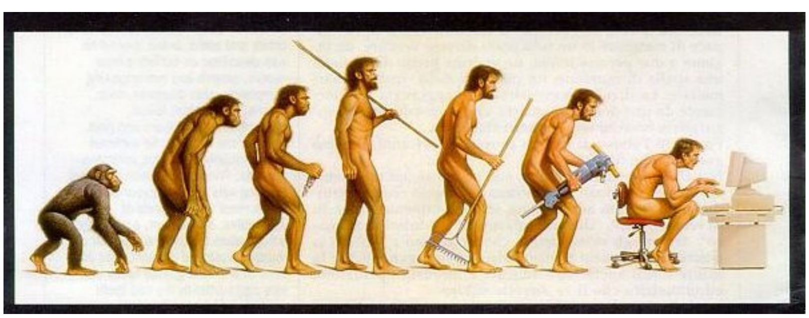
Similarity/Distance functions

81

Tutorial | Time-Series with Matlab

Notion of Similarity I

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



Similarity is always subjective.

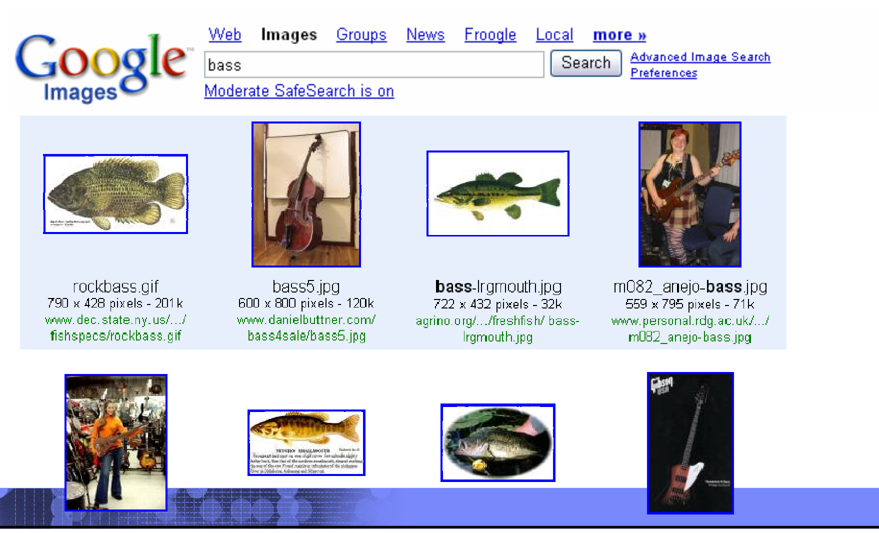
(*i.e. it depends on the application*)

82

Notion of Similarity II

Similarity depends on the features we consider

(i.e. how we will describe or compress the sequences)



Metric and Non-metric Distance Functions

Distance functions

Metric	**Non-Metric**
Euclidean Distance	*Time Warping*
Correlation	*LCSS*

Properties

Positivity: **d(x,y) ≥0 and d(x,y)=0, if x=y**

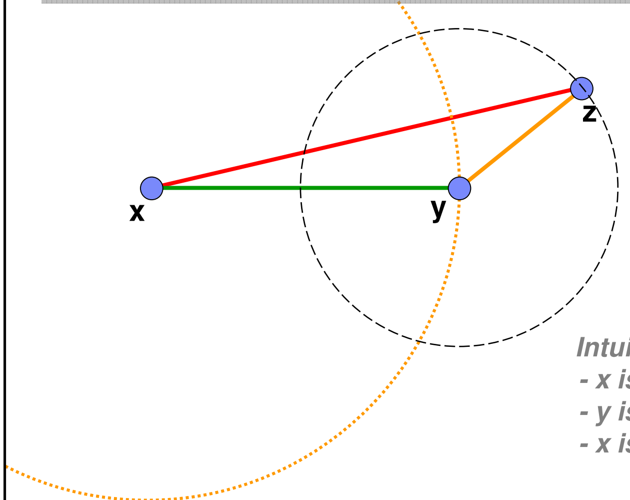
Symmetry: **d(x,y) = d(y,x)**

Triangle Inequality: **d(x,z) ≤ d(x,y) + d(y,z)**

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

Triangle Inequality

Triangle Inequality: **d(x,z) ≤ d(x,y) + d(y,z)**

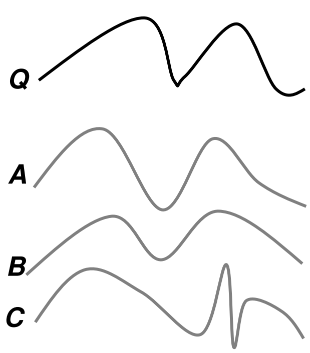


Metric distance functions can exploit the triangle inequality to speed-up search

Intuitively, if:
- x is similar to y and,
- y is similar to z, then,
- x is similar to z too.

Triangle Inequality (Importance)

Triangle Inequality: **d(x,z) ≤ d(x,y) + d(y,z)**



Assume:	*d(Q,bestMatch) = 20*	
and	*d(Q,B) =150*	

Then, since d(A,B)=20

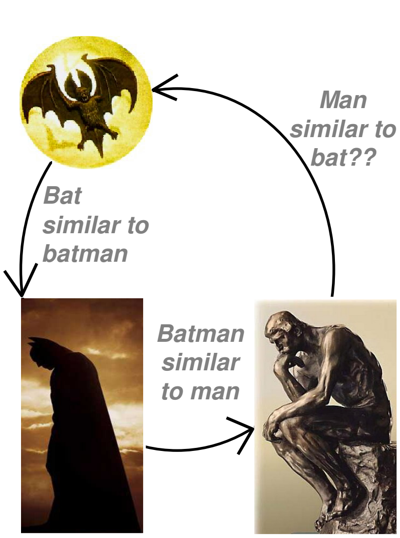
d(Q,A) ≥ d(Q,B) – d(B,A)

d(Q,A) ≥ 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

Non-Metric Distance Functions

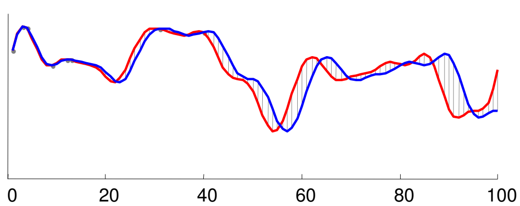


- ***Matching flexibility***
- ***Robustness to outliers***
- ***Stretching in time/space***
- ***Support for different sizes/lengths***

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

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
```

Euclidean Distance (Vectorization)

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

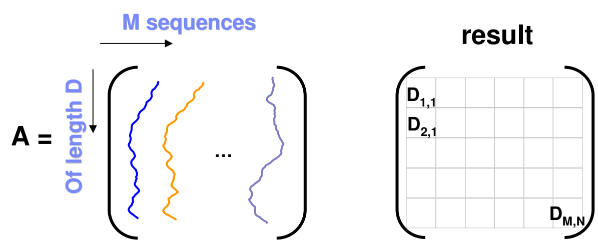
Answer: *No, one can use the following equation to perform matrix computations only…*

$$||A-B|| = \mathrm{sqrt}\,(\,||A||^2 + ||B||^2 - 2*A.B\,)$$

A: DxM matrix

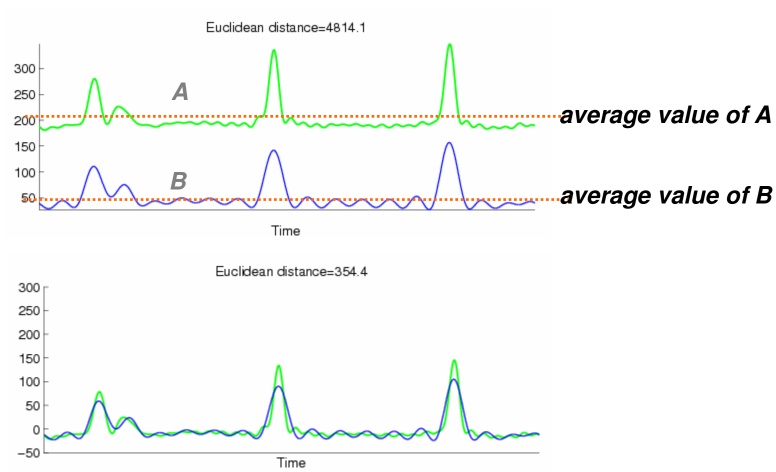
B: DxN matrix

Result is MxN matrix



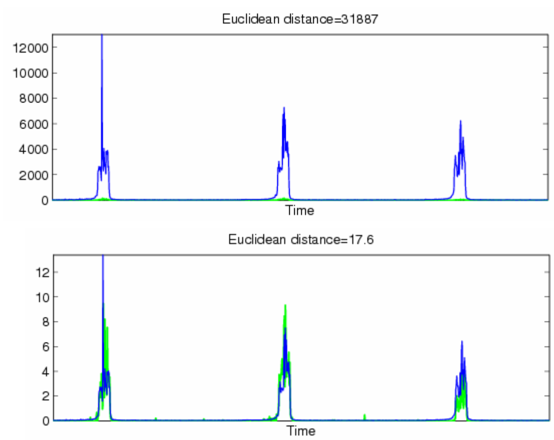

```
aa=sum(a.*a); bb=sum(b.*b); ab=a'*b;
d = sqrt(repmat(aa',[1 size(bb,2)]) + repmat(bb,[size(aa,2) 1]) - 2*ab);
```


Data Preprocessing (Baseline Removal)




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

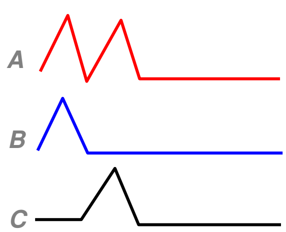
Data Preprocessing (Rescaling)




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

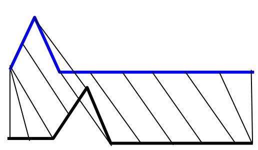

Dynamic Time-Warping (Motivation)

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

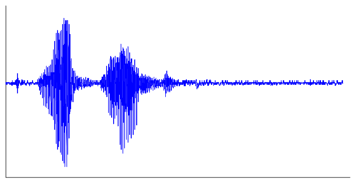
Solution: Allow for compression & decompression in time



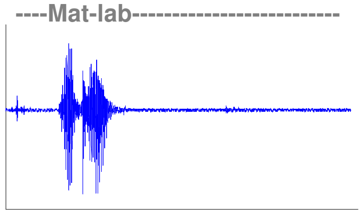
Dynamic Time-Warping

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

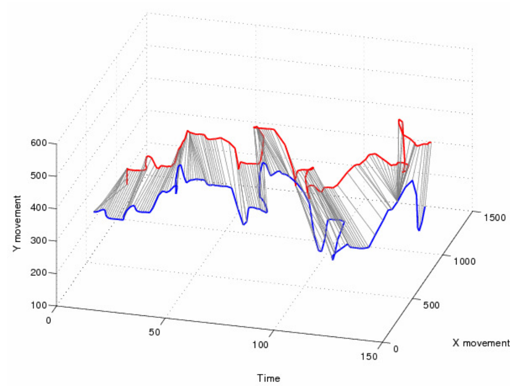
---Maat--llaabb-------------------



----Mat-lab--------------------------



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



Dynamic Time-Warping (how does it work?)

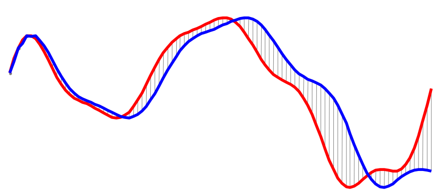
The intuition is that we copy an element multiple times so as to achieve a better matching

Euclidean distance

T1 = [1, 1, 2, 2]

T2 = [1, 2, 2, 2]

d = 1



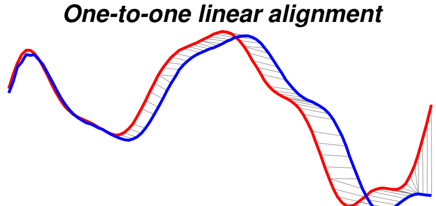
One-to-one linear alignment

Warping distance

T1 = [1, 1, 2, 2]

T2 = [1, 2, 2, 2]

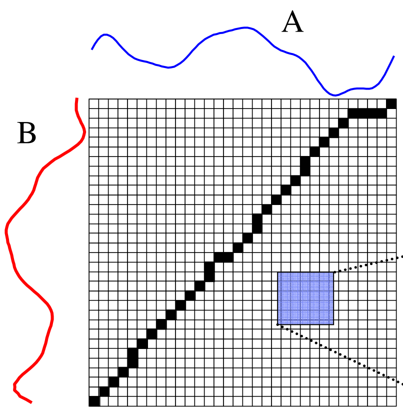
d = 0



One-to-many non-linear alignment

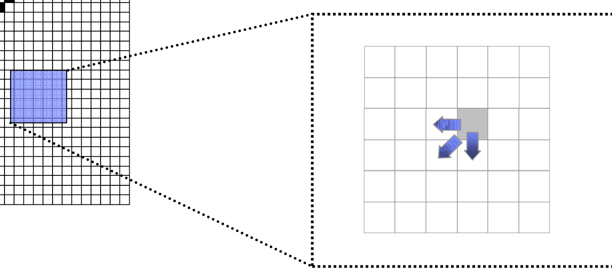
Dynamic Time-Warping (implementation)

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



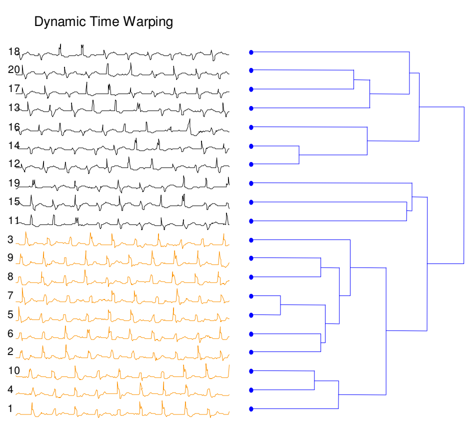
$$c(i,j) = D(A_i,B_j) + \min\{\, c(i-1,j-1)\ ,\ c(i-1,j\,)\ ,\ c(i,j-1)\,\}$$

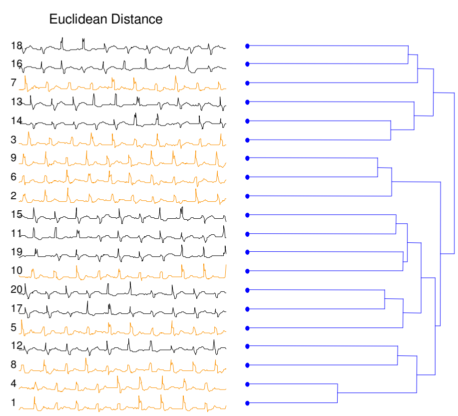
Recursive equation



Dynamic Time-Warping (Examples)

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

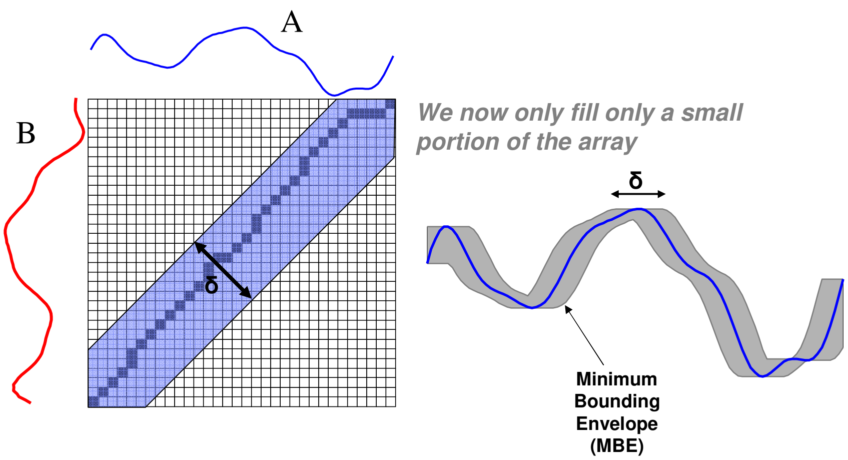




MIT arrhythmia database

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.



Dynamic Time-Warping (restricted warping)

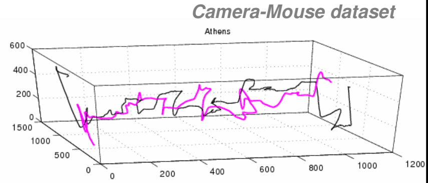
The restriction of the warping path helps:

A. Speed-up execution

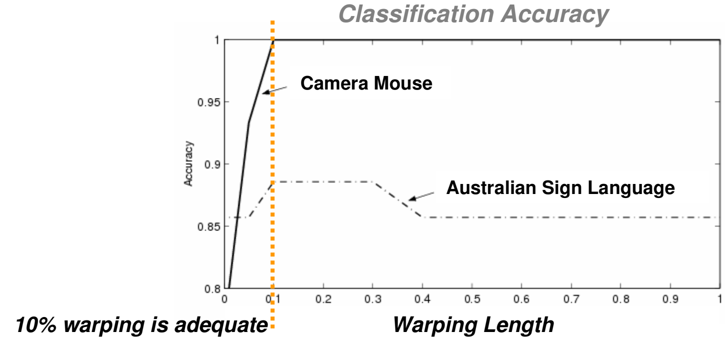
B. Avoid extreme (degenerate) matchings

C. Improve clustering/classification accuracy

Camera-Mouse dataset

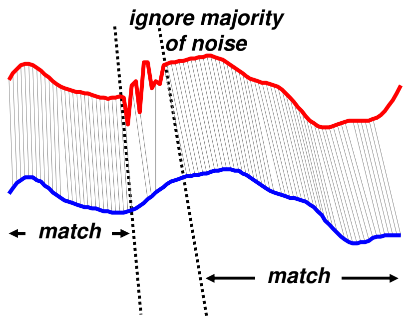


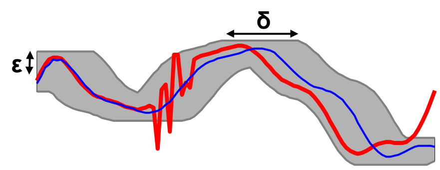
Classification Accuracy



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.



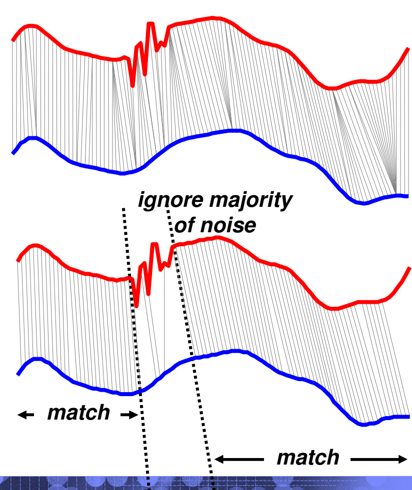


Matching within δ time and ε in space

Everything that is outside the bounding envelope can never be matched

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:

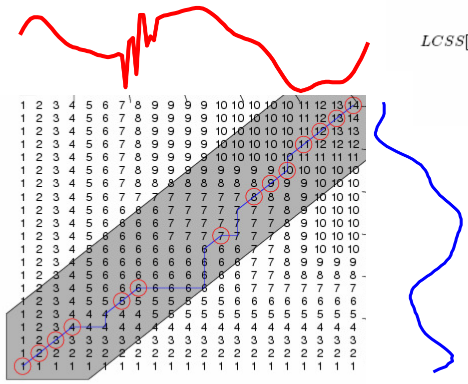
A. Outlying values not matched

B. Distance/Similarity distorted less

C. Constraints in time & space

Longest Common Subsequence (Implementation)

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



$$LCSS[i,j] = \begin{cases} 0 & \text{if } \; i = 0 \\ 0 & \text{if } \; j = 0 \\ 1 + LCSS[i-1, j-1] & \text{if } \; |a_{i,k} - b_{j,k}| < \epsilon, \\ max(LCSS[i-1, j], LCSS[i, j-1]) & \text{otherwise} \end{cases}$$

Can also be expressed as distance

$$D_{LCSS}(A,B) = 1 - \frac{LCSS_{\delta,\epsilon}(A,B)}{\min(n,m) \; or \max(n,m)}$$

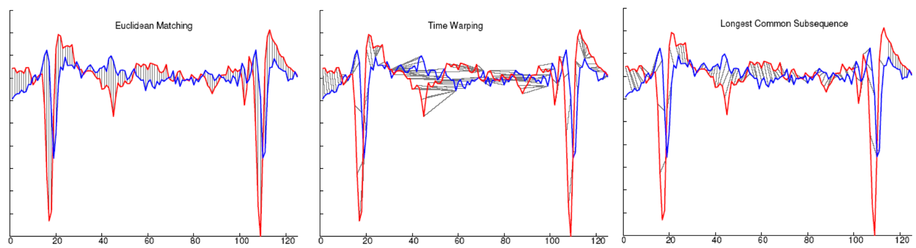
Distance Measure Comparison

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

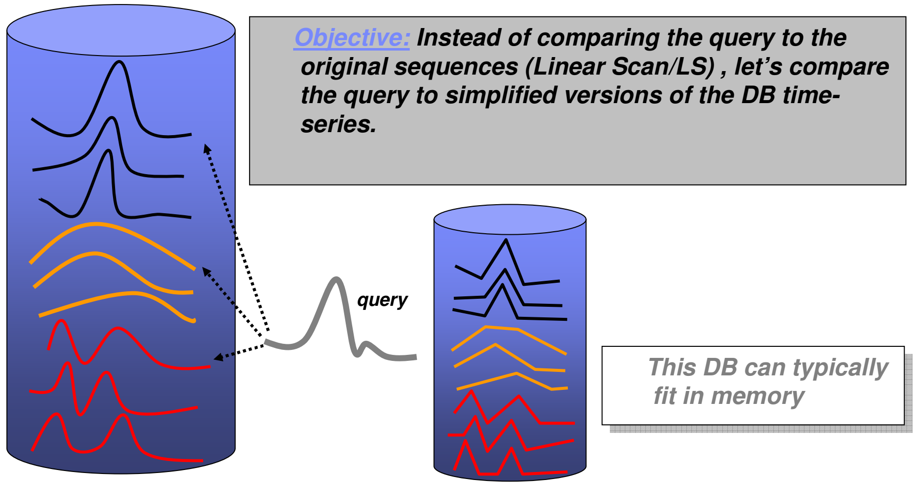
Distance Measure Comparison (Overview)

Method	Complexity	Elastic Matching	One-to-one Matching	Noise Robustness
Euclidean	*O(n)*	×	✓	×
DTW	*O(n*δ)*	✓	×	×
LCSS	*O(n*δ)*	✓	✓	✓



•PART II: Time Series Matching
Lower Bounding

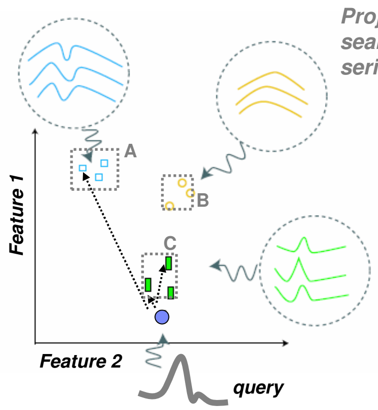
Basic Time-Series Problem Revisited

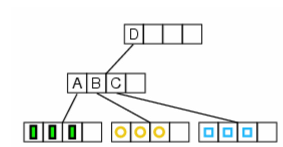


Objective: **Instead of comparing the query to the original sequences (Linear Scan/LS) , let's compare the query to simplified versions of the DB time-series.**

Compression – Dimensionality Reduction

Project all sequences into a new space, and search this space instead (eg project time-series from 100-D space to 2-D space)





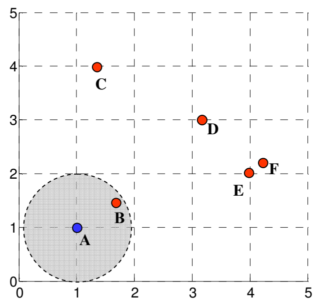
One can also organize the low-dimensional points into a hierarchical 'index' structure. In this tutorial we will not go over indexing techniques.

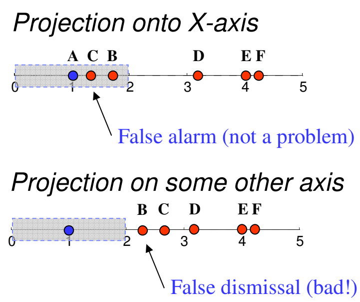
Question: **When searching the original space it is guaranteed that we will find the best match. Does this hold (or under which circumstances) in the new compressed space?**

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→a, B→b etc

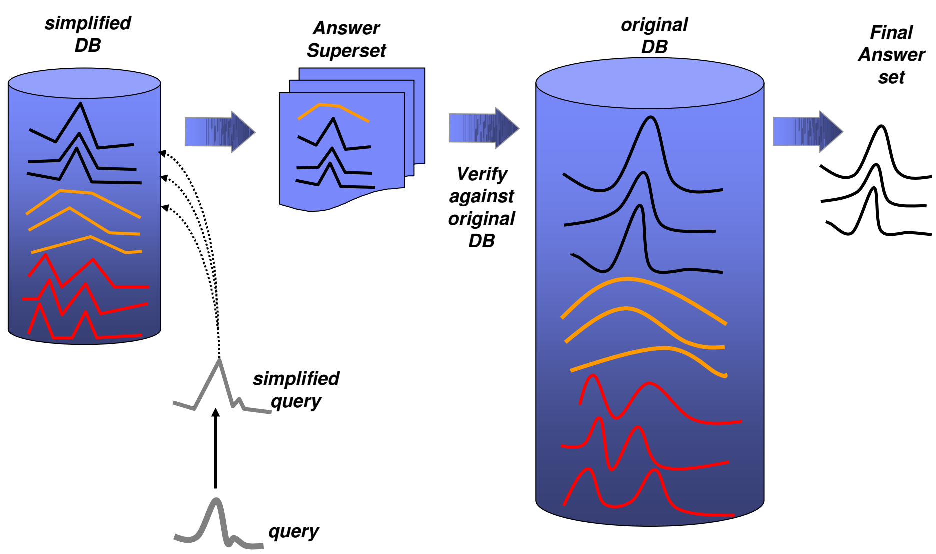
$$D_{LB}(a,b) <= D_{true}(A,B)$$



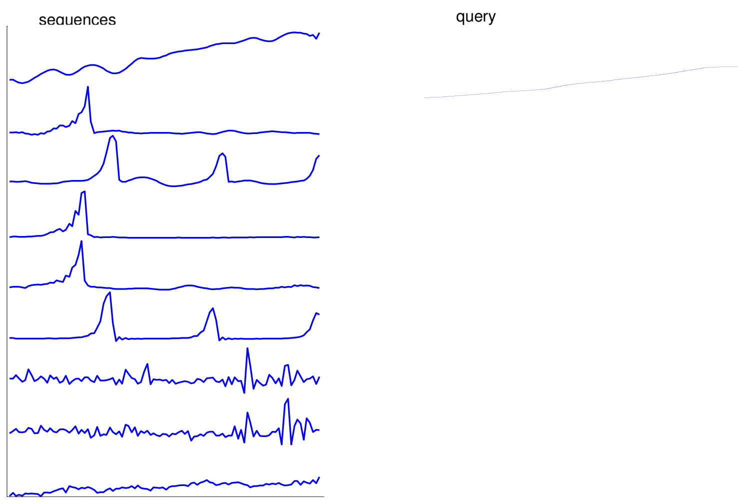


"Find everything within range of 1 from A"

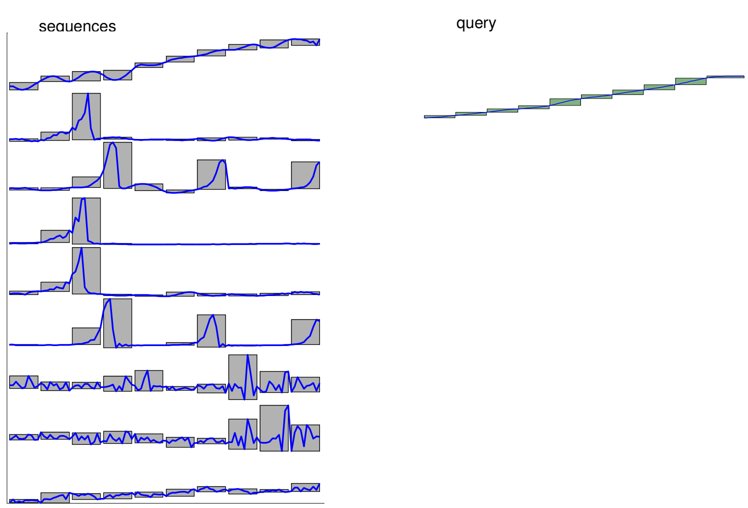
Generic Search using Lower Bounding



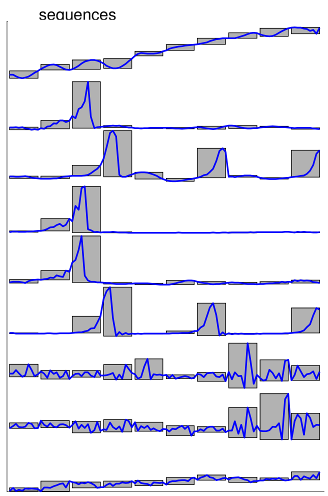
Lower Bounding Example



Lower Bounding Example



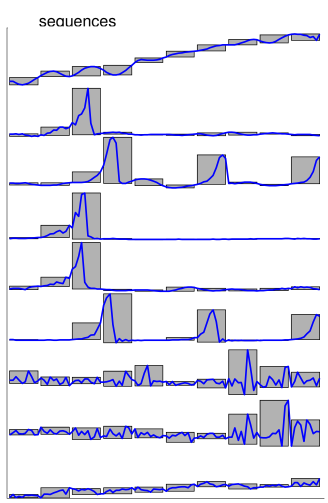
Lower Bounding Example



Lower Bounds

4.6399
37.9032
19.5174
72.1846
67.1436
78.0920
70.9273
63.7253
1.4121

Lower Bounding Example



Lower Bounds	True Distance	
4.6399	46.7790	
37.9032	108.8856	
19.5174	113.5873	
72.1846	104.5062	
67.1436	119.4087	
78.0920	120.0066	
70.9273	111.6011	
63.7253	119.0635	
1.4121	17.2540	BestSoFar

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.

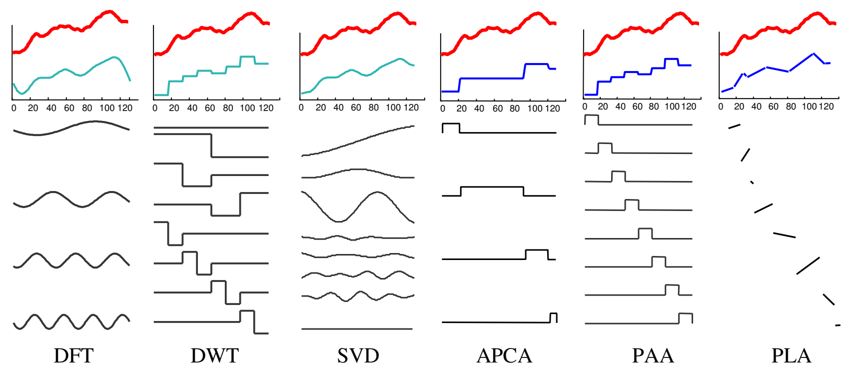


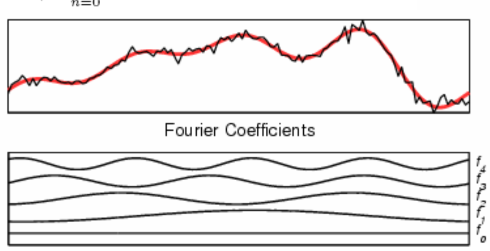
Figure by Eamonn Keogh, 'Time-Series Tutorial'

Fourier Decomposition

Decompose a time-series into sum of sine waves

DFT: $X(f_{k/N}) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) e^{-\frac{j2\pi kn}{N}}, \quad k = 0, 1 \ldots N-1$

IDFT: $x(n) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} X(f_{k/N}) e^{\frac{j2\pi kn}{N}}, \quad k = 0, 1 \ldots N-1$



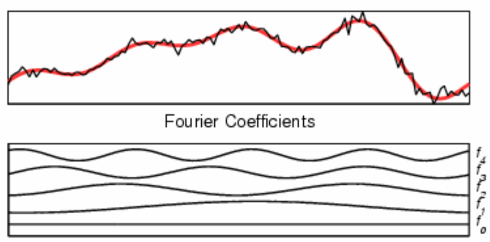
"Every signal can be represented as a superposition of sines and cosines" (…alas nobody believes me…)

Fourier Decomposition

Decompose a time-series into sum of sine waves

DFT: $X(f_{k/N}) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} x(n) e^{-\frac{j2\pi kn}{N}}, \quad k = 0, 1 \ldots N-1$

IDFT: $x(n) = \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} X(f_{k/N}) e^{\frac{j2\pi kn}{N}}, \quad k = 0, 1 \ldots N-1$

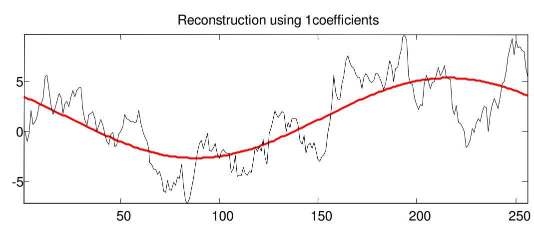



```
fa = fft(a); % Fourier decomposition
fa(5:end) = 0; % keep first 5 coefficients (low frequencies)
reconstr = real(ifft(fa)); % reconstruct signal
```

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

Fourier Decomposition

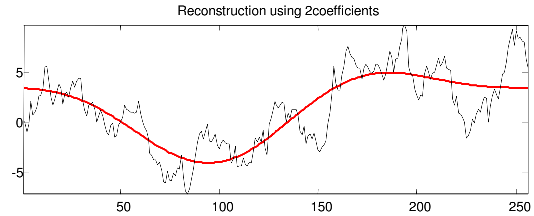
How much space we gain by compressing random walk data?



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

Fourier Decomposition

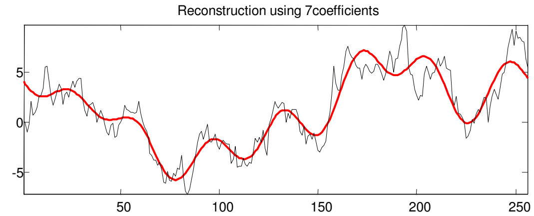
How much space we gain by compressing random walk data?



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

Fourier Decomposition

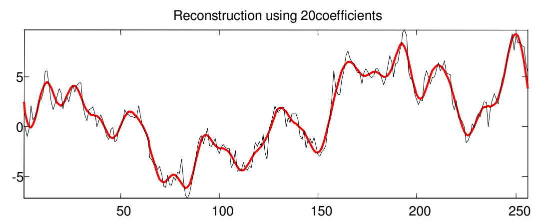
How much space we gain by compressing random walk data?



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

Fourier Decomposition

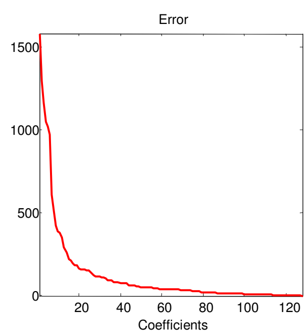
How much space we gain by compressing random walk data?

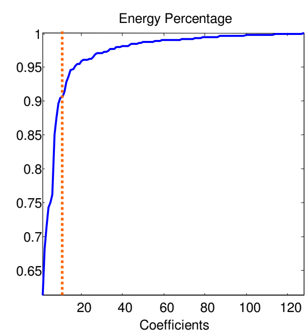


- **1 coeff > 60% of energy**
- **10 coeff > 90% of energy**

Fourier Decomposition

How much space we gain by compressing random walk data?



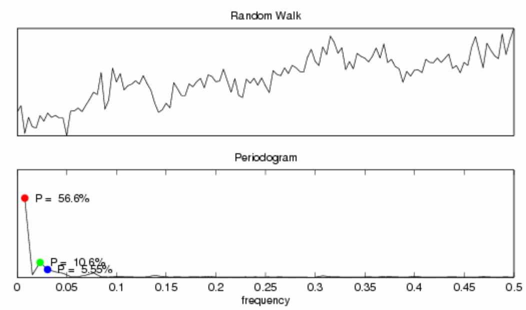


- **1 coeff > 60% of energy**
- **10 coeff > 90% of energy**

Fourier Decomposition

Which coefficients are important?

- *We can measure the 'energy' of each coefficient*
- *Energy = Real(X(f_k))^2 + 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
```

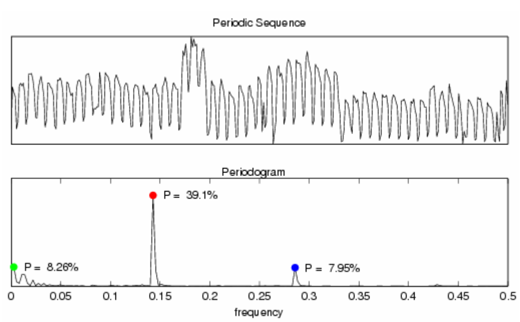
Most of data-mining research uses first k coefficients:

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

Fourier Decomposition

Which coefficients are important?

- *We can measure the 'energy' of each coefficient*
- *Energy = Real(X(f_k))^2 + 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*

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
```

X(f)	
0	
-0.6280 + 0.2709i	keep
-0.4929 + 0.0399i	
-1.0143 + 0.9520i	
0.7200 - 1.0571i	Ignore
-0.0411 + 0.1674i	
-0.5120 - 0.3572i	
0.9860 + 0.8043i	
-0.3680 - 0.1296i	
-0.0517 - 0.0830i	
-0.9158 + 0.4481i	
1.1212 - 0.6795i	
0.2667 + 0.1100i	
0.2667 - 0.1100i	
1.1212 + 0.6795i	
-0.9158 - 0.4481i	
-0.0517 + 0.0830i	
-0.3680 + 0.1296i	
0.9860 - 0.8043i	
-0.5120 + 0.3572i	
-0.0411 - 0.1674i	
0.7200 + 1.0571i	
-1.0143 - 0.9520i	keep
-0.4929 - 0.0399i	
-0.6280 - 0.2709i	

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

Lower Bounding using Fourier coefficients

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

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

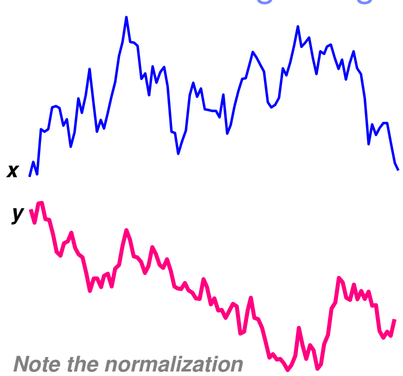
or, that

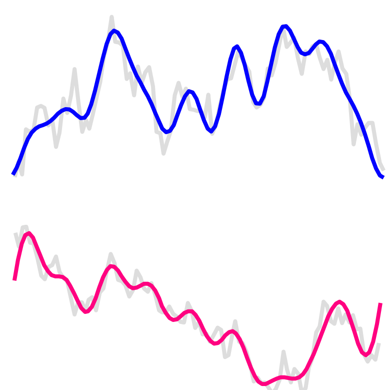
$$\boxed{\sum_{t=0}^{N-1} \|x(t) - y(t)\|^2 = \sum_{k=0}^{N-1} \|X(f_{k/N}) - Y(f_{k/N})\|^2 \quad \textit{Euclidean distance}}$$

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

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

Lower Bounding using Fourier coefficients -Example





Note the normalization

```
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

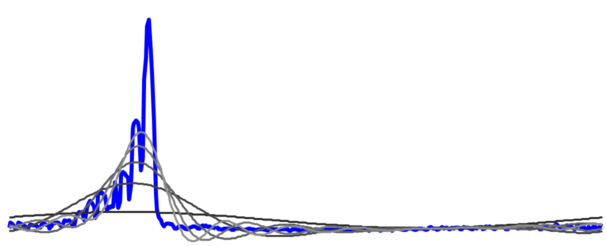
Fourier Decomposition

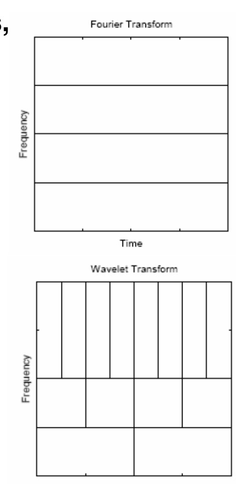
- **O(nlogn) complexity**
- **Tried and tested**
- **Hardware implementations**
- **Many applications:**
  - compression
  - smoothing
  - periodicity detection

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

Wavelets – Why exist?

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

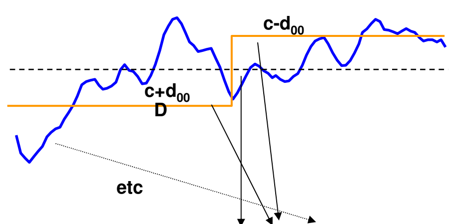




Fourier is good for smooth, random walk data, but not for bursty data or flat data

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)**



Haar coefficients: {c, d_{00}, d_{10}, d_{11},…}

An arithmetic example

X = [9,7,3,5]

Haar = [6,2,1,-1]

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

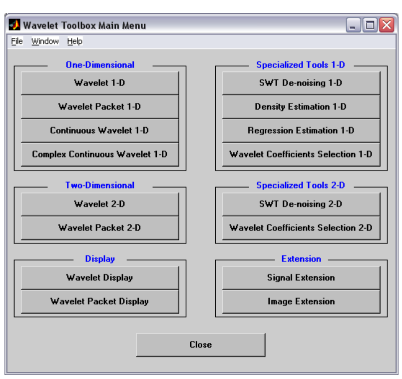
c + d_{00} = 6+2 = 8 = (9+7)/2

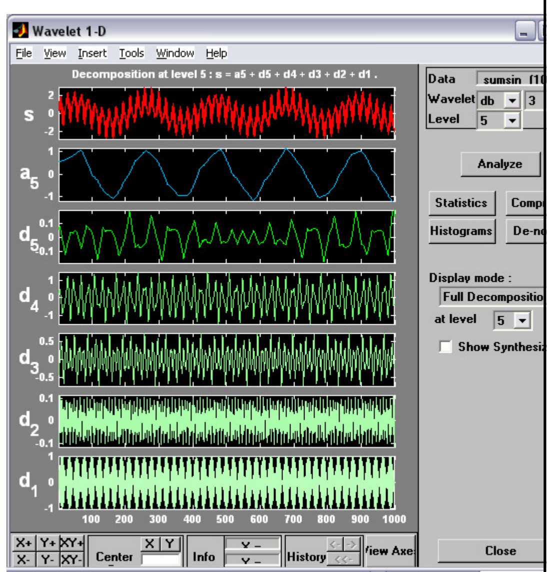
c - d_{00} = 6-2 = 4 = (3+5)/2

etc

Wavelets in Matlab

Specialized Matlab interface for wavelets

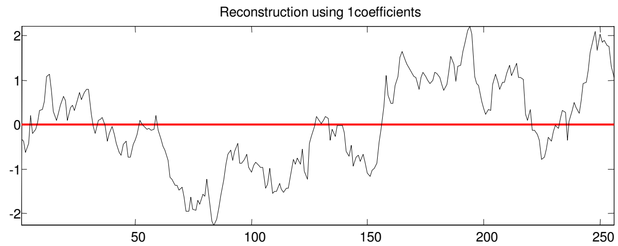




PAA (Piecewise Aggregate Approximation)

also featured as **Piecewise Constant Approximation**

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

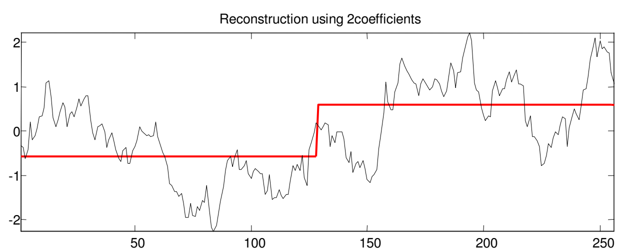


Tutorial | Time-Series with Matlab

PAA (Piecewise Aggregate Approximation)

also featured as **Piecewise Constant Approximation**

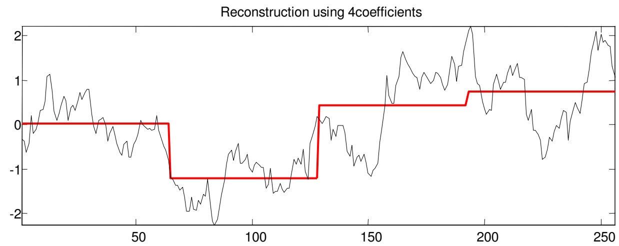
- **Represent time-series as a sequence of segments**
- **Essentially a projection of the Haar coefficients in time**



PAA (Piecewise Aggregate Approximation)

also featured as **Piecewise Constant Approximation**

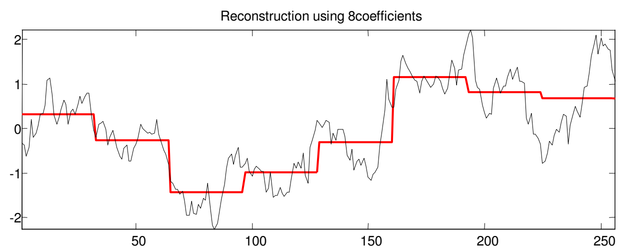
- **Represent time-series as a sequence of segments**
- **Essentially a projection of the Haar coefficients in time**



PAA (Piecewise Aggregate Approximation)

also featured as **Piecewise Constant Approximation**

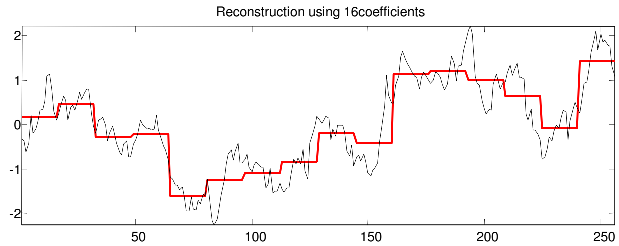
- **Represent time-series as a sequence of segments**
- **Essentially a projection of the Haar coefficients in time**



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**

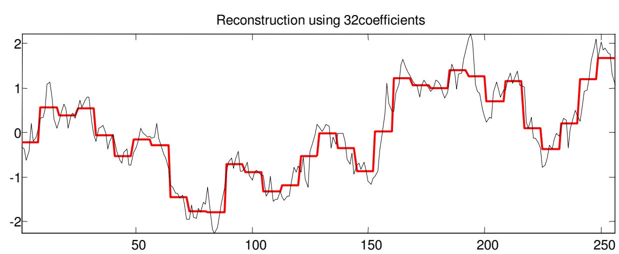


Tutorial | Time-Series with Matlab

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**



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

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

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

2 4

s: 1 2 3 4 5 6 7 8

numCoeff: 4

sN:

1	3	5	7
2	4	6	8

Tutorial | Time-Series with Matlab

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

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

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

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

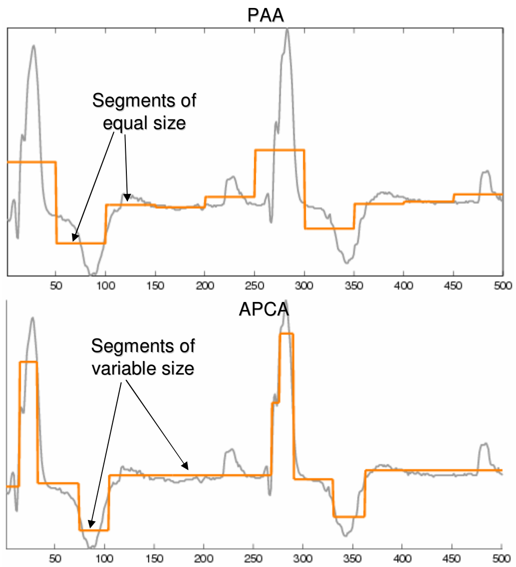
data

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

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]**

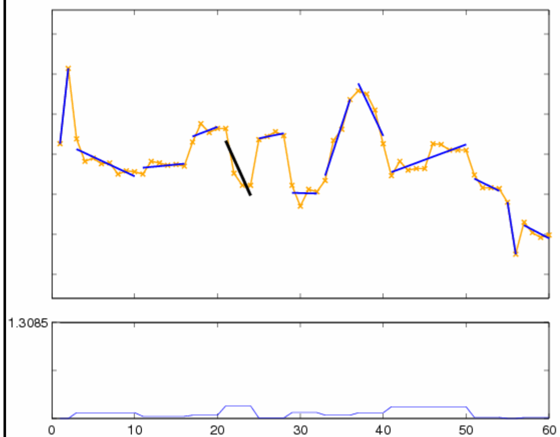
E.g. 10 bits for a sequence of 1024 points

Wavelet Decomposition

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

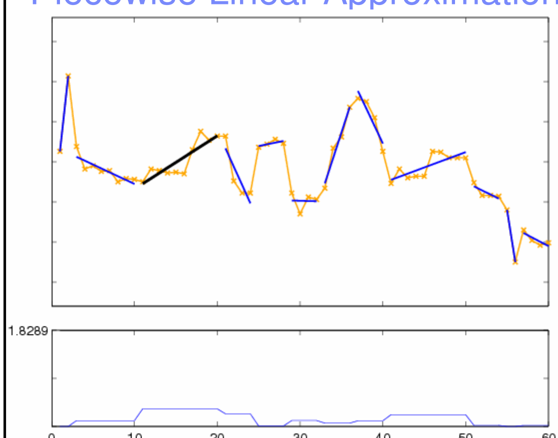
- **Most data-mining research still utilizes Haar wavelets because of their simplicity.**

Piecewise Linear Approximation (PLA)



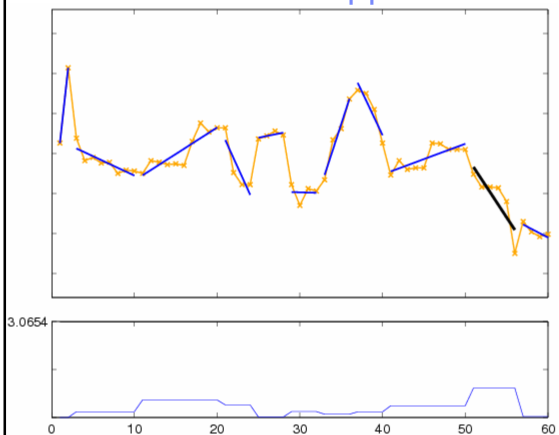
- **Approximate a sequence with multiple linear segments**
- **First such algorithms appeared in *cartography* for map approximation**
- **Many implementations**
  - Optimal
  - Greedy Bottom-Up
  - Greedy Top-down
  - Genetic, etc
- **You can find a bottom-up implementation here:**
  - http://www.cs.ucr.edu/~eamonn/TSDMA/time_series_toolbox/

Piecewise Linear Approximation (PLA)



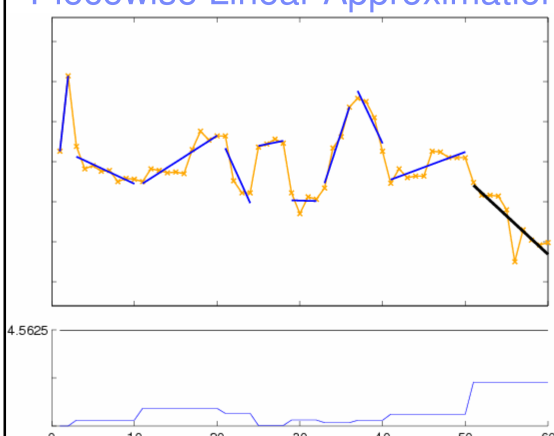
- **Approximate a sequence with multiple linear segments**
- **First such algorithms appeared in *cartography* for map approximation**

Piecewise Linear Approximation (PLA)



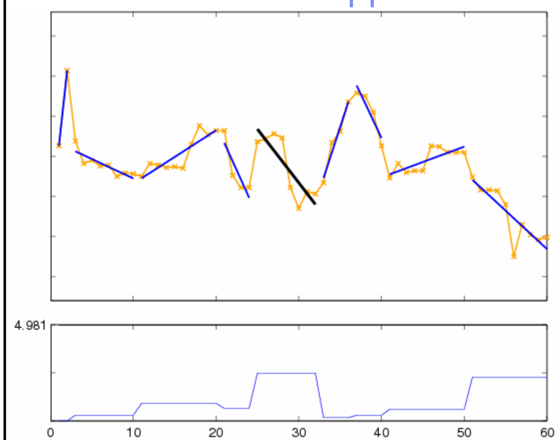
- **Approximate a sequence with multiple linear segments**
- **First such algorithms appeared in *cartography* for map approximation**

Piecewise Linear Approximation (PLA)



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

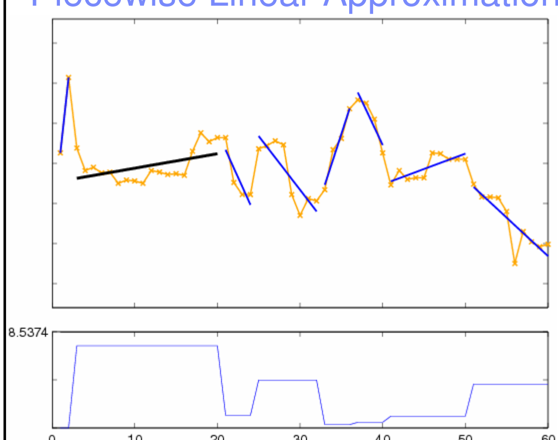
Piecewise Linear Approximation (PLA)



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

Tutorial | Time-Series with Matlab

Piecewise Linear Approximation (PLA)



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

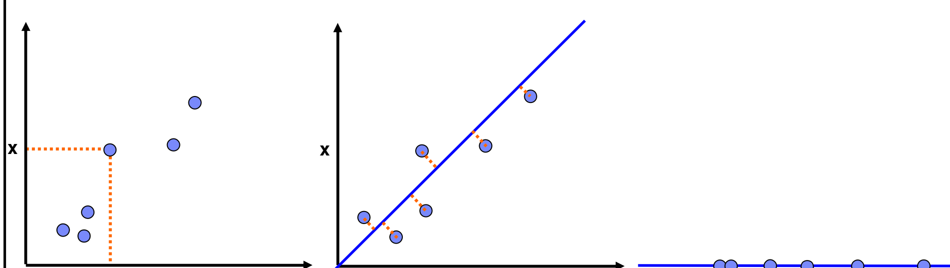
Piecewise Linear Approximation (PLA)

- O(nlogn) complexity for "bottom up" algorithm
- Incremental computation possible
- Provable error bounds
- Applications for:
  - Image / signal simplification
  - Trend detection

- Visually not very smooth or pleasing.

Singular Value Decomposition (SVD)

- **SVD attempts to find the 'optimal' basis for describing a <u>set</u> of multidimensional points**
- **Objective: Find the axis ('directions') that describe better the data variance**



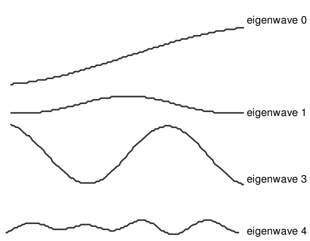
We need 2 numbers (x,y) for every point

Now we can describe each point with 1 number, their projection on the line

New axis and position of points (after projection and rotation)

Singular Value Decomposition (SVD)

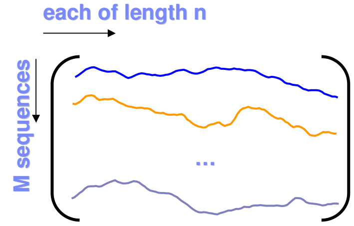
- **Each time-series is essentially a multidimensional point**
- **Objective: Find the 'eigenwaves' (basis) whose linear combination describes best the sequences. Eigenwaves are data-dependent.**



A linear combination of the eigenwaves can produce any sequence in the database

$$A_{Mxn} = U_{Mxr} * \Sigma_{rxr} * V^T_{nxr}$$

Factoring of data array into 3 matrices




```
[U,S,V] = svd(A)
```

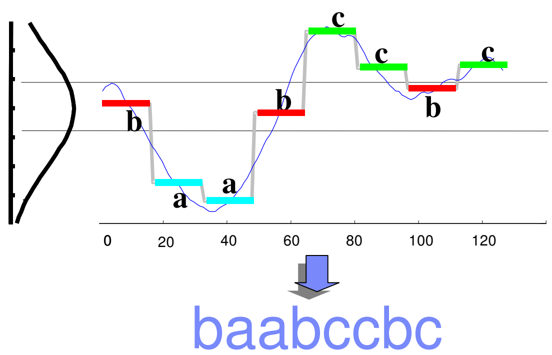

Singular Value Decomposition

- **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

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

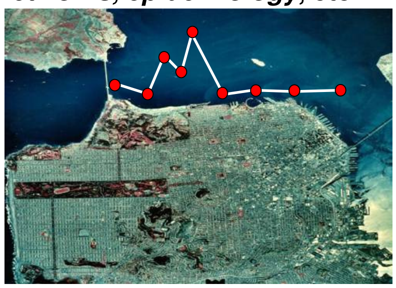
Symbolic Approximations

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

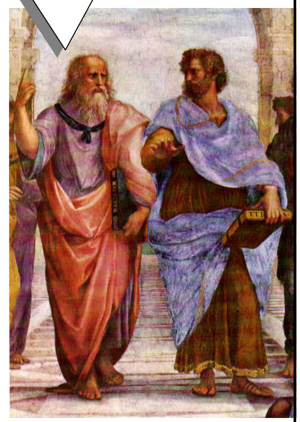
- **Number of regions (alphabet length) can affect the quality of result**

Multidimensional Time-Series

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



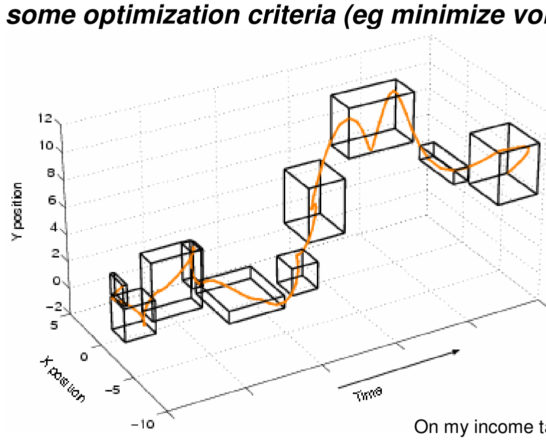
- **Let's see how to approximate 2D trajectories with** *Minimum Bounding Rectangles*



Aristotle

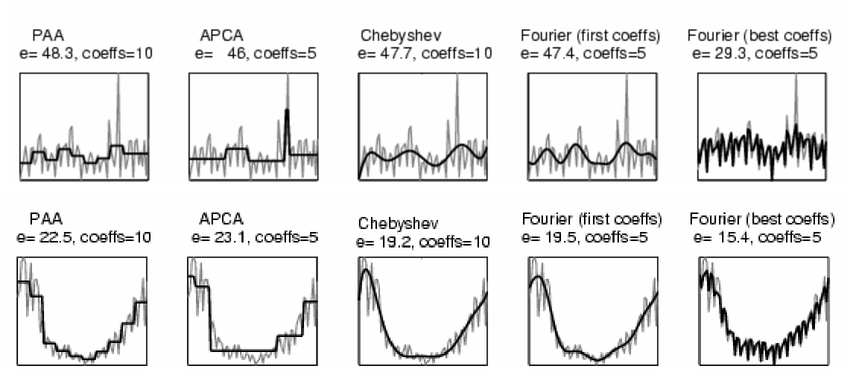
Multidimensional MBRs

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

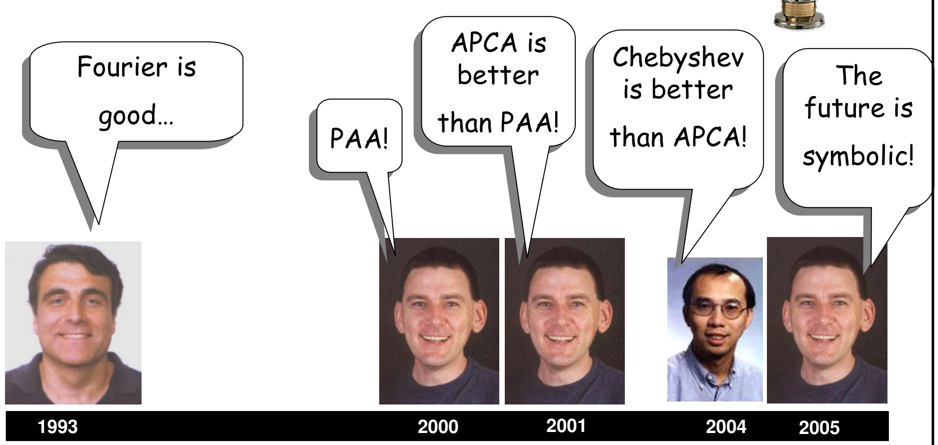


On my income tax 1040 it says "Check this **box** if you are blind." I wanted to put a check mark about three inches away.
- Tom Lehrer

Comparison of different Dim. Reduction Techniques



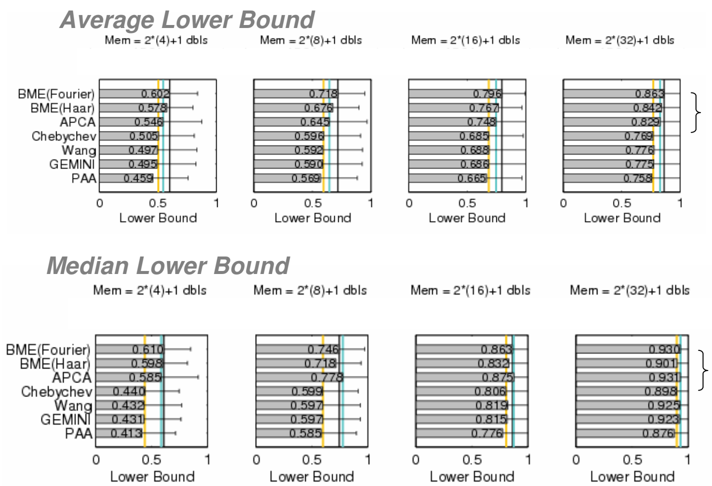
So which dimensionality reduction is the best?



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

Comparisons

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



A. No approach is better on all datasets

B. Best coeff. techniques can offer tighter bounds

C. Choice of compression depends on application

Note: similar results also reported by Keogh in SIGKDD02

•PART II: Time Series Matching

Lower Bounding the DTW and LCSS

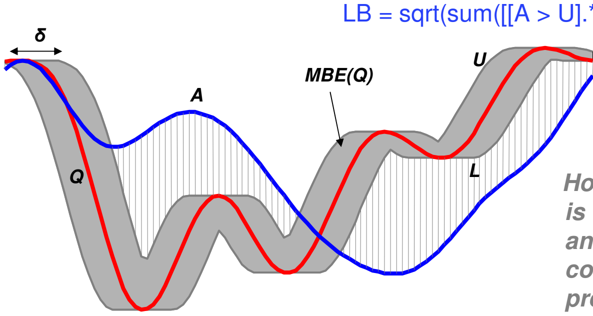
Lower Bounding the Dynamic Time Warping

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:* ***$D(MBE(Q)_\delta, A) < DTW(Q,A)$***

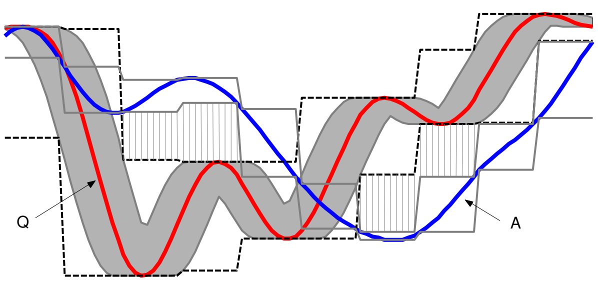
LB = sqrt(sum([[A > U].* [A-U]; [A < L].* [L-A]].^2));

One Matlab command!



However, this representation is uncompressed. Both MBE and the DB sequence can be compressed using any of the previously mentioned techniques.

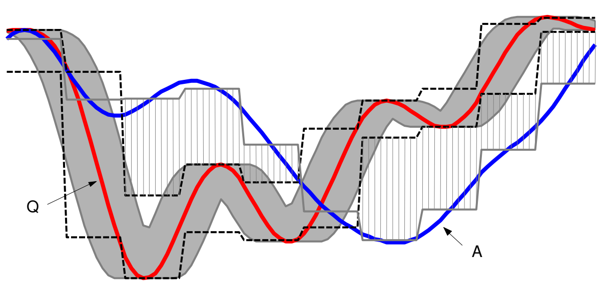
Lower Bounding the Dynamic Time Warping



LB by Keogh

approximate MBE and sequence using MBRs

LB = 13.84



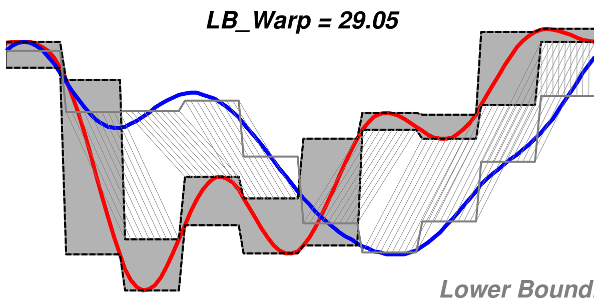
LB by Zhu and Shasha

approximate MBE and sequence using PAA

LB = 25.41

Lower Bounding the Dynamic Time Warping

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

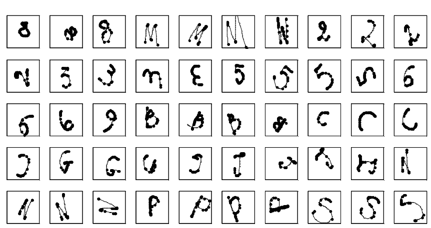


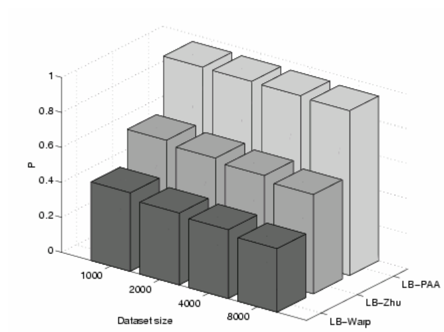
Lower Bounding approaches for DTW, will typically yield at least an order of magnitude speed improvement compared to the naïve approach.

Let's compare the 3 LB approaches:

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-Keogh***

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

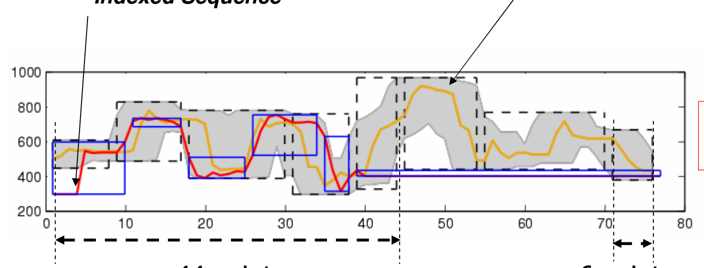
Upper Bounding the LCSS

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

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

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

$LCSS(MBE_Q,A) >= LCSS(Q,A)$



44 points + 6 points

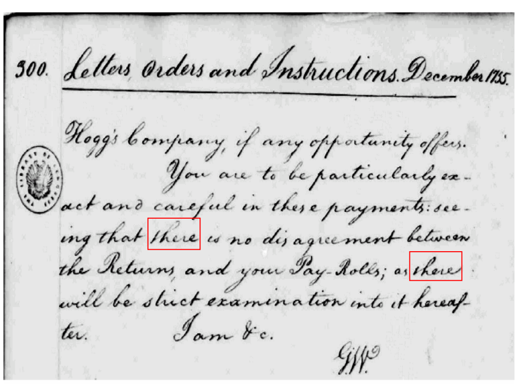
Sim.=50/77 = 0.64

LCSS Application – Image Handwriting

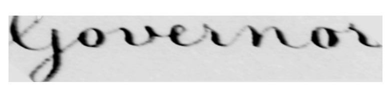
- ***Library of Congress has 54 million manuscripts (20TB of text)***
- ***Increasing interest for automatic transcribing***

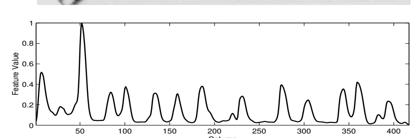
Word annotation:

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



George Washington Manuscript





Features:

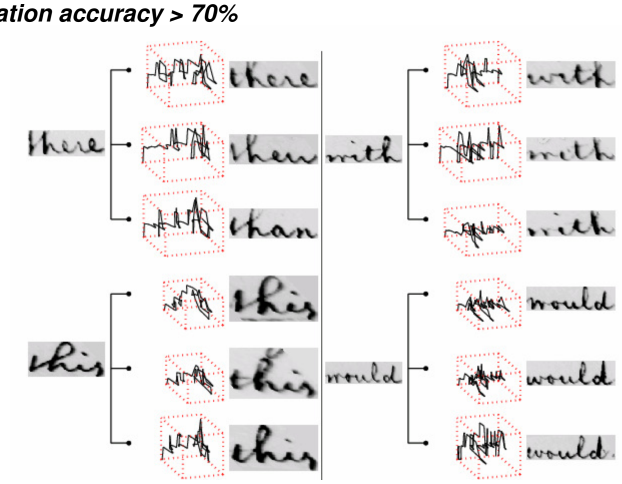
- Black pixels / column
- Ink-paper transitions/ col , etc

LCSS Application – Image Handwriting

Utilized 2D time-series (2 features)

Returned 3-Nearest Neighbors of following words

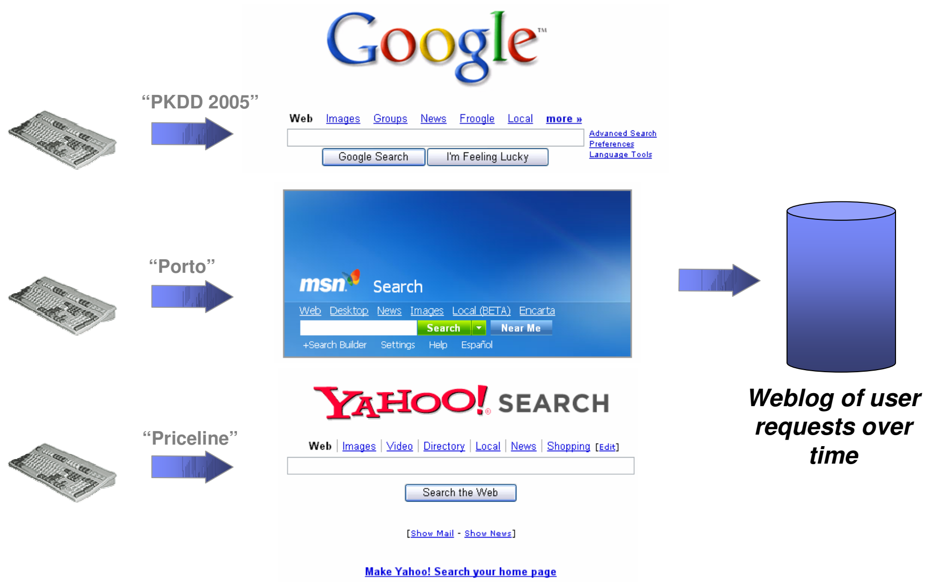
Classification accuracy > 70%



•PART II: Time Series Analysis

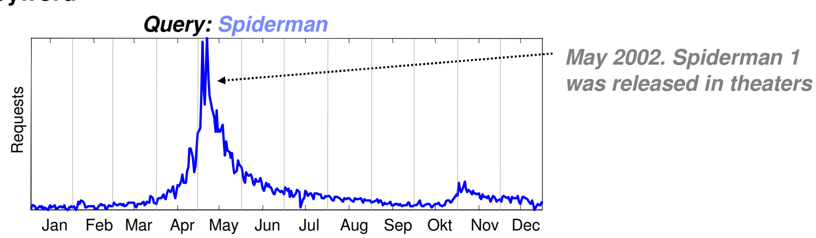
Test Case and Structural Similarity Measures

Analyzing Time-Series Weblogs



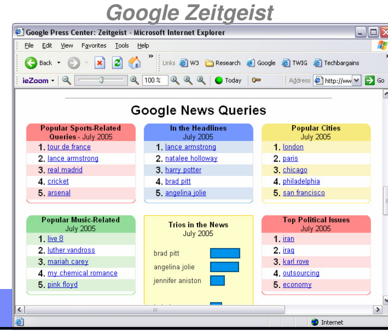
Weblog Data Representation

Record aggregate information, eg, number of requests per day for each keyword



- ***Capture trends and periodicities***
- ***Privacy preserving***

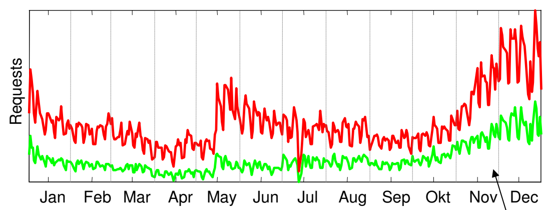
Google Zeitgeist



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)*



Query: *xbox*

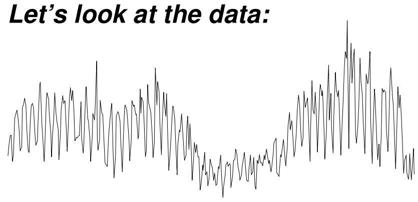
Query: *ps2*

Game consoles are more popular closer to Christmas

Matching of Weblog data

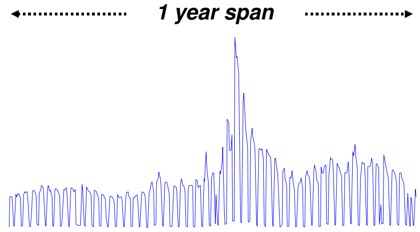
Use Euclidean distance to match time-series. But which dimensionality reduction technique to use?

Let's look at the data:



Query "Bach"

1 year span



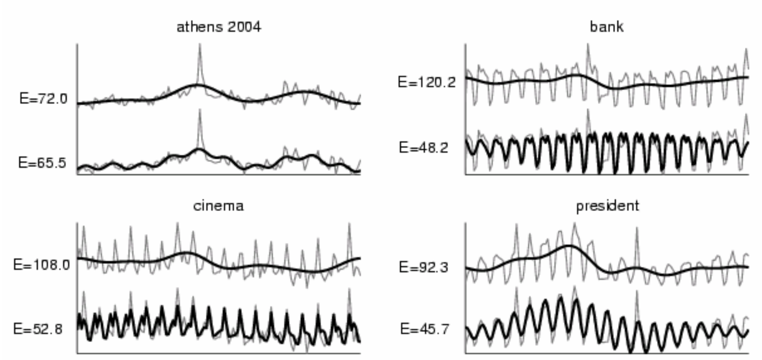
Query "stock market"

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:

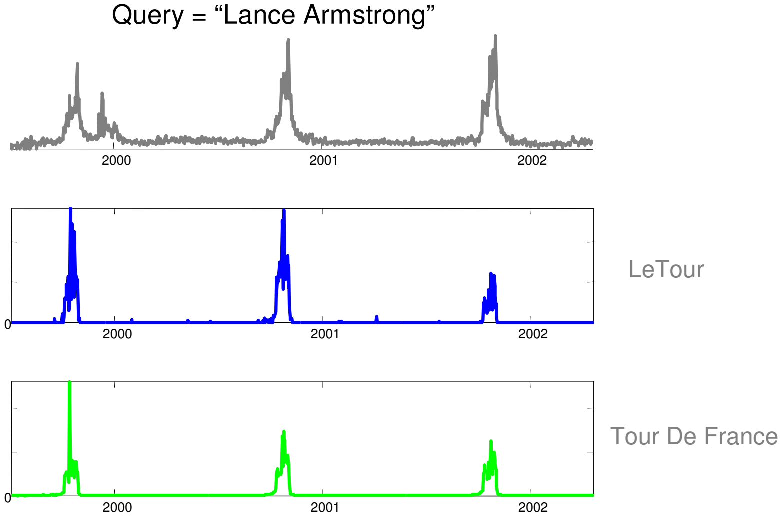
First Fourier Coefficients vs Best Fourier Coefficients



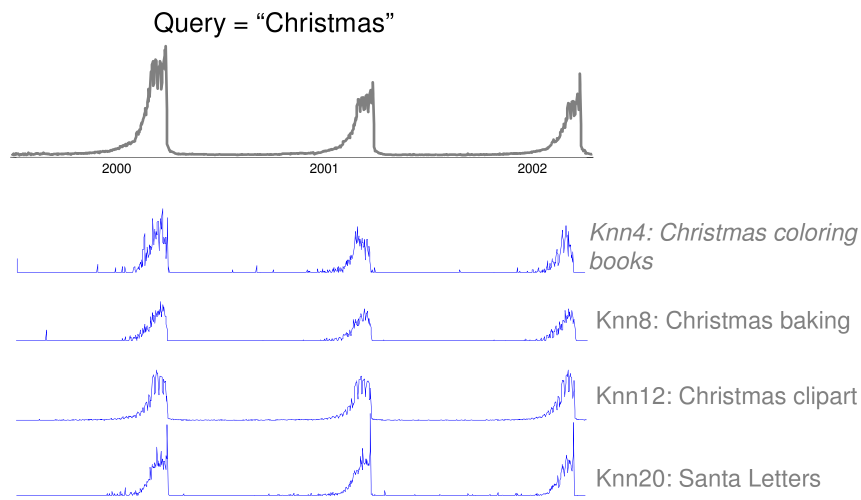
Using the best coefficients, provides a very high quality approximation of the original time-series

Matching results I

Query = "Lance Armstrong"



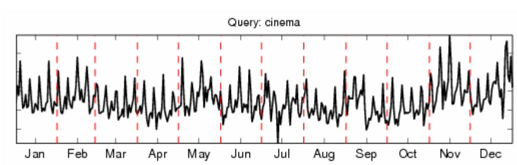
Matching results II



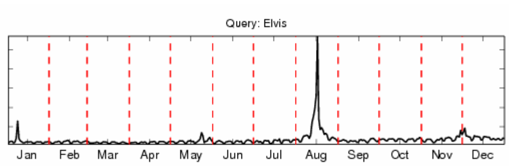
Finding Structural Matches

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.***

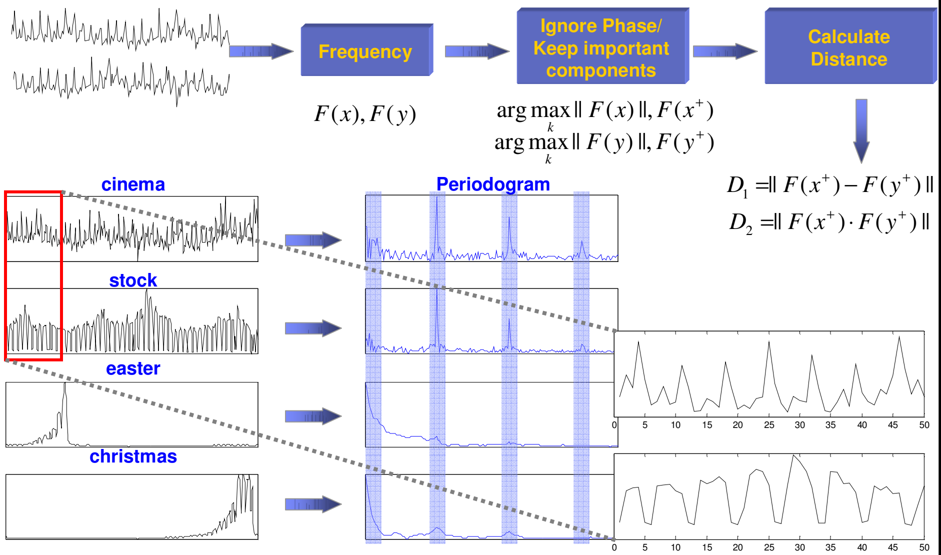


Query “cinema”. Weakly periodicity. Peak of period every Friday.



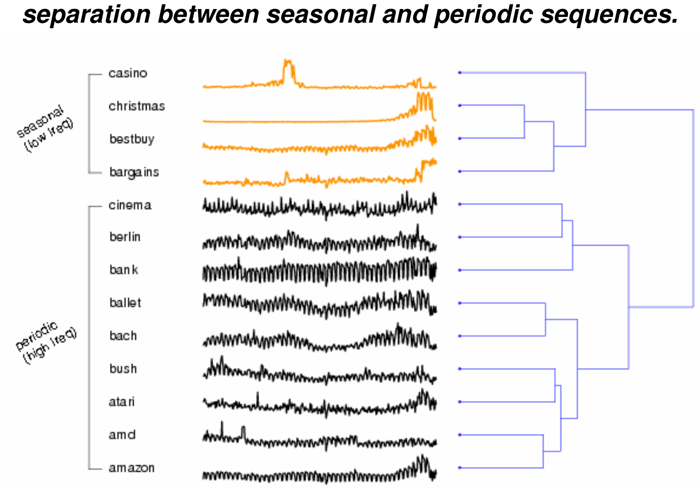
Query “Elvis”. Burst in demand on 16th August. Death anniversary of Elvis Presley

Periodic Matching



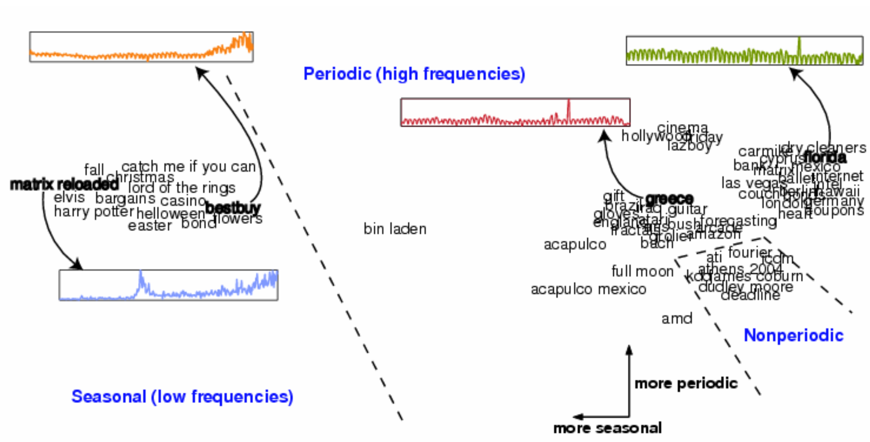
Matching Results with Periodic Measure

Now we can discover more flexible matches. We observe a clear separation between seasonal and periodic sequences.



Matching Results with Periodic Measure

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

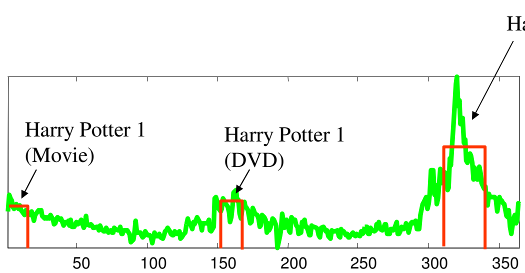


Matching Based on Bursts

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'*

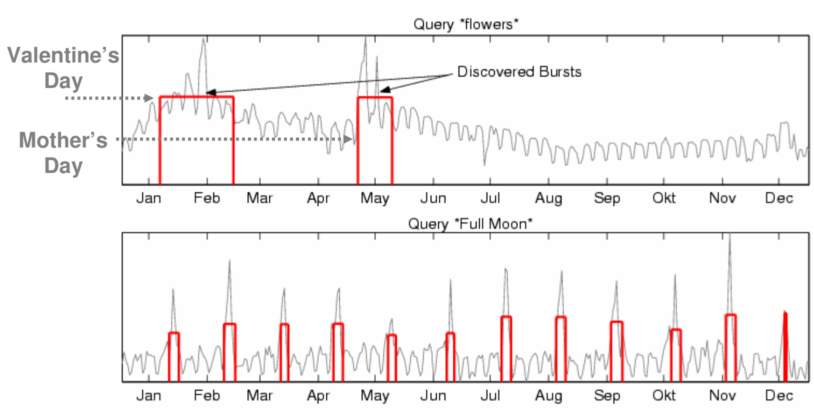


2002: Harry Potter demand

Burst Detection

Burst detection is similar to anomaly detection.

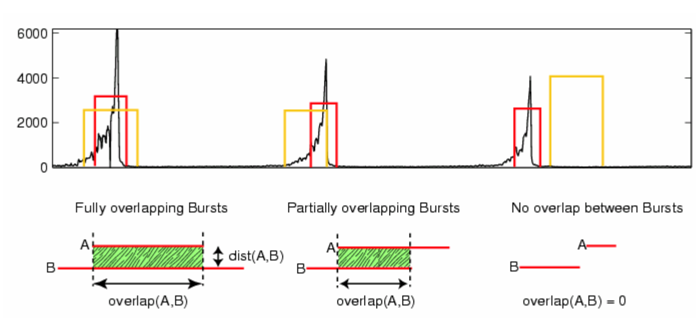
- *Create distribution of values (eg gaussian model)*
- *Any value that deviates from the observed distribution (eg more than 3 std) can be considered as burst.*



Query-by-burst

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

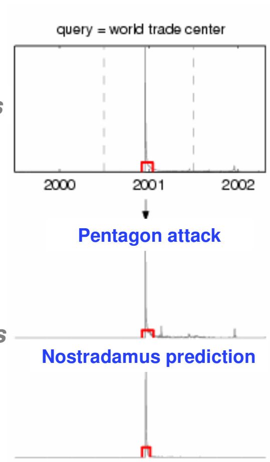
1. ***Find burst regions in given query***
2. ***Represent query bursts as time segments***
3. ***Find which sequences in DB have overlapping burst regions.***



Query-by-burst Results

Queries

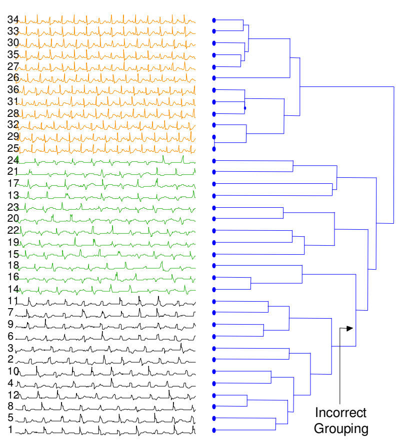
Matches



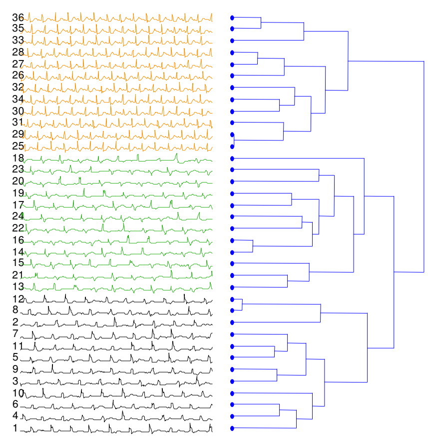
Structural Similarity Measures

Periodic similarity achieves high clustering/classification accuracy in ECG data

DTW

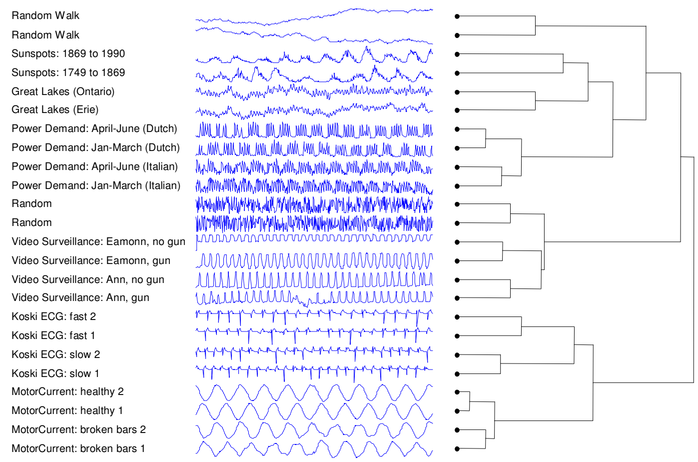


Periodic Measure



Structural Similarity Measures

Periodic similarity is a very powerful visualization tool.



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)***

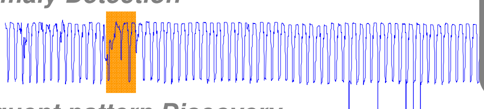
Query: Which stocks exhibited trading bursts during 9/11 attacks?

Conclusion

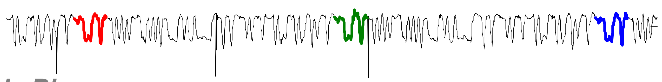
The traditional shape matching measures cannot address all time-series matching problems and applications.
Structural distance measures can provide more flexibility.

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

- *Anomaly Detection*



- *Frequent pattern Discovery*



- *Rule Discovery*
- *etc*