Time Series
Mining and Forecasting

Duen Horng (Polo) Chau
Assistant Professor
Associate Director, MS Analytics
Georgia Tech

Partly based on materials by
Professors Guy Lebanon, Jeffrey Heer, John Stasko, Christos Faloutsos, Parishit Ram (GT PhD alum; SkyTree), Alex Gray
Outline

• Motivation
• Similarity search – distance functions
• Linear Forecasting
• Non-linear forecasting
• Conclusions
Problem definition

• **Given**: one or more sequences

\[ x_1, x_2, \ldots, x_t, \ldots \]

\[ (y_1, y_2, \ldots, y_t, \ldots) \]

\[ (\ldots) \]

• **Find**

  – similar sequences; forecasts
  – patterns; clusters; outliers
Motivation - Applications

• Financial, sales, economic series

• Medical
  – ECGs +; blood pressure etc monitoring
  – reactions to new drugs
  – elderly care
Motivation - Applications (cont’d)

• ‘Smart house’
  – sensors monitor temperature, humidity, air quality

• video surveillance
Motivation - Applications (cont’d)

• Weather, environment/anti-pollution
  – volcano monitoring
  – air/water pollutant monitoring
Motivation - Applications (cont’d)

• Computer systems
  – ‘Active Disks’ (buffering, prefetching)
  – web servers (ditto)
  – network traffic monitoring
  – ...

Stream Data: Disk accesses

- **Disk traffic**
  - #bytes
  - time
Problem #1:

Goal: given a signal (e.g., #packets over time)
Find: patterns, periodicities, and/or compress lynx caught per year (packets per day; temperature per day)
Problem#2: Forecast

Given $x_t, x_{t-1}, \ldots$, forecast $x_{t+1}$
Problem #2': Similarity search

E.g., Find a 3-tick pattern, similar to the last one
Problem #3:

- Given: A set of correlated time sequences
- Forecast ‘Sent(t)’
Important observations

Patterns, rules, forecasting and similarity indexing are closely related:

• To do forecasting, we need
  – to find patterns/rules
  – to find similar settings in the past

• to find outliers, we need to have forecasts
  – (outlier = too far away from our forecast)
Outline

• Motivation

• Similarity search and distance functions
  – Euclidean
  – Time-warping

• ...
Importance of distance functions

Subtle, but absolutely necessary:

• A ‘must’ for similarity indexing (-> forecasting)
• A ‘must’ for clustering

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

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

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

\( L_1: \text{city-block} = \text{Manhattan} \)

\( L_2 = \text{Euclidean} \)

\( L_\infty \)
Observation #1

Time sequence -> n-d vector
Observation #2

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

• allow accelerations - decelerations
  – (with or without penalty)
• THEN compute the (Euclidean) distance (+ penalty)
• related to the string-editing distance
Time Warping

‘stutters’:
Time warping

Q: how to compute it?
A: dynamic programming

\[ D(i, j) = \text{cost to match} \]

gives

prefix of length \( i \) of first sequence \( x \) with prefix of length \( j \) of second sequence \( y \)
Time warping

Thus, with no penalty for stutter, for sequences

\[ x_1, x_2, \ldots, x_i,; \quad y_1, y_2, \ldots, y_j \]

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

Time warping

VERY SIMILAR to the string-editing distance

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

• Complexity: $O(M*N)$ - quadratic on the length of the strings

• Many variations (penalty for stutters; limit on the number/percentage of stutters; …)

• popular in voice processing

[Rabiner + Juang]
Other Distance functions

• piece-wise linear/flat approx.; compare pieces [Keogh+01] [Faloutsos+97]
• ‘cepstrum’ (for voice [Rabiner+Juang])
  – do DFT; take log of amplitude; do DFT again!
• Allow for small gaps [Agrawal+95]
See tutorial by [Gunopulos + Das, SIGMOD01]
Other Distance functions

• In [Keogh+, KDD’04]: parameter-free, MDL based
Conclusions

Prevailing distances:
  – Euclidean and
  – time-warping
Outline

• Motivation
• Similarity search and distance functions
• Linear Forecasting
• Non-linear forecasting
• Conclusions
Linear Forecasting
Outline

• Motivation
• ...
• Linear Forecasting
  – Auto-regression: Least Squares; RLS
  – Co-evolving time sequences
  – Examples
  – Conclusions
Problem#2: Forecast

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

MANUALLY:

- remove trends
- spot periodicities

7 days
Problem#2: Forecast

- Solution: try to express $x_t$ as a linear function of the past: $x_{t-1}, x_{t-2}, \ldots$, (up to a window of $w$)

Formally:

$$x_t \approx a_1 x_{t-1} + \ldots + a_w x_{t-w} + \text{noise}$$
(Problem: Back-cast; interpolate)

- Solution - interpolate: try to express \( x_t \) as a linear function of the past AND the future:
  \[ x_{t+1}, x_{t+2}, \ldots x_{t+w_{future}}; x_{t-1}, \ldots x_{t-w_{past}} \]
  (up to windows of \( w_{past}, w_{future} \))

- EXACTLY the same algo's
Refresher: Linear Regression

Express what we don’t know (= “dependent variable”) as a linear function of what we know (= “independent variable(s)”)
Refresher: Linear Regression

<table>
<thead>
<tr>
<th>patient</th>
<th>weight</th>
<th>height</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>27</td>
<td>43</td>
</tr>
<tr>
<td>2</td>
<td>43</td>
<td>54</td>
</tr>
<tr>
<td>3</td>
<td>54</td>
<td>72</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>25</td>
<td>??</td>
</tr>
</tbody>
</table>

Express what we don’t know (= “dependent variable”) as a linear function of what we know (= “independent variable(s)”)
Refresher: Linear Regression

<table>
<thead>
<tr>
<th>patient</th>
<th>weight</th>
<th>height</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>27</td>
<td>43</td>
</tr>
<tr>
<td>2</td>
<td>43</td>
<td>54</td>
</tr>
<tr>
<td>3</td>
<td>54</td>
<td>72</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>25</td>
<td>??</td>
</tr>
</tbody>
</table>

Express what we **don’t know** (= “dependent variable”) as a linear function of what we **know** (= “independent variable(s)”)
Refresher: Linear Regression

<table>
<thead>
<tr>
<th>patient</th>
<th>weight</th>
<th>height</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>27</td>
<td>43</td>
</tr>
<tr>
<td>2</td>
<td>43</td>
<td>54</td>
</tr>
<tr>
<td>3</td>
<td>54</td>
<td>72</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>25</td>
<td>??</td>
</tr>
</tbody>
</table>

Express what we don’t know (= “dependent variable”) as a linear function of what we know (= “independent variable(s)”)
Linear **Auto** Regression

<table>
<thead>
<tr>
<th>Time</th>
<th>Packets Sent(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>43</td>
</tr>
<tr>
<td>2</td>
<td>54</td>
</tr>
<tr>
<td>3</td>
<td>72</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>??</td>
</tr>
</tbody>
</table>
### Linear Auto Regression

#### Lag $w = 1$

**Dependent variable** = # of packets sent ($S[t]$)

**Independent variable** = # of packets sent ($S[t-1]$)

<table>
<thead>
<tr>
<th>Time</th>
<th>Packets Sent ($t-1$)</th>
<th>Packets Sent ($t$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>43</td>
</tr>
<tr>
<td>2</td>
<td>43</td>
<td>54</td>
</tr>
<tr>
<td>3</td>
<td>54</td>
<td>72</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>25</td>
<td>??</td>
</tr>
</tbody>
</table>

**‘lag-plot’**

- The $x$-axis represents the number of packets sent at time $t-1$.
- The $y$-axis represents the number of packets sent at time $t$.

**Lag-plot**

- The pattern of the lag plot suggests a linear relationship between the number of packets sent at time $t-1$ and the number of packets sent at time $t$.

---

**Example:**

- At time $t = 1$, there were 43 packets sent.
- At time $t = 2$, there were 43 packets sent ($S[1]$) and 54 packets sent ($S[2]$).
- At time $t = 3$, there were 54 packets sent ($S[2]$) and 72 packets sent ($S[3]$).

---

**Note:**

- The lag plot helps visualize the relationship between past and current data points.
- Linear Auto Regression models the relationship between the current value and one or more past values.
## Linear Auto Regression

### Lag $w = 1$

**Dependent variable** = # of packets sent ($S[t]$)

**Independent variable** = # of packets sent ($S[t-1]$)

<table>
<thead>
<tr>
<th>Time</th>
<th>Packets Sent (t-1)</th>
<th>Packets Sent (t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>43</td>
</tr>
<tr>
<td>2</td>
<td>43</td>
<td>54</td>
</tr>
<tr>
<td>3</td>
<td>54</td>
<td>72</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>25</td>
<td>??</td>
</tr>
</tbody>
</table>

#packets sent at time $t$

#packets sent at time $t-1$

‘lag-plot’
Linear **Auto** Regression

<table>
<thead>
<tr>
<th>Time</th>
<th>Packets Sent (t-1)</th>
<th>Packets Sent(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>43</td>
</tr>
<tr>
<td>2</td>
<td>43</td>
<td>54</td>
</tr>
<tr>
<td>3</td>
<td>54</td>
<td>72</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>25</td>
<td>??</td>
</tr>
</tbody>
</table>

Lag \( w = 1 \)

Dependent variable = \# of packets sent \((S[t])\)

Independent variable = \# of packets sent \((S[t-1])\)
Linear **Auto** Regression

<table>
<thead>
<tr>
<th>Time</th>
<th>Packets Sent (t-1)</th>
<th>Packets Sent(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>43</td>
</tr>
<tr>
<td>2</td>
<td>43</td>
<td>54</td>
</tr>
<tr>
<td>3</td>
<td>54</td>
<td>72</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>N</td>
<td>25</td>
<td>??</td>
</tr>
</tbody>
</table>

**Lag \ w = 1**

**Dependent variable** = # of packets sent (S [t])

**Independent variable** = # of packets sent (S[t-1])
More details:

• Q1: Can it work with window \( w > 1 \)?
• A1: YES!
More details:

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

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

• Q1: Can it work with window \( w > 1 \)?
• A1: YES! The problem becomes:

\[
X_{[N \times w]} \times a_{[w \times 1]} = y_{[N \times 1]}
\]

• OVER-CONSTRAINED
  – \( a \) is the vector of the regression coefficients
  – \( X \) has the \( N \) values of the \( w \) indep. variables
  – \( y \) has the \( N \) values of the dependent variable
More details:

- \( X_{[N \times w]} \times a_{[w \times 1]} = y_{[N \times 1]} \)

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

- \( X_{[N \times w]} \times a_{[w \times 1]} = y_{[N \times 1]} \)

\( \begin{align*} 
\text{Ind-var1} & \quad \text{Ind-var-w} \\
\text{time} & \\
& \downarrow \\
\begin{bmatrix} 
X_{11}, X_{12}, \ldots, X_{1w} \\
X_{21}, X_{22}, \ldots, X_{2w} \\
\vdots \\
X_{N1}, X_{N2}, \ldots, X_{nw} 
\end{bmatrix} \times 
\begin{bmatrix} 
a_1 \\
a_2 \\
\vdots \\
a_w 
\end{bmatrix} = 
\begin{bmatrix} 
y_1 \\
y_2 \\
\vdots \\
y_N 
\end{bmatrix} 
\end{align*} \)
More details

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

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

• (Moore-Penrose pseudo-inverse)
• $a$ is the vector that minimizes the RMSE from $y$
More details

• Straightforward solution:

\[ a = (X^T \times X)^{-1} \times (X^T \times y) \]

- \( a \) : Regression Coeff. Vector
- \( X \) : Sample Matrix

• Observations:
  - Sample matrix \( X \) grows over time
  - needs matrix inversion
  - \( O(N \times w^2) \) computation
  - \( O(N \times w) \) storage
Even more details

• Q3: Can we estimate a incrementally?
• A3: Yes, with the brilliant, classic method of “Recursive Least Squares” (RLS) (see, e.g., [Yi+00], for details).
• We can do the matrix inversion, WITHOUT inversion! (How is that possible?!)
Even more details

• Q3: Can we estimate a incrementally?
• A3: Yes, with the brilliant, classic method of “Recursive Least Squares” (RLS) (see, e.g., [Yi+00], for details).
• We can do the matrix inversion, WITHOUT inversion! (How is that possible?!) 
• A: our matrix has special form: \((X^T X)\)
More details

At the $N+1$ time tick:

$X_{N+1} \quad X_N$
More details: key ideas

- Let $G_N = (X_N^T \times X_N)^{-1}$ ("gain matrix")
- $G_{N+1}$ can be computed recursively from $G_N$ without matrix inversion
Comparison:

• **Straightforward Least Squares**
  – Needs huge matrix (growing in size)
    \( O(N \times w) \)
  – Costly matrix operation
    \( O(N \times w^2) \)

• **Recursive LS**
  – Need much smaller, fixed size matrix
    \( O(w \times w) \)
  – Fast, incremental computation
    \( O(1 \times w^2) \)
  – no matrix inversion

\( N = 10^6, \quad w = 1-100 \)
EVEN more details:

\[ G_{N+1} = G_N - [c]^{-1} \times [G_N \times x_{N+1}^T] \times x_{N+1} \times G_N \]

\[ c = [1 + x_{N+1} \times G_N \times x_{N+1}^T] \]

Let’s elaborate
(Very important, very valuable!)
EVEN more details:

\[ a = \left[ X_{N+1}^T \times X_{N+1} \right]^{-1} \times \left[ X_{N+1}^T \times y_{N+1} \right] \]
EVEN more details:

\[
\alpha = \left[ X_{N+1}^T \times X_{N+1} \right]^{-1} \times \left[ X_{N+1}^T \times y_{N+1} \right]
\]

\[
\begin{align*}
[w \times 1] & \quad [w \times (N+1)] & \quad [(N+1) \times w] & \quad [w \times (N+1)] & \quad [(N+1) \times 1] 
\end{align*}
\]
EVEN more details:

\[ a = \left[ X_{N+1}^T \times X_{N+1} \right]^{-1} \times \left[ X_{N+1}^T \times y_{N+1} \right] \]

[w x (N+1)] \quad [(N+1) x w]
EVEN more details:

\[ a = \left[ X_{N+1}^T \times X_{N+1} \right]^{-1} \times \left[ X_{N+1}^T \times y_{N+1} \right] \]

\( G_{N+1} \equiv \left[ X_{N+1}^T \times X_{N+1} \right]^{-1} \)

\[ G_{N+1} = G_N - [c]^{-1} \times \left[ G_N \times x_{N+1}^T \right] \times x_{N+1} \times G_N \]

SCALAR!

\[ c = \left[ 1 + x_{N+1} \times G_N \times x_{N+1}^T \right] \]
Altogether:

\[ G_0 \equiv \delta I \]

where
- \( I \): \( w \times w \) identity matrix
- \( \delta \): a large positive number
Comparison:

- **Straightforward Least Squares**
  - Needs huge matrix (growing in size) \( O(N \times w) \)
  - Costly matrix operation \( O(N \times w^2) \)

- **Recursive LS**
  - Need much smaller, fixed size matrix \( O(w \times w) \)
  - Fast, incremental computation \( O(1 \times w^2) \)
  - no matrix inversion

\[ N = 10^6, \quad w = 1-100 \]
Pictorially:

• Given:

  Independent Variable

  Dependent Variable
Pictorially:

RLS: quickly compute new best fit

new point
Even more details

• Q4: can we ‘forget’ the older samples?
• A4: Yes - RLS can easily handle that [Yi+00]:
Adaptability - ‘forgetting’

Independent Variable
e.g., #packets sent

Dependent Variable
e.g., #bytes sent
Adaptability - ‘forgetting’

Trend change

(R)LS with no forgetting

Independent Variable
eg. #packets sent

Dependent Variable
eg., #bytes sent
Adaptability - ‘forgetting’

- RLS: can *trivially* handle ‘forgetting’