# CS6220: DATA MINING TECHNIQUES

## Mining Sequential and Time Series Data

**Instructor: Yizhou Sun**

yzsun@ccs.neu.edu

March 30, 2016

# Announcement

- About course project
  - You can gain bonus points
- Call for code contribution
  - Sign-up one or several algorithm to implement: wiki link soon
  - Java
  - With a “toy” dataset
  - Clear documentation
  - Clear readme
  - 1 point for each algorithm if approved

# Methods to Learn

| | Matrix Data | Text Data | Set Data | Sequence Data | Time Series | Graph & Network | Images |
|---|---|---|---|---|---|---|---|
| **Classification** | Decision Tree; Naïve Bayes; Logistic Regression SVM; kNN | | | HMM* | | Label Propagation* | Neural Network |
| **Clustering** | K-means; hierarchical clustering; DBSCAN; Mixture Models; kernel k-means* | PLSA | | | | SCAN*; Spectral Clustering | |
| **Frequent Pattern Mining** | | | Apriori; FP-growth | **GSP**; PrefixSpan* | | | |
| **Prediction** | Linear Regression | | | | **Autoregression** | Recommenda tion | |
| **Similarity Search** | | | | | **DTW** | P-PageRank | |
| **Ranking** | | | | | | PageRank | |

# Sequence Data

- What is sequence data?
- Sequential pattern mining
- Summary

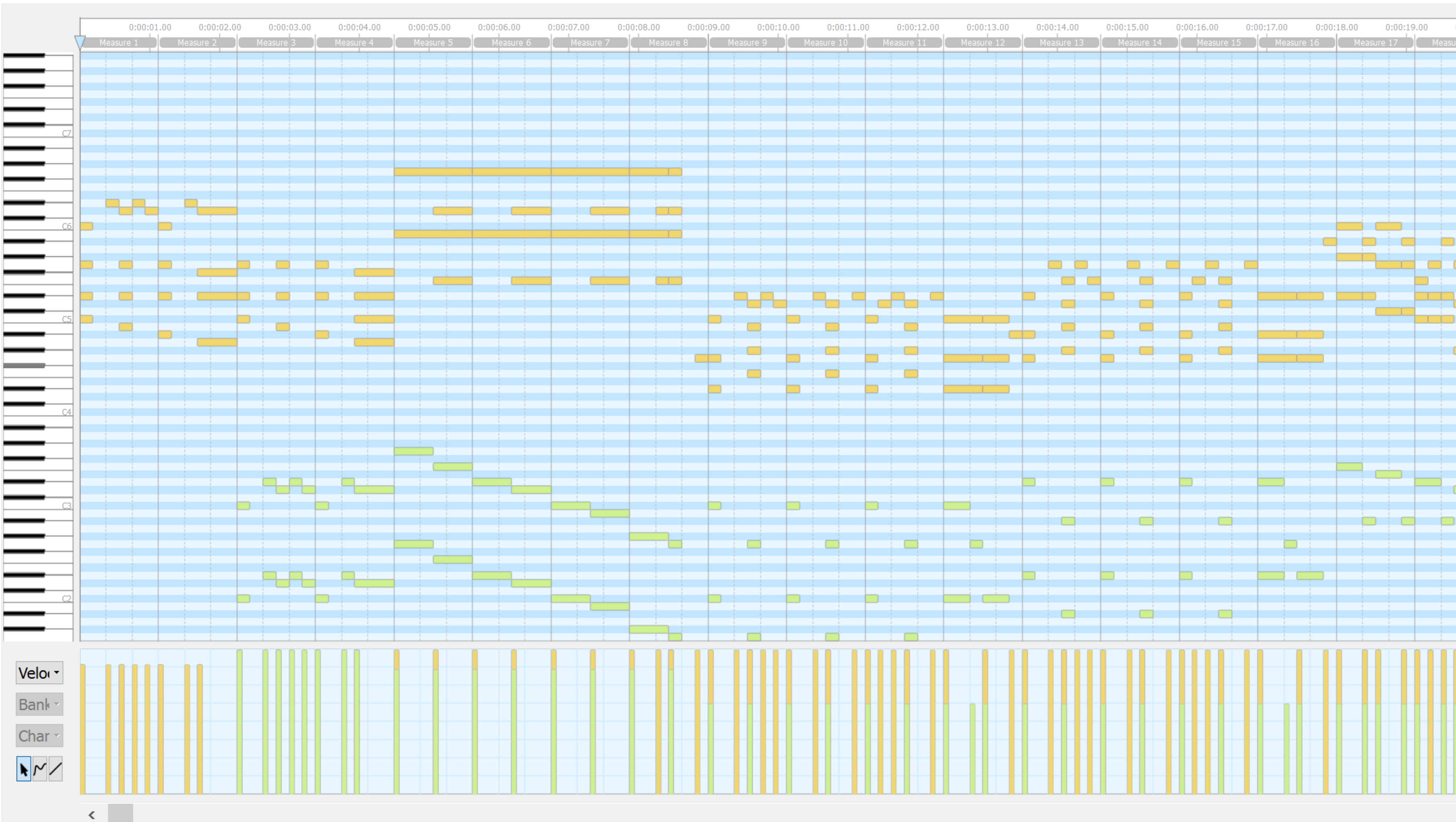
# Sequence Database

- A sequence database consists of sequences of ordered elements or events, recorded with or without a concrete notion of time.

| SID | sequence |
|---|---|
| 10 | <a(abc)(ac)d(cf)> |
| 20 | <(ad)c(bc)(ae)> |
| 30 | <(ef)(ab)(df)cb> |
| 40 | <eg(af)cbc> |

# Example

- Music: midi files



# Sequence Data

- What is sequence data?
- Sequential pattern mining
- Summary

# Sequence Databases & Sequential Patterns

- Transaction databases vs. sequence databases
- Frequent patterns vs. (frequent) sequential patterns
- Applications of sequential pattern mining
  - Customer shopping sequences:
    - First buy computer, then CD-ROM, and then digital camera, within 3 months.
  - Medical treatments, natural disasters (e.g., earthquakes), science & eng. processes, stocks and markets, etc.
  - Telephone calling patterns, Weblog click streams
  - Program execution sequence data sets
  - DNA sequences and gene structures

# What Is Sequential Pattern Mining?

- Given a set of sequences, find the complete set of *frequent* subsequences

A *sequence*: < (ef) (ab) (df) c b >

A *sequence database*

| SID | sequence |
|---|---|
| 10 | <a(abc)(ac)d(cf)> |
| 20 | <(ad)c(bc)(ae)> |
| 30 | <(ef)(ab)(df)cb> |
| 40 | <eg(af)cbc> |

An element may contain a set of items. Items within an element are unordered and we list them alphabetically.

<a(bc)dc> is a *subsequence* of <a(abc)(ac)d(cf)>

Given *support threshold* *min_sup* =2, <(ab)c> is a *sequential pattern*

# Sequence

- Event / element
  - An non-empty set of items, e.g., e=(ab)
- Sequence
  - An ordered list  of events, e.g., $s = < e_1 e_2 \ldots e_l >$
- Length of a sequence
  - The number of instances of items in a sequence
  - The length of < (ef) (ab)  (df) c b > is 8 (Not 5!)

# Subsequence

- Subsequence
  - For two sequences $\alpha =< a_1 a_2 \ldots a_n >$ and $\beta =< b_1 b_2 \ldots b_m >$, $\alpha$ is called a subsequence of $\beta$ if there exists integers $1 \leq j_1 < j_2 < \cdots < j_n \leq m$, such that $a_1 \subseteq b_{j_1}, \ldots, a_n \subseteq b_{j_n}$
- Supersequence
  - If $\alpha$ is a subsequence of $\beta$, $\beta$ is a supersequence of $\alpha$

<a(bc)dc> is a *subsequence* of
<a(abc)(ac)d(cf)>

# Sequential Pattern

- Support of a sequence $\alpha$
  - Number of sequences in the database that are supersequence of $\alpha$
  - $Support_S(\alpha)$
- $\alpha$ is frequent if $Support_S(\alpha) \geq \min\_support$
- A frequent sequence is called sequential pattern
  - l-pattern if the length of the sequence is l

# Example

A *sequence database*

| SID | sequence |
|---|---|
| 10 | <a(abc)(ac)d(cf)> |
| 20 | <(ad)c(bc)(ae)> |
| 30 | <(ef)(ab)(df)cb> |
| 40 | <eg(af)cbc> |

Given *support threshold* *min_sup* =2, <(ab)c> is a *sequential pattern*

# Challenges on Sequential Pattern Mining

- A huge number of possible sequential patterns are hidden in databases
- A mining algorithm should
  - find the complete set of patterns, when possible, satisfying the minimum support (frequency) threshold
  - be highly efficient, scalable, involving only a small number of database scans
  - be able to incorporate various kinds of user-specific constraints

# Sequential Pattern Mining Algorithms

- Concept introduction and an initial Apriori-like algorithm
  - Agrawal & Srikant. Mining sequential patterns, ICDE'95
- Apriori-based method: GSP (Generalized Sequential Patterns: Srikant & Agrawal @ EDBT'96)
- Pattern-growth methods: FreeSpan & PrefixSpan (Han et al.@KDD'00; Pei, et al.@ICDE'01)
- Vertical format-based mining: SPADE (Zaki@Machine Leanining'00)
- Constraint-based sequential pattern mining (SPIRIT: Garofalakis, Rastogi, Shim@VLDB'99; Pei, Han, Wang @ CIKM'02)
- Mining closed sequential patterns: CloSpan (Yan, Han & Afshar @SDM'03)

# The Apriori Property of Sequential Patterns

- A basic property: Apriori (Agrawal & Sirkant'94)
  - If a sequence S is not frequent
  - Then none of the super-sequences of S is frequent
  - E.g, <hb> is infrequent → so do <hab> and <(ah)b>

| Seq. ID | Sequence |
|---|---|
| 10 | <(bd)cb(ac)> |
| 20 | <(bf)(ce)b(fg)> |
| 30 | <(ah)(bf)abf> |
| 40 | <(be)(ce)d> |
| 50 | <a(bd)bcb(ade)> |

Given *support threshold*

*min_sup* =2

# GSP—Generalized Sequential Pattern Mining

- GSP (Generalized Sequential Pattern) mining algorithm
  - proposed by Agrawal and Srikant, EDBT'96
- Outline of the method
  - Initially, every item in DB is a candidate of length-1
  - for each level (i.e., sequences of length-k) do
    - scan database to collect support count for each candidate sequence
    - generate candidate length-(k+1) sequences from length-k frequent sequences using Apriori
  - repeat until no frequent sequence or no candidate can be found
- Major strength: Candidate pruning by Apriori

# Finding Length-1 Sequential Patterns

- Examine GSP using an example
- Initial candidates: all singleton sequences
  - <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>
- Scan database once, count support for candidates

| Cand | Sup |
|---|---|
| **<a>** | **3** |
| **<b>** | **5** |
| **<c>** | **4** |
| **<d>** | **3** |
| **<e>** | **3** |
| **<f>** | **2** |
| ~~<g>~~ | 1 |
| ~~<h>~~ | 1 |

*min_sup* =2

| Seq. ID | Sequence |
|---|---|
| 10 | <(bd)cb(ac)> |
| 20 | <(bf)(ce)b(fg)> |
| 30 | <(ah)(bf)abf> |
| 40 | <(be)(ce)d> |
| 50 | <a(bd)bcb(ade)> |

# GSP: Generating Length-2 Candidates

51 length-2 Candidates

| | <a> | <b> | <c> | <d> | <e> | <f> |
|---|---|---|---|---|---|---|
| <a> | <aa> | <ab> | <ac> | <ad> | <ae> | <af> |
| <b> | <ba> | <bb> | <bc> | <bd> | <be> | <bf> |
| <c> | <ca> | <cb> | <cc> | <cd> | <ce> | <cf> |
| <d> | <da> | <db> | <dc> | <dd> | <de> | <df> |
| <e> | <ea> | <eb> | <ec> | <ed> | <ee> | <ef> |
| <f> | <fa> | <fb> | <fc> | <fd> | <fe> | <ff> |

| | <a> | <b> | <c> | <d> | <e> | <f> |
|---|---|---|---|---|---|---|
| <a> | | <(ab)> | <(ac)> | <(ad)> | <(ae)> | <(af)> |
| <b> | | | <(bc)> | <(bd)> | <(be)> | <(bf)> |
| <c> | | | | <(cd)> | <(ce)> | <(cf)> |
| <d> | | | | | <(de)> | <(df)> |
| <e> | | | | | | <(ef)> |
| <f> | | | | | | |

Without Apriori property, 8*8+8*7/2=92 candidates

Apriori prunes 44.57% candidates

# How to Generate Candidates in General?

- From $L_{k-1}$ to $C_k$
- Step 1: join
  - $s_1$ $and$ $s_2$ can join, if dropping first item in $s_1$ is the same as dropping the last item in $s_2$
  - Examples:
    - <(12)3> join <(2)34> = <(12)34>
    - <(12)3> join <(2)(34)> = <(12)(34)>
- Step 2: pruning
  - Check whether all length k-1 subsequences of a candidate is contained in $L_{k-1}$

# The GSP Mining Process

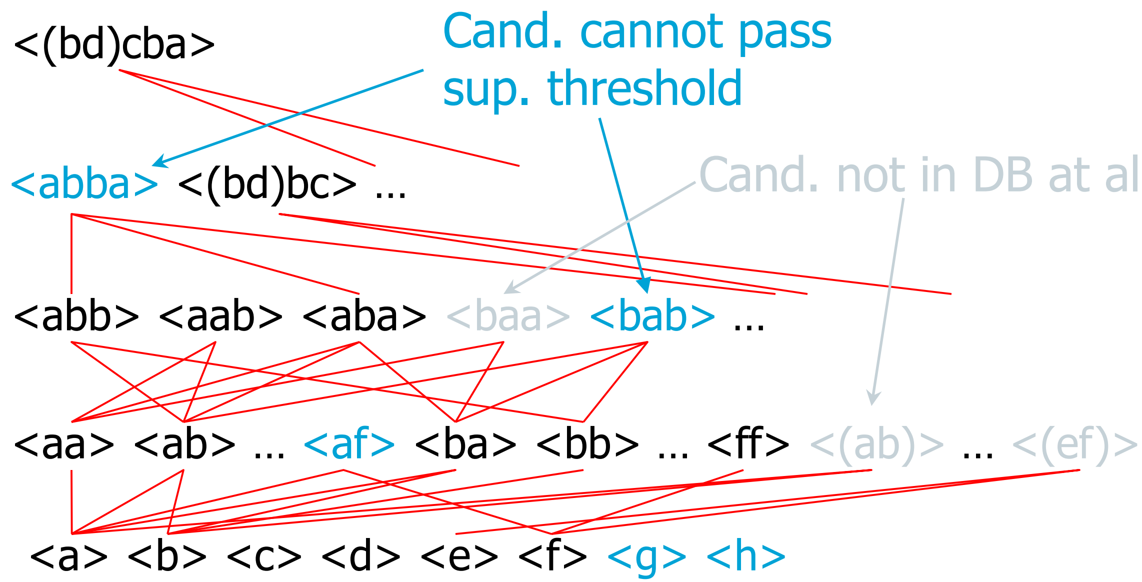
5th scan: 1 cand. 1 length-5 seq. pat.

4th scan: 8 cand. 7 length-4 seq. pat.

3rd scan: 46 cand. 20 length-3 seq. pat. 20 cand. not in DB at all

2nd scan: 51 cand. 19 length-2 seq. pat. 10 cand. not in DB at all

1st scan: 8 cand. 6 length-1 seq. pat.




*min_sup* =2

| Seq. ID | Sequence |
|---|---|
| 10 | <(bd)cb(ac)> |
| 20 | <(bf)(ce)b(fg)> |
| 30 | <(ah)(bf)abf> |
| 40 | <(be)(ce)d> |
| 50 | <a(bd)bcb(ade)> |

# Candidate Generate-and-test: Drawbacks

- A huge set of candidate sequences generated.
  - Especially 2-item candidate sequence.
- Multiple Scans of database needed.
  - The length of each candidate grows by one at each database scan.
- Inefficient for mining long sequential patterns.
  - A long pattern grow up from short patterns
  - The number of short patterns is exponential to the length of mined patterns.

# Sequence Data

- What is sequence data?
- Sequential pattern mining
- Summary

# Summary

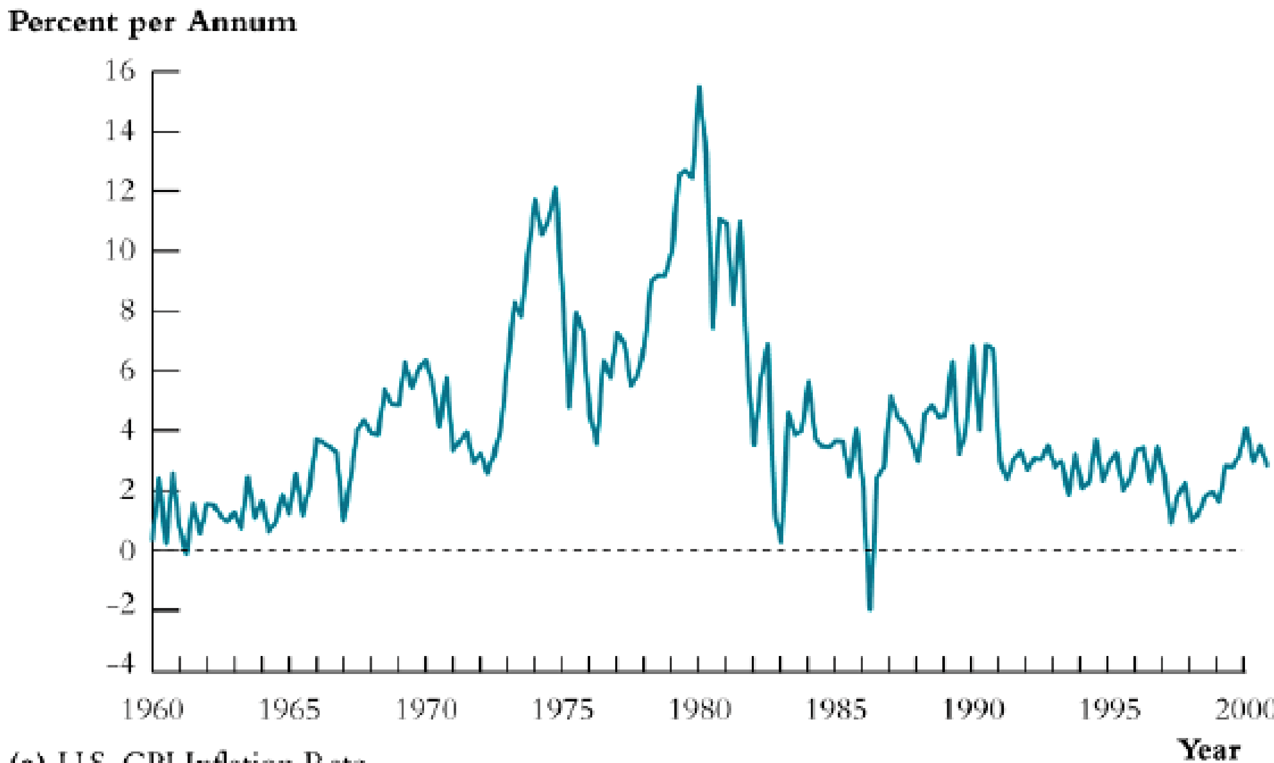
- Sequence data definition and examples
- GSP for sequential pattern mining

# Mining Time Series Data

- Basic Concepts
- Time Series Prediction and Forecasting
- Time Series Similarity Search
- Summary

# Example: Inflation Rate Time Series

**FIGURE 12.1** Inflation and Unemployment in the United States, 1960–1999

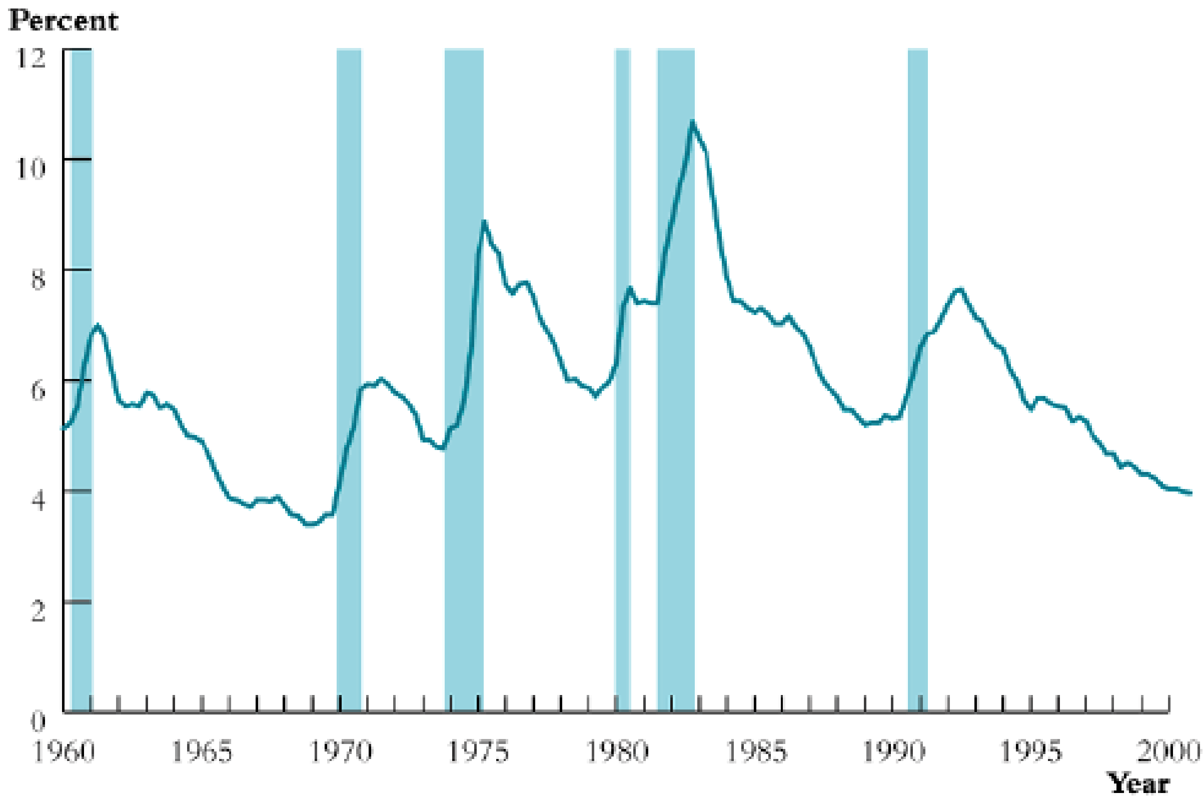



**(a)** U.S. CPI Inflation Rate

Price inflation in the United States (Figure 12.1a) drifted upwards from 1960 until 1980, and then fell sharply during the early 1980s.The unemployment rate in the United States (Figure 12.1b) rises during recessions (the shaded episodes) and falls during expansions.

# Example: Unemployment Rate Time Series

**FIGURE 12.1** Inflation and Unemployment in the United States, 1960–1999



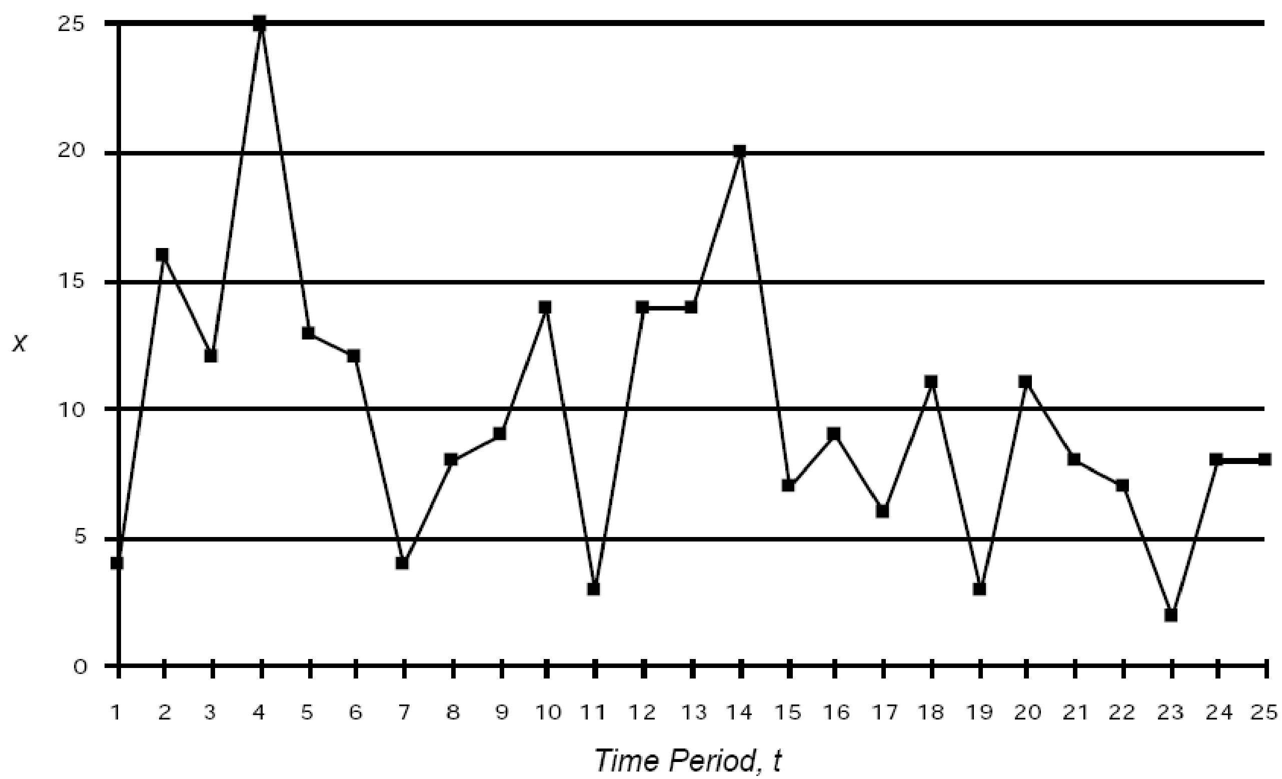

**(b)** U.S. Unemployment Rate

Price inflation in the United States (Figure 12.1a) drifted upwards from 1960 until 1980, and then fell sharply during the early 1980s. The unemployment rate in the United States (Figure 12.1b) rises during recessions (the shaded episodes) and falls during expansions.

# Example: Stock

# Example: Product Sale

| Time | Observations | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 - 10 | 4 | 16 | 12 | 25 | 13 | 12 | 4 | 8 | 9 | 14 |
| 11 - 20 | 3 | 14 | 14 | 20 | 7 | 9 | 6 | 11 | 3 | 11 |
| 20 - 25 | 8 | 7 | 2 | 8 | 8 | 10 | 7 | 16 | 9 | 4 |



# Time Series

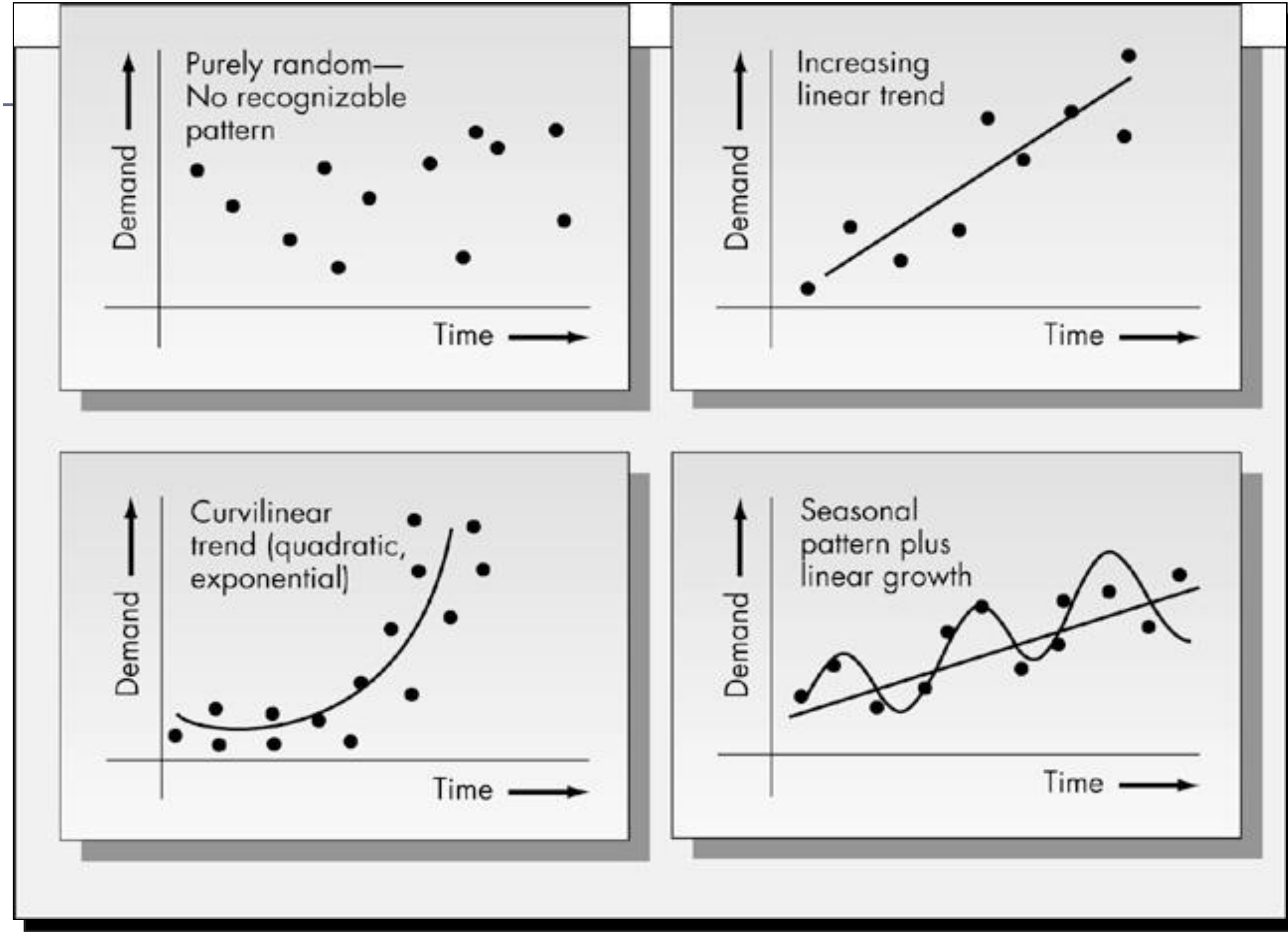
- A **time series** is a sequence of numerical data points, measured typically at successive times, spaced at (often uniform) time intervals
  - *Random variables for a time series are Represented as:*
    - $Y = \{Y_1, Y_2, \dots\}, or$
    - $Y = \{Y_t : t \in T\}, where\ T\ is\ the\ index\ set$
  - An observation of a time series with length N is represent as:
    - $Y = \{y_1, y_2, \dots, y_N\}$

# Mining Time Series Data

- Basic Concepts
- Time Series Prediction and Forecasting
- Time Series Similarity Search
- Summary

# Categories of Time-Series Movements

- Categories of Time-Series Movements (T, C, S, I)
  - Long-term or trend movements (trend curve): general direction in which a time series is moving over a long interval of time
  - Cyclic movements or cycle variations: long term oscillations about a trend line or curve
    - e.g., business cycles, may or may not be periodic
  - Seasonal movements or seasonal variations
    - E.g., almost identical patterns that a time series appears to follow during corresponding months of successive years.
  - Irregular or random movements


Purely random—
No recognizable
pattern
Demand
Time
Increasing
linear trend
Demand
Time
Curvilinear
trend (quadratic,
exponential)
Demand
Time
Seasonal
pattern plus
linear growth
Demand
Time

# Lag, Difference

- The first lag of $Y_t$ is $Y_{t-1}$; the jth lag of $Y_t$ is $Y_{t-j}$
- The first difference of a time series, $\Delta Y_t = Y_t - Y_{t-1}$
  - Sometimes difference in logarithm is used $\Delta\ln(Y_t) = \ln(Y_t) - \ln(Y_{t-1})$

# Example: First Lag and First Difference

**TABLE 12.1** Inflation in the United States in 1999 and the First Quarter of 2000

| Quarter | U.S. CPI | Rate of Inflation at an Annual Rate ($Inf_t$) | First Lag ($Inf_{t-1}$) | Change in Inflation ($\Delta Inf_t$) |
|---|---|---|---|---|
| 1999:I | 164.87 | 1.6 | 2.0 | –0.4 |
| 1999:II | 166.03 | 2.8 | 1.6 | 1.2 |
| 1999:III | 167.20 | 2.8 | 2.8 | 0.0 |
| 1999:IV | 168.53 | 3.2 | 2.8 | 0.4 |
| 2000:I | 170.27 | 4.1 | 3.2 | 0.9 |

# Autocorrelation

- Autocorrelation: the correlation between a time series and its lagged values
  - The first autocorrelation $\rho_1$

$$\operatorname{corr}(Y_t, Y_{t-1}) = \frac{\operatorname{cov}(Y_t, Y_{t-1})}{\sqrt{\operatorname{var}(Y_t)\operatorname{var}(Y_{t-1})}}$$

  - The jth autocorrelation $\rho_j$

$$\operatorname{corr}(Y_t, Y_{t-j}) = \frac{\operatorname{cov}(Y_t, Y_{t-j})}{\sqrt{\operatorname{var}(Y_t)\operatorname{var}(Y_{t-j})}}.$$

**Autocovariance**

# Sample Autocorrelation Calculation

- The jth sample autocorrelation
  - $\hat{\rho}_j = \frac{\widehat{cov}(Y_t, Y_{t-j})}{\widehat{var}(Y_t)}$
  - Where $\widehat{cov}(Y_t, Y_{t-j})$ is calculated as:

$$\frac{1}{T-j-1}\sum_{t=j+1}^{T}(Y_t - \overline{Y}_{j+1,T})(Y_{t-j} - \overline{Y}_{1,T-j})$$

| $Y_t$ | $Y_{t-j}$ |
|---|---|
| $y_{j+1}$ | $y_1$ |
| $y_{j+2}$ | $y_2$ |
| $\vdots$ | $\vdots$ |
| $y_{T-1}$ | $y_{T-j-1}$ |
| $y_T$ | $y_{T-j}$ |

  - i.e., considering two time series: Y(1,...,T-j) and Y(j+1,...,T)

# Example of Autocorrelation

- For inflation and its change

**TABLE 12.2** First Four Sample Autocorrelations of the U.S. Inflation Rate and Its Change, 1960:I–1999:IV

| | Autocorrelation of: | |
|---|---|---|
| Lag | Inflation Rate ($Inf_t$) | Change of Inflation Rate ($\Delta Inf_t$) |
| 1 | 0.85 | −0.24 |
| 2 | 0.77 | −0.27 |
| 3 | 0.77 | 0.32 |
| 4 | 0.68 | −0.06 |

**$\rho_1 = 0.85$, very high: Last quarter's inflation rate contains much information about this quarter's inflation rate**

# Focus on Stationary Time Series

- Stationary is key for time series regression: Future is similar to the past in terms of distribution

A time series $Y_t$ is **stationary** if its probability distribution does not change over time, that is, if the joint distribution of $(Y_{s+1}, Y_{s+2}, \ldots, Y_{s+T})$ does not depend on $s$; otherwise, $Y_t$ is said to be **nonstationary**. A pair of time series, $X_t$ and $Y_t$, are said to be **jointly stationary** if the joint distribution of $(X_{s+1}, Y_{s+1}, X_{s+2}, Y_{s+2}, \ldots, X_{s+T}, Y_{s+T})$ does not depend on $s$. Stationarity requires the future to be like the past, at least in a probabilistic sense.

# Autoregression

- Use past values $Y_{t-1}, Y_{t-2}$, … to predict $Y_t$
  - An *autoregression* is a regression model in which $Y_t$ is regressed against its own lagged values.
  - The number of lags used as regressors is called the *order* of the autoregression.
    - In a ***first order autoregression***, $Y_t$ is regressed against $Y_{t-1}$
    - In a ***pth order autoregression***, $Y_t$ is regressed against $Y_{t-1}, Y_{t-2}, \ldots, Y_{t-p}$

# The First Order Autoregression Model AR(1)

- AR(1) model:

$$Y_t = \beta_0 + \beta_1 Y_{t-1} + u_t$$

- The AR(1) model can be estimated by OLS regression of $Y_t$ against $Y_{t-1}$
- Testing $\beta_1 = 0$ vs. $\beta_1 \neq 0$ provides a test of the hypothesis that $Y_{t-1}$ is not useful for forecasting $Y_t$

# Prediction vs. Forecast

- A *predicted value* refers to the value of *Y* predicted (using a regression) for an observation in the sample used to estimate the regression – this is the usual definition
  - Predicted values are "in sample"
- A *forecast* refers to the value of *Y* forecasted for an observation *not* in the sample used to estimate the regression.
  - Forecasts are forecasts of the future – which cannot have been used to estimate the regression.

# Time Series Regression with Additional Predictors

- So far we have considered forecasting models that use only past values of *Y*
- It makes sense to add other variables (*X*) that might be useful predictors of *Y*, above and beyond the predictive value of lagged values of *Y*:
  - $$Y_t = \beta_0 + \beta_1 Y_{t-1} + \ldots + \beta_p Y_{t-p} + \delta_1 X_{t-1} + \ldots + \delta_r X_{t-r} + u_t$$

# Mining Time Series Data

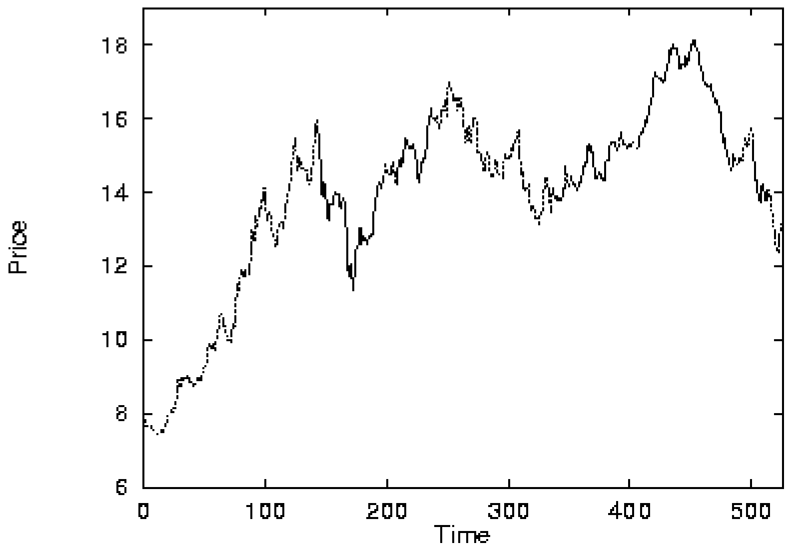
- Basic Concepts
- Time Series Prediction and Forecasting
- Time Series Similarity Search
- Summary

# Why Similarity Search?

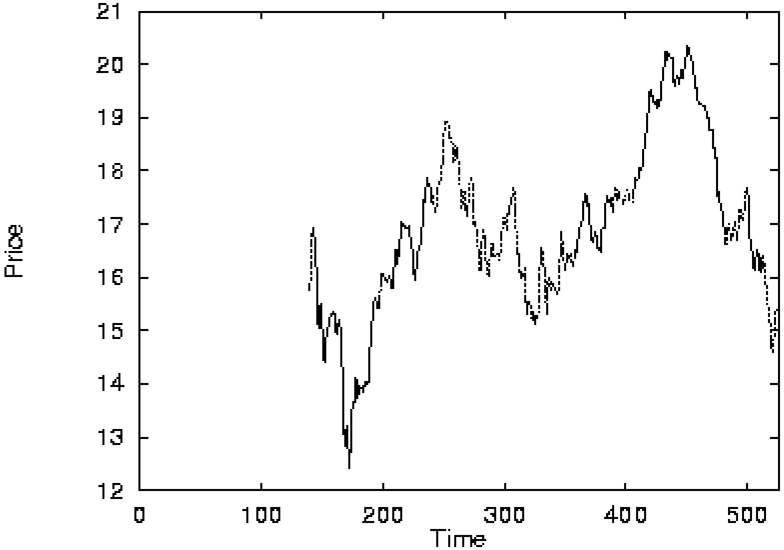
- Wide applications
  - Find a time period with similar inflation rate and unemployment time series?
  - Find a similar stock to Facebook?
  - Find a similar product to a query one according to sale time series?
  - …

# Example

VanEck International Fund




Fidelity Selective Precious Metal and Mineral Fund



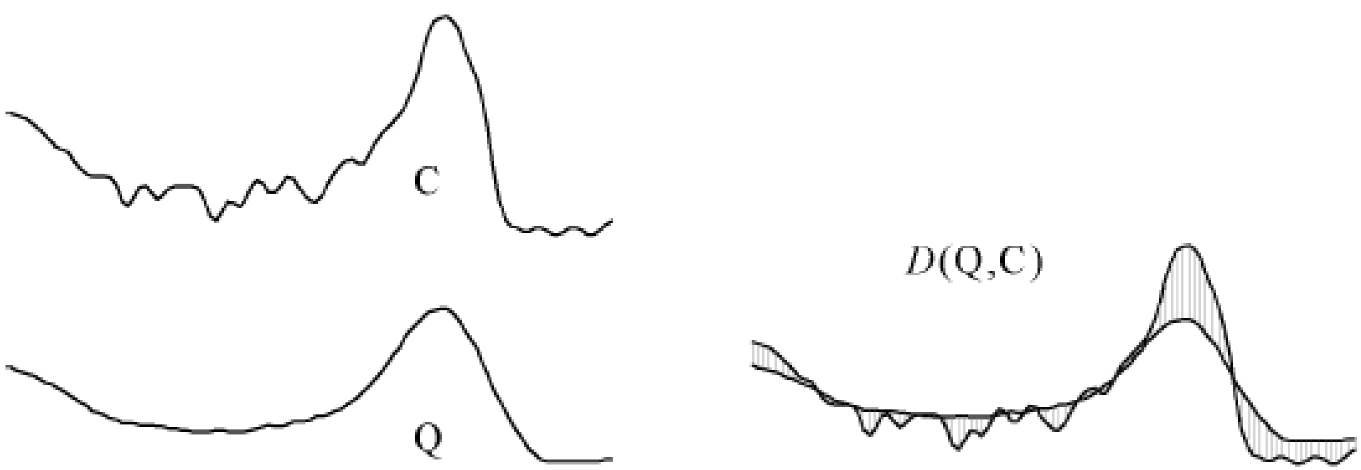

Two similar mutual funds in the different fund group

# Similarity Search for Time Series Data

- Time Series Similarity Search
  - Euclidean distances and $L_p$ norms
  - Dynamic Time Warping (DTW)
  - Time Domain vs. Frequency Domain

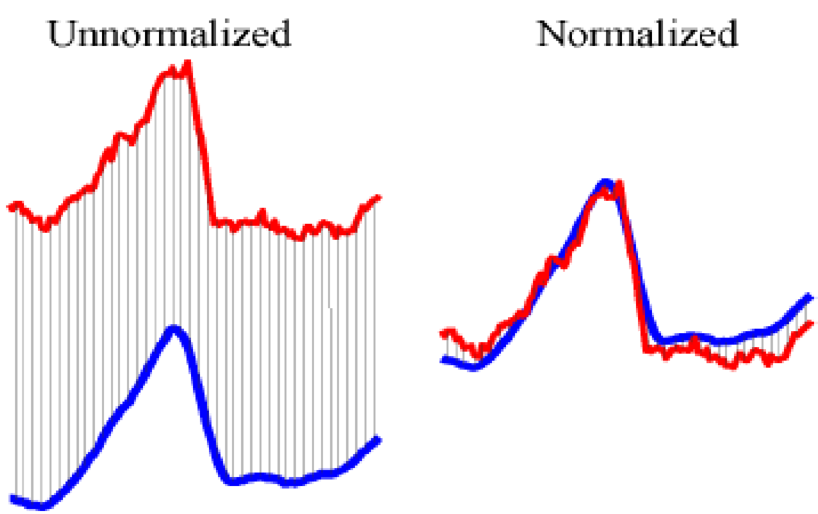
# Euclidean Distance and Lp Norms

- Given two time series with equal length n
  - $C = \{c_1, c_2, \dots, c_n\}$
  - $Q = \{q_1, q_2, \dots, q_n\}$
  - $d(C, Q) = (\sum |c_i - q_i|^p)^{1/p}$
    - When p=2, it is Euclidean distance



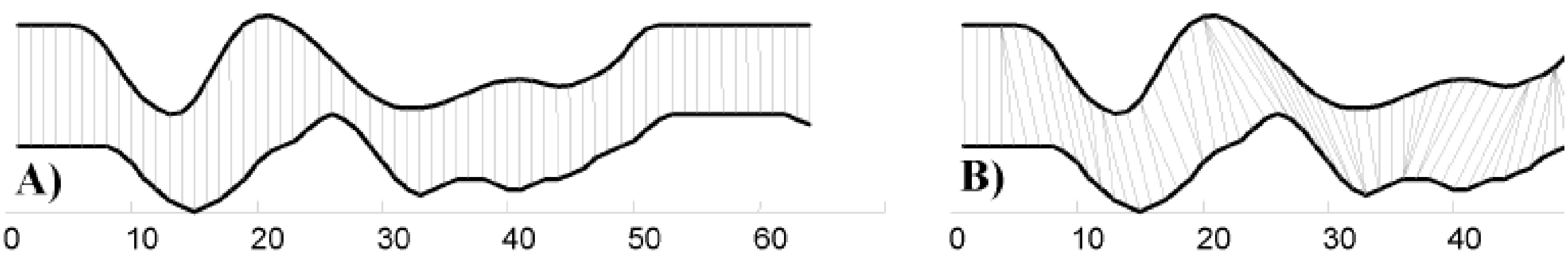
# Enhanced Lp Norm-based Distance

- Issues with Lp Norm: cannot deal with offset and scaling in the Y-axis
- Solution: normalizing the time series
  - $c_i' = \frac{c_i - \mu(C)}{\sigma(C)}$



# Dynamic Time Warping (DTW)

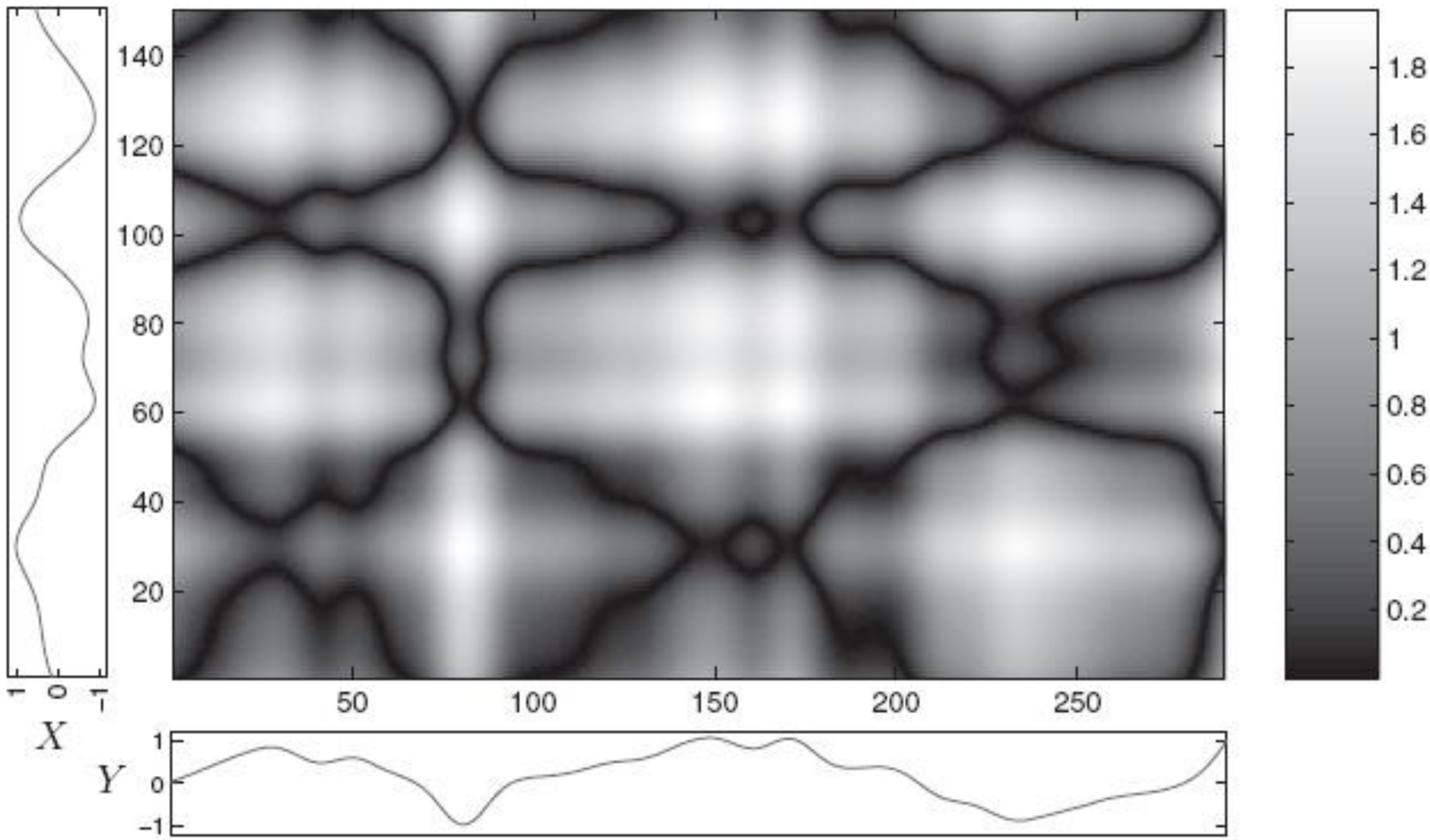
- For two sequences that do not line up well in X-axis, but share roughly similar shape
  - We need to warp the time axis to make better alignment



# Goal of DTW

- Given
  - Two sequences (with possible different lengths):
    - $X = \{x_1, x_2, \dots, x_N\}$
    - $Y = \{y_1, y_2, \dots, y_M\}$
  - A local distance (cost) measure between $x_n$ and $y_m$
- Goal:
  - Find an alignment between X and Y, such that, the overall cost is minimized

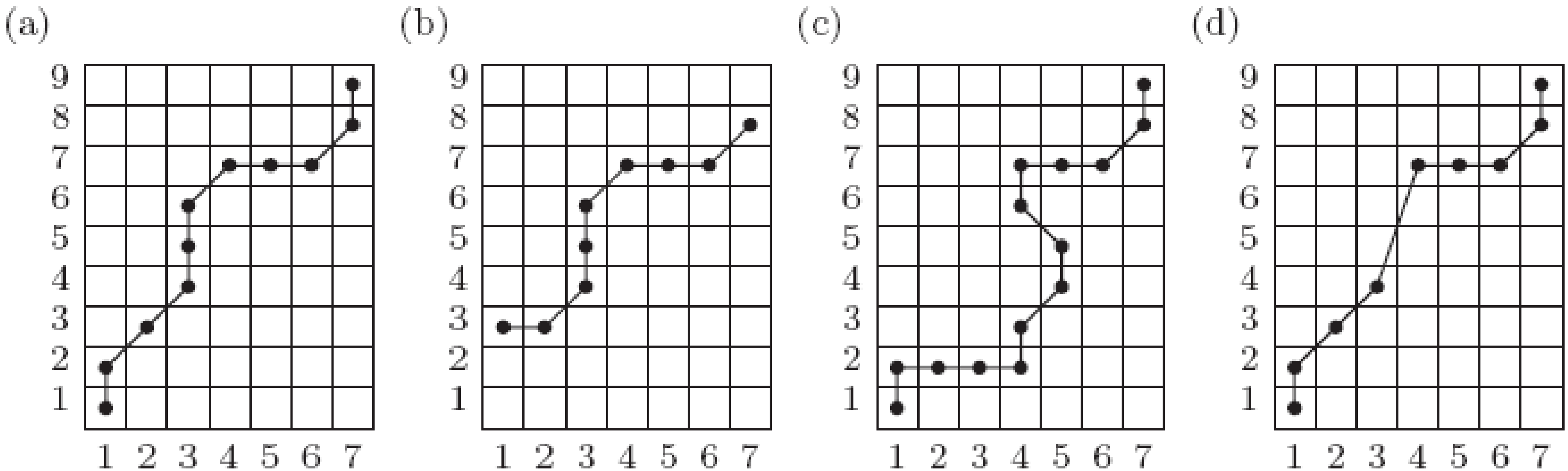
# Cost Matrix of Two Time Series



# Represent an Alignment by Warping Path

- An (N,M)-warping path is a sequence $p = (p_1, p_2, \ldots, p_L)$ with $p_l = (n_l, m_l)$, satisfying the three conditions:
  - Boundary condition: $p_1 = (1,1), p_L = (N, M)$
    - Starting from the first point and ending at last point
  - Monotonicity condition: $n_l$ and $m_l$ are non-decreasing with $l$
  - Step size condition: $p_{l+1} - p_l \in \{(0,1), (1,0), (1,1)\}$
    - Move one step right, up, or up-right

# Q: Which Path is a Warping Path?



# Optimal Warping Path

- The total cost given a warping path p
  - $c_p(X,Y) = \sum_l c(x_{n_l}, y_{m_l})$
- The optimal warping path p*
  - $c_{p^*}(X,Y) = \min\{c_p(X,Y) | p \; is \; an \; (N,M) - warping \; path\}$
- DTW distance between X and Y is defined as:
  - the optimal cost $c_{p^*}(X,Y)$

# How to Find p*?

- Naïve solution:
  - Enumerate all the possible warping path
  - Exponential in N and M!

# Dynamic Programming for DTW

- Dynamic programming:
  - Let D(n,m) denote the DTW distance between X(1,...,n) and Y(1,...,m)
  - D is called accumulative cost matrix
    - Note D(N,M) = DTW(X,Y)
  - Recursively calculate D(n,m)
    - $D(n, m) = \min\{D(n-1, m), D(n, m-1), D(n-1, m-1)\} + c(x_n, y_m)$
    - When m or n = 1
      - $D(n, 1) = \sum_{k=1:n} c(x_k, 1);$
      - $D(1, m) = \sum_{k=1:m} c(1, y_k);$

**Time complexity: O(MN)**

# Trace back to Get p* from D

**Algorithm:** OptimalWarpingPath

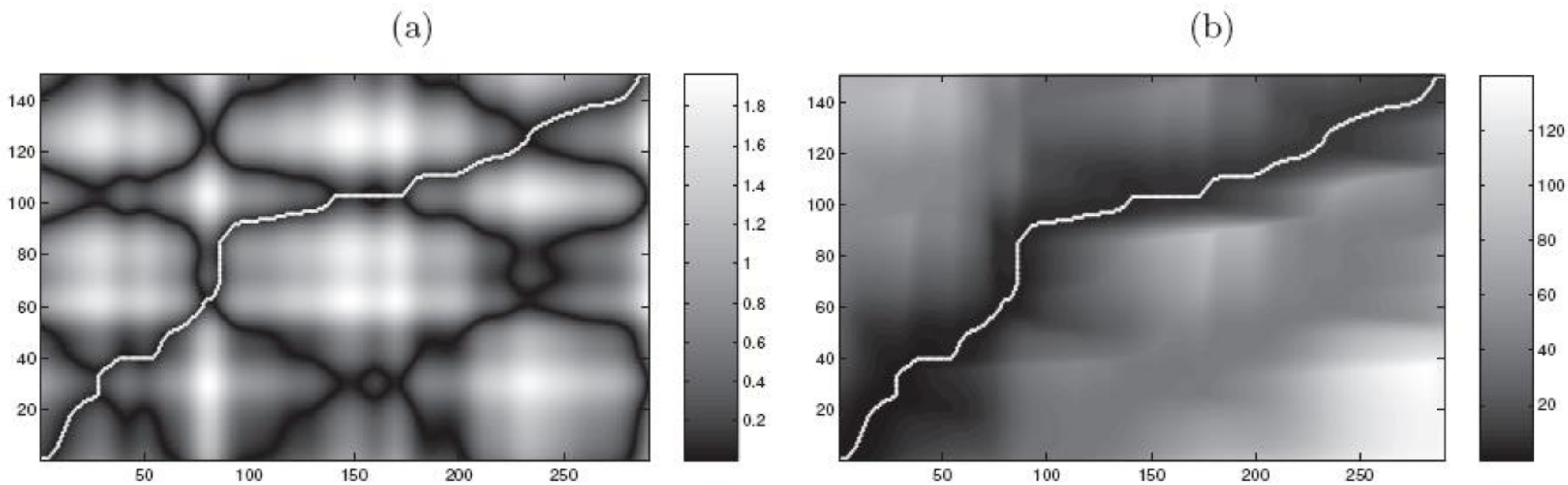
**Input:** Accumulated cost matrix $D$.
**Output:** Optimal warping path $p^*$.

**Procedure:** The optimal path $p^* = (p_1, \ldots, p_L)$ is computed in reverse order of the indices starting with $p_L = (N, M)$. Suppose $p_\ell = (n, m)$ has been computed. In case $(n, m) = (1, 1)$, one must have $\ell = 1$ and we are finished. Otherwise,

$$p_{\ell-1} := \begin{cases} (1, m-1), & \text{if } n = 1 \\ (n-1, 1), & \text{if } m = 1 \\ \operatorname{argmin}\{D(n-1, m-1), \\ \qquad D(n-1, m), D(n, m-1)\}, & \text{otherwise,} \end{cases} \tag{4.6}$$

where we take the lexicographically smallest pair in case "argmin" is not unique.

# Example

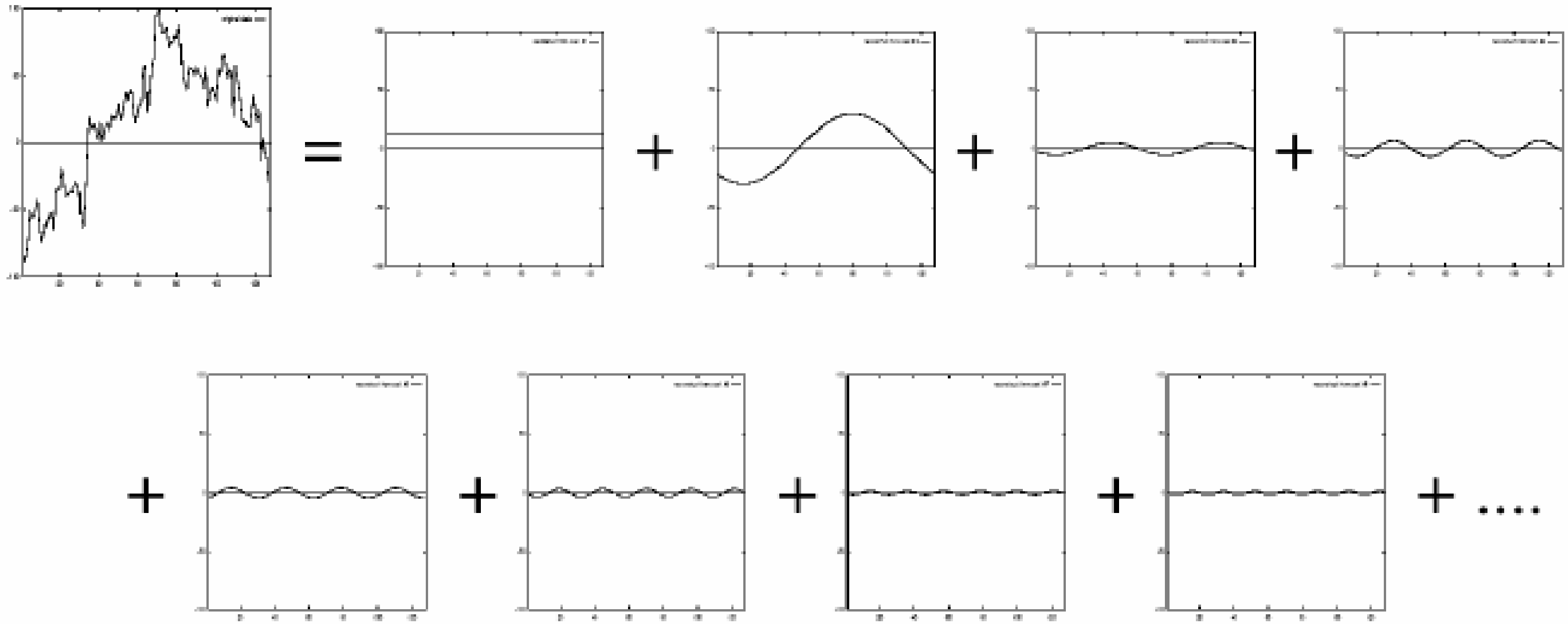


(a)
(b)
140
120
100
80
60
40
20
50
100
150
200
250
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0.2
120
100
80
60
40
20

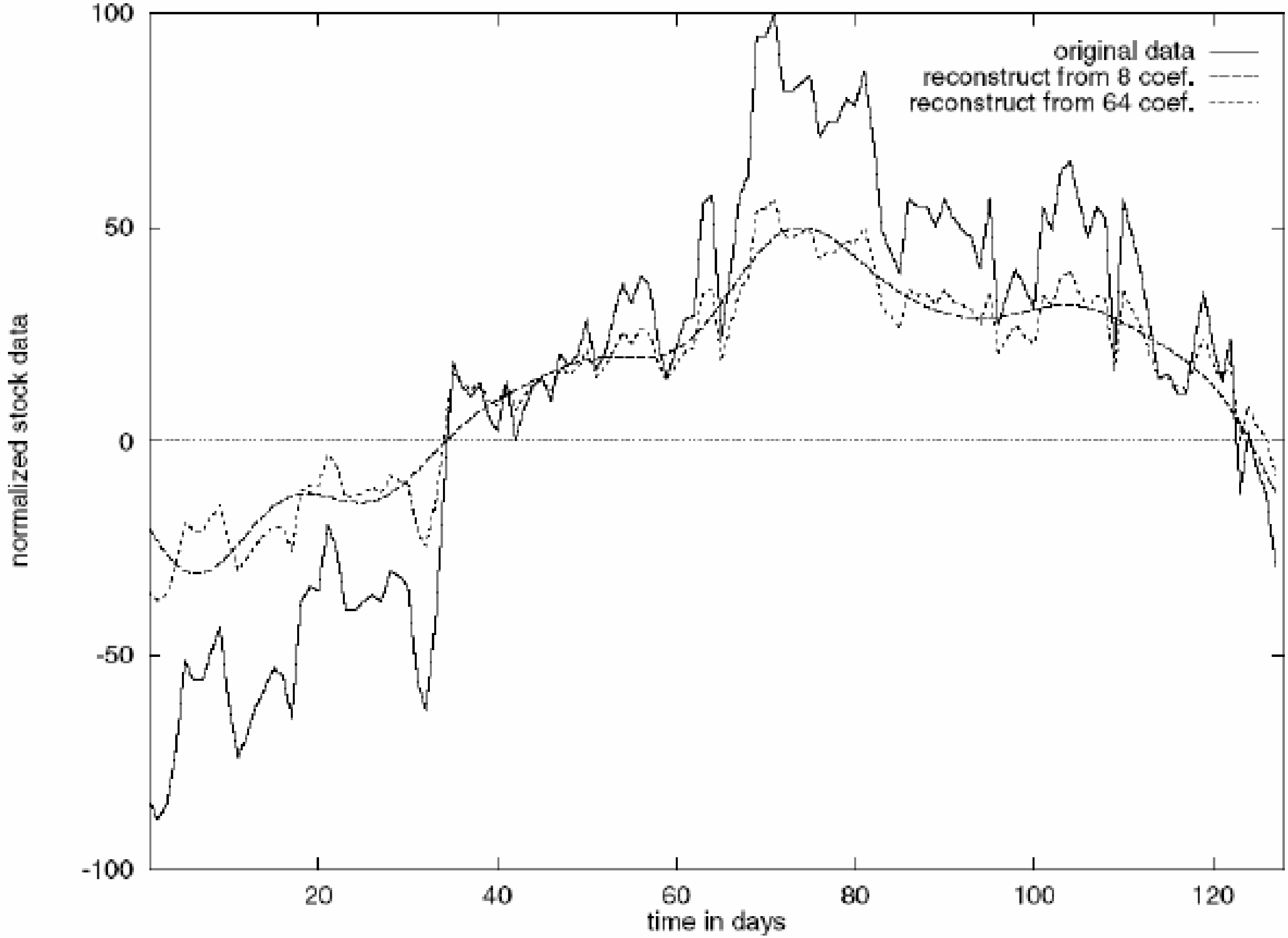
# Time Domain vs. Frequency Domain

- Many techniques for signal analysis require the data to be in the frequency domain
- Usually data-independent transformations are used
  - The transformation matrix is determined a priori
    - discrete Fourier transform (DFT)
    - discrete wavelet transform (DWT)
- The distance between two signals in the time domain is the same as their Euclidean distance in the frequency domain

# Example of DFT

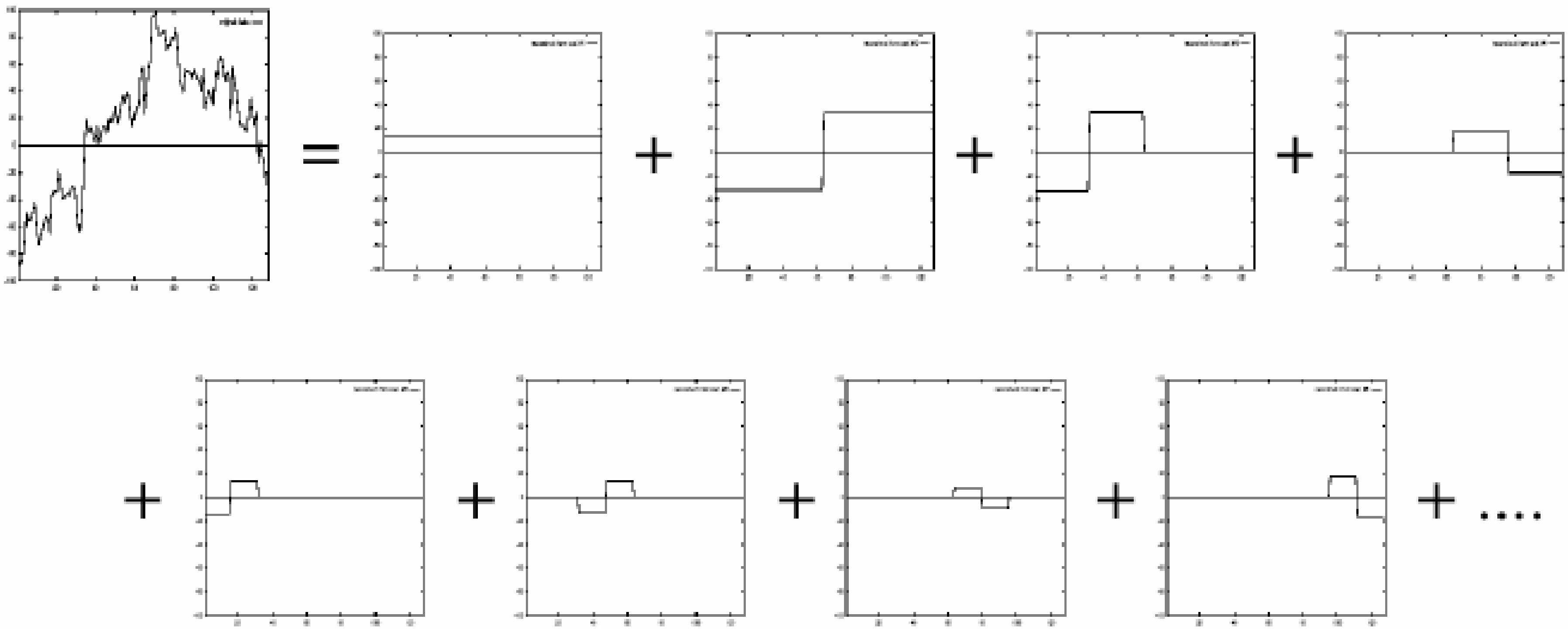


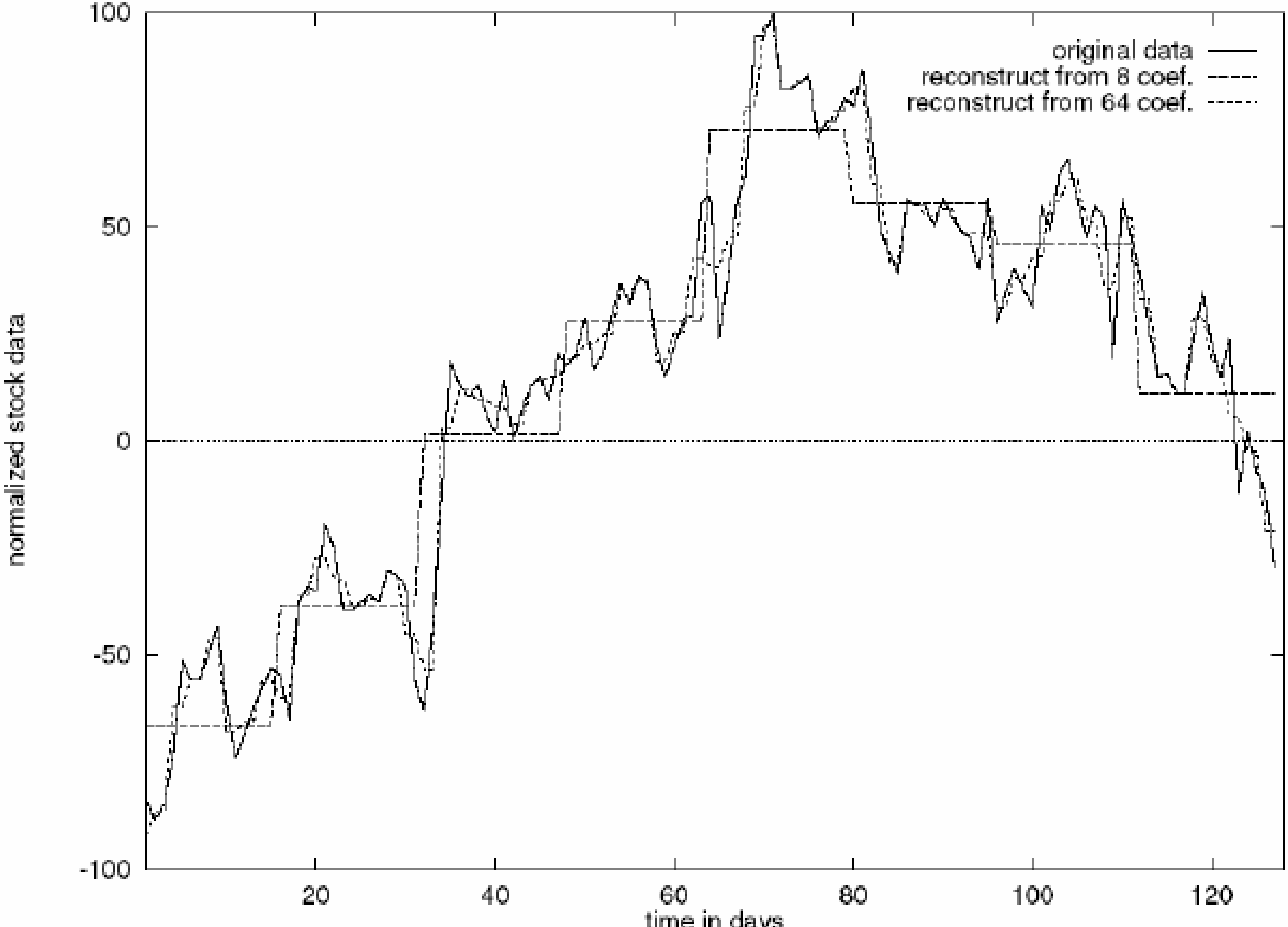
Figures taken from: “A comparison of DFT and DWT based similarity search in Time-series Databases” (Also figures on slide 9,17,18,24,25)



original data
reconstruct from 8 coef.
reconstruct from 64 coef.
normalized stock data
time in days
100
50
0
-50
-100
20
40
60
80
100
120

# Example of DWT (with Harr Wavelet)




original data
reconstruct from 8 coef.
reconstruct from 64 coef.
normalized stock data
time in days
100
50
0
-50
-100
20
40
60
80
100
120

# *Discrete Fourier Transformation

from $\vec{x} = [x_t], t = 0, \ldots, n-1$ to $\vec{X} = [X_f], f = 0, \ldots, n-1$:

$$X_f = \frac{1}{\sqrt{n}} \sum_{t=0}^{n-1} x_t \exp(-j2\pi ft/n),\ f = 0, 1, \ldots, n-1$$

- DFT does a good job of concentrating energy in the first few coefficients
- If we keep only first a few coefficients in DFT, we can compute the lower bounds of the actual distance
- Feature extraction: keep the first few coefficients (F-index) as representative of the sequence

# *DFT (Cont.)

- Parseval's Theorem

$$\sum_{t=0}^{n-1} |x_t|^2 = \sum_{f=0}^{n-1} |X_f|^2$$

  - The Euclidean distance between two signals in the time domain is the same as their distance in the frequency domain

- Keep the first few (say, 3) coefficients underestimates the distance and there will be no false dismissals!

$$\sum_{t=0}^{n} |S[t]-Q[t]|^2 \leq \varepsilon \Rightarrow \sum_{f=0}^{3} |F(S)[f]-F(Q)[f]|^2 \leq \varepsilon$$

# Mining Time Series Data

- Basic Concepts
- Time Series Prediction and Forecasting
- Time Series Similarity Search
- Summary

# Summary

- Time Series Prediction and Forecasting
  - Autocorrelation; autoregression
- Time series similarity search
  - Euclidean distance and Lp norm
  - Dynamic time warping
  - Time domain vs. frequency domain