Data Mining II
Time Series Analysis

Heiko Paulheim
Introduction

• So far, we have only looked at data without a time dimension
  – or simply ignored the temporal aspect

• Many “classic” DM problems have variants that respect time
  – frequent pattern mining → sequential pattern mining
  – classification → predicting sequences of nominals
  – regression → predicting the continuation of a numeric series
Contents

• Sequential Pattern Mining
  – Finding frequent subsequences in set of sequences
  – the GSP algorithm

• Trend analysis
  – Is a time series moving up or down?
  – Simple models and smoothing
  – Identifying seasonal effects

• Forecasting
  – Predicting future developments from the past
  – Autoregressive models and windowing
  – Exponential smoothing and its extensions
Sequential Pattern Mining: Application 1

• Web usage mining (navigation analysis)
• Input
  – Server logs
• Patterns
  – typical sequences of pages
• Usage
  – restructuring web sites
Sequential Pattern Mining: Application 2

- Recurring customers
  - Typical book store example:
    - (Twilight) (New Moon) → (Eclipse)

- Recommendation in online stores
- Allows more fine grained suggestions than frequent pattern mining
- Example:
  - mobile phone → charger vs. charger → mobile phone
    - are indistinguishable by frequent pattern mining
  - customers will select a charger after a mobile phone
    - but not the other way around!
    - however, Amazon does not respect sequences...
Sequential Pattern Mining: Application 3

- Using texts as a corpus
  - looking for common sequences of words
  - allows for intelligent suggestions for autocompletion
Sequential Pattern Mining: Application 4

- Chord progressions in music
  - supporting musicians (or even computers) in jam sessions
  - supporting producers in writing top 10 hits :-)

Chords following em

- C: < 1%
- dm: 4%
- F: 59%
- G: < 1%
- am: 34%
- other: 2%

http://www.hooktheory.com/blog/i-analyzed-the-chords-of-1300-popular-songs-for-patterns-this-is-what-i-found/
**Sequence Data**

- Data Model: transactions containing items

<table>
<thead>
<tr>
<th>Sequence Database</th>
<th>Sequence</th>
<th>Element (Transaction)</th>
<th>Event (Item)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Customer Data</td>
<td>Purchase history of a given customer</td>
<td>A set of items bought by a customer at time t</td>
<td>Books, dairy products, CDs, etc</td>
</tr>
<tr>
<td>Web Server Logs</td>
<td>Browsing activity of a particular Web visitor</td>
<td>A collection of files viewed by a Web visitor after a single mouse click</td>
<td>Home page, index page, contact info, etc</td>
</tr>
<tr>
<td>Chord Progressions</td>
<td>Chords played in a song</td>
<td>Individual notes hit at a time</td>
<td>Notes (C, C#, D, ...)</td>
</tr>
</tbody>
</table>

**Sequence**

- E1
- E2
- E3
- E4

**Event**

- E1
- E2
- E3
- E4

**Element**

- (Transaction)
## Sequence Database

<table>
<thead>
<tr>
<th>Object</th>
<th>Timestamp</th>
<th>Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td>2, 3, 5</td>
</tr>
<tr>
<td>A</td>
<td>20</td>
<td>6, 1</td>
</tr>
<tr>
<td>A</td>
<td>23</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>11</td>
<td>4, 5, 6</td>
</tr>
<tr>
<td>B</td>
<td>17</td>
<td>2</td>
</tr>
<tr>
<td>B</td>
<td>21</td>
<td>7, 8, 1, 2</td>
</tr>
<tr>
<td>B</td>
<td>28</td>
<td>1, 6</td>
</tr>
<tr>
<td>C</td>
<td>14</td>
<td>1, 8, 7</td>
</tr>
</tbody>
</table>
Formal Definition of a Sequence

- A **sequence** is an ordered list of elements (transactions)

\[ s = < e_1 e_2 e_3 \ldots > \]

- Each element contains a collection of items (events)

\[ e_i = \{i_1, i_2, \ldots, i_k\} \]

- Each element is attributed to a specific time

- **Length of a sequence** $|s|$ is given by the number of elements of the sequence.

- A **k-sequence** is a sequence that contains k **events** (items).
Further Examples of Sequences

• Web browsing sequence:

  < {Homepage} {Electronics} {Digital Cameras} {Canon EOS} {Shopping Cart} {Order Confirmation} {Homepage} >

• Sequence of books checked out at a library:

  < {Fellowship of the Ring} {The Two Towers, Return of the King} >

• Sequence of initiating events causing the nuclear accident at 3-mile Island:

  < {clogged resin} {outlet valve closure} {loss of feedwater} {condenser polisher outlet valve shut} {booster pumps stop} {main waterpump stops, main turbine stops} {reactor pressure increases} >
Formal Definition of a Subsequence

- A sequence \(<a_1 a_2 \ldots a_n>\) is contained in another sequence \(<b_1 b_2 \ldots b_m>\) (\(m \geq n\)) if there exist integers 
  \(i_1 < i_2 < \ldots < i_n\) such that \(a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, \ldots, a_n \subseteq b_{i_n}\)

<table>
<thead>
<tr>
<th>Data sequence (&lt;b&gt;)</th>
<th>Subsequence (&lt;a&gt;)</th>
<th>Contain?</th>
</tr>
</thead>
<tbody>
<tr>
<td>(&lt;{2,4} {3,5,6} {8}&gt;)</td>
<td>(&lt;{2} {3,5}&gt;)</td>
<td>Yes</td>
</tr>
<tr>
<td>(&lt;{1,2} {3,4}&gt;)</td>
<td>(&lt;{1} {2}&gt;)</td>
<td>No</td>
</tr>
<tr>
<td>(&lt;{2,4} {2,4} {2,5}&gt;)</td>
<td>(&lt;{2} {4}&gt;)</td>
<td>Yes</td>
</tr>
</tbody>
</table>

- The **support** of a subsequence \(w\) is defined as the fraction of data sequences that contain \(w\)

- A **sequential pattern** is a frequent subsequence (i.e., a subsequence whose support is \(\geq \text{minsup}\))
Table 1. A set of transactions sorted by customer ID and transaction time

<table>
<thead>
<tr>
<th>Customer ID</th>
<th>Transaction Time</th>
<th>Transaction (items bought)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>July 20, 2005</td>
<td>30</td>
</tr>
<tr>
<td>1</td>
<td>July 25, 2005</td>
<td>90</td>
</tr>
<tr>
<td>2</td>
<td>July 9, 2005</td>
<td>10, 20</td>
</tr>
<tr>
<td>2</td>
<td>July 14, 2005</td>
<td>30</td>
</tr>
<tr>
<td>2</td>
<td>July 20, 2005</td>
<td>40, 60, 70</td>
</tr>
<tr>
<td>3</td>
<td>July 25, 2005</td>
<td>30, 50, 70</td>
</tr>
<tr>
<td>4</td>
<td>July 25, 2005</td>
<td>30</td>
</tr>
<tr>
<td>4</td>
<td>July 29, 2005</td>
<td>40, 70</td>
</tr>
<tr>
<td>4</td>
<td>August 2, 2005</td>
<td>90</td>
</tr>
<tr>
<td>5</td>
<td>July 12, 2005</td>
<td>90</td>
</tr>
</tbody>
</table>
Examples of Sequential Patterns

Table 2. Data sequences produced from the transaction database in Table 1.

<table>
<thead>
<tr>
<th>Customer ID</th>
<th>Data Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>⟨{30} {90}⟩</td>
</tr>
<tr>
<td>2</td>
<td>⟨{10, 20} {30} {40, 60, 70}⟩</td>
</tr>
<tr>
<td>3</td>
<td>⟨{30, 50, 70}⟩</td>
</tr>
<tr>
<td>4</td>
<td>⟨{30} {40, 70} {90}⟩</td>
</tr>
<tr>
<td>5</td>
<td>⟨{90}⟩</td>
</tr>
</tbody>
</table>

Table 3. The final output sequential patterns

<table>
<thead>
<tr>
<th></th>
<th>Sequential Patterns with Support ≥ 25%</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-sequences</td>
<td>⟨{30}⟩, ⟨{40}⟩, ⟨{70}⟩, ⟨{90}⟩</td>
</tr>
<tr>
<td>2-sequences</td>
<td>⟨{30} {40}⟩, ⟨{30} {70}⟩, ⟨{30} {90}⟩, ⟨{40, 70}⟩</td>
</tr>
<tr>
<td>3-sequences</td>
<td>⟨{30} {40, 70}⟩</td>
</tr>
</tbody>
</table>
Sequential Pattern Mining

• Given:
  – a database of sequences
  – a user-specified minimum support threshold, $\text{minsup}$

• Task:
  – Find all subsequences with support $\geq \text{minsup}$

• Challenge:
  – Very large number of candidate subsequences that need to be checked against the sequence database
  – By applying the Apriori principle, the number of candidates can be pruned significantly
Determining the Candidate Subsequences

- Given \( n \) events: \( i_1, i_2, i_3, \ldots, i_n \)

- Candidate 1-subsequences:
  \(<\{i_1\}>, <\{i_2\}>, <\{i_3\}>, \ldots, <\{i_n\}>\)

- Candidate 2-subsequences:
  \(<\{i_1, i_2\}>, <\{i_1, i_3\}>, \ldots, <\{i_{n-1}, i_n\}>, <\{i_1\} \{i_2\}>, \ldots, <\{i_{n-1}\} \{i_n\}>, <\{i_n\} \{i_n\}>, <\{i_2, i_1\}>, <\{i_3, i_1\}>, \ldots, <\{i_n, i_{n-1}\}>\)

- Candidate 3-subsequences:
  \(<\{i_1, i_2, i_3\}>, <\{i_1, i_2, i_4\}>, \ldots, <\{i_1, i_2\} \{i_1\}>, <\{i_1, i_2\} \{i_2\}>, \ldots, <\{i_1\} \{i_1, i_2\}>, <\{i_1\} \{i_1, i_3\}>, \ldots, <\{i_1\} \{i_1\} \{i_1\}>, <\{i_1\} \{i_1\} \{i_2\}>, \ldots\)
Generalized Sequential Pattern Algorithm (GSP)

- **Step 1:**
  - Make the first pass over the sequence database D to yield all the 1-element frequent subsequences

- **Step 2:** Repeat until no new frequent subsequences are found
  1. **Candidate Generation:**
     - Merge pairs of frequent subsequences found in the (k-1)th pass to generate candidate sequences that contain k items
  2. **Candidate Pruning:**
     - Prune candidate k-sequences that contain infrequent (k-1)-subsequences (Apriori principle)
  3. **Support Counting:**
     - Make a new pass over the sequence database D to find the support for these candidate sequences
  4. **Candidate Elimination:**
     - Eliminate candidate k-sequences whose actual support is less than \( \text{minsup} \)
GSP Example

- Only one 4-sequence survives the candidate pruning step
- All other 4-sequences are removed because they contain subsequences that are not part of the set of frequent 3-sequences
Trend Detection

• Task
  – given a time series
  – find out what the general trend is (e.g., rising or falling)

• Possible obstacles
  – random effects: ice cream sales have been low this week due to rain
    • but what does that tell about next week?
  – seasonal effects: sales have risen in December
    • but what does that tell about January?
  – cyclical effects: less people attend a lecture towards the end of the semester
    • but what does that tell about the next semester?
Trend Detection

- Example: Data Analysis at Facebook

Estimation of Trend Curves

- The freehand method
  - Fit the curve by looking at the graph
  - Costly and barely reliable for large-scale data mining

- The least-squares method
  - Find the curve minimizing the sum of the squares of the deviation of points on the curve from the corresponding data points
  - cf. linear regression

- The moving-average method
  - The time series exhibit a downward trend pattern.
  - Predicted value
Example: Average Global Temperature

http://www.bbc.co.uk/schools/gcsebitesize/science/aqa_pre_2011/rocks/fuelsrev6.shtml
Example: German DAX 2013
Linear Trend

• Given a time series that has timestamps and values, i.e.,
  – \((t_i, v_i)\), where \(t_i\) is a time stamp, and \(v_i\) is a value at that time stamp
• A linear trend is a linear function
  – \(m \cdot t_i + b\)
• We can find via linear regression, e.g., using the least squares fit
Example: German DAX 2013
A Component Model of Time Series

A time series can consist of four components:

- Long-term trend ($T_t$)
- Cyclical effect ($C_t$)
- Seasonal effect ($S_t$)
- Random variation ($R_t$)

Additive Model:
- Series = $T_t + C_t + S_t + R_t$

Multiplicative Model:
- Series = $T_t \times C_t \times S_t \times R_t$

this is what we want to find

we need to eliminate those
Seasonal and Cyclical Effects

• Seasonal effects occur regularly each year
  – quarters
  – months
  – ...

• Cyclical effects occur regularly over other intervals
  – every N years
  – in the beginning/end of the month
  – on certain weekdays or on weekends
  – at certain times of the day
  – ...

Identifying Seasonal and Cyclical Effects

- There are methods of identifying and isolating those effects
  - given that the periodicity is known

- Python: statsmodels package

```python
from pandas import Series
from matplotlib import pyplot
from statsmodels.tsa.seasonal
    import seasonal_decompose
series = Series.from_csv
    ('data.csv', header=0)
result = seasonal_decompose
    (series, model='multiplicative')
result.plot()
pyplot.show()
```
Identifying Seasonal and Cyclical Effects

• Variation may occur within a year or another period
• To measure the seasonal effects we compute *seasonal indexes*
• Seasonal index
  - degree of variation of seasons in relation to global average

http://davidsills.blogspot.de/2011/10/seasons.html
Identifying Seasonal and Cyclical Effects

• Algorithm
  – Compute the trend \( \hat{y}_t \) (i.e., linear regression)
  – For each time period
    • compute the ratio \( y_t / \hat{y}_t \)
  – For each season (or other relevant period)
    • compute the average of \( y_t / \hat{y}_t \)
    • this gives us the average deviation for that season

\[
\frac{y_t}{\hat{y}_t} = \frac{T_t \times S_t \times R_t}{R_t} = S_t \times R_t
\]

here, we assume the multiplicative model

the computed ratios isolate the seasonal and random variation from the overall trend*

*) given that no additional cyclical variation exists
Example for Seasonal Effects

- Calculate the quarterly seasonal indexes for hotel occupancy rate in order to measure seasonal variation
- Data:

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1996</td>
<td>1</td>
<td>0.561</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.702</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.8</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.568</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997</td>
<td>1</td>
<td>0.575</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.738</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.868</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.605</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1998</td>
<td>1</td>
<td>0.594</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.738</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.729</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1999</td>
<td>1</td>
<td>0.622</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.708</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.806</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.632</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000</td>
<td>1</td>
<td>0.665</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>0.835</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>0.873</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>0.67</td>
</tr>
</tbody>
</table>

This example is taken from the course “Regression Analysis” at University of Umeå, Department of Statistics
Example for Seasonal Effects

• First step: compute trend from the data
  – e.g., linear regression

\[
\hat{y} = 0.639368 + 0.005246t
\]

<table>
<thead>
<tr>
<th>Time (t)</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.561</td>
</tr>
<tr>
<td>2</td>
<td>0.702</td>
</tr>
<tr>
<td>3</td>
<td>0.800</td>
</tr>
<tr>
<td>4</td>
<td>0.568</td>
</tr>
<tr>
<td>5</td>
<td>0.575</td>
</tr>
<tr>
<td>6</td>
<td>0.738</td>
</tr>
<tr>
<td>7</td>
<td>0.868</td>
</tr>
<tr>
<td>8</td>
<td>0.605</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Example for Seasonal Effects

- Second step: compute ratios $y_t/\hat{y}_t$

<table>
<thead>
<tr>
<th>$t$</th>
<th>$y_t$</th>
<th>$\hat{y}_t$</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.561</td>
<td>0.645</td>
<td>0.561/0.645=0.870</td>
</tr>
<tr>
<td>2</td>
<td>0.702</td>
<td>0.650</td>
<td>0.702/0.650=1.08</td>
</tr>
<tr>
<td>3</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

=0.639368+0.005245*t

No trend is observed, but seasonality and randomness still exist.
Example for Seasonal Effects

- Third step: compute average ratios by season

Average ratio for quarter 1: \((.870 + .864 + .865 + .879 + .913)/5 = .878\)

Average ratio for quarter 2: \((1.080 + 1.100 + 1.067 + .993 + 1.138)/5 = 1.076\)

Average ratio for quarter 3: \((1.221 + 1.284 + 1.046 + 1.122 + 1.181)/5 = 1.171\)

Average ratio for quarter 4: \((.860 + .888 + .854 + .874 + .900)/5 = .875\)
Example for Seasonal Effects

- Interpretation of seasonal indexes:
  - ratio between the time series' value at a certain season and the overall seasonal average

- In our problem:
Example for Seasonal Effects

- Deseasonalizing time series
  - when ignoring seasonal effects, is there still an increase?

**Seasonally adjusted time series** = \( \frac{\text{Actual time series}}{\text{Seasonal index}} \)

Trend on deseasonalized time series: slightly positive
Determining the Periodicity

• There are methods of identifying and isolating those effects
  – given that the periodicity is known

• What if we don’t know the periodicity?
Determining the Periodicity

• Assumption: time series is a sum of sine waves
  – With different periodicity
  – Different representation of the time series

• The frequencies of those sine waves is called *spectrum*
  – *Fourier transformation* transforms between spectrum and series
  – Spectrum gives hints at the frequency of periodic effects
  – Details: see textbooks
Determining the Periodicity

- Example: three interfering sine waves with noise added
Determining the Periodicity

- The corresponding spectrum
Dealing with Random Variations

- Moving average of order n

\[
\frac{y_1 + y_2 + \cdots + y_n}{n}, \frac{y_2 + y_3 + \cdots + y_{n+1}}{n}, \frac{y_3 + y_4 + \cdots + y_{n+2}}{n}, \cdots
\]

- Key idea:
  - upcoming value is the average of the last n
  - cf.: nearest neighbors

- Properties:
  - Smoothes the data
  - Eliminates \textit{random} movements
  - Loses the data at the beginning or end of a series
  - Sensitive to outliers (can be reduced by weighted moving average)
Moving Average in RapidMiner and Python

- Python:
  - e.g., rolling_mean in pandas
- Alternatives for average:
  - median, mode, …
Moving Average and Decomposition

- Often, moving averages are used for the trend
  - instead of a linear trend
  - less susceptible to outliers
  - the remaining computations stay the same
Dealing with Random Variations

• Exponential Smoothing
  – $S_t = \alpha y_t + (1-\alpha)S_{t-1}$
  – $\alpha$ is a smoothing factor
  – recursive definition
    • in practice, start with $S_0 = y_0$

• Properties:
  – Smoothes the data
  – Eliminates random movements
    • and even seasonal effects for smaller values of $\alpha$
  – Smoothing values for whole series
  – More recent values have higher influence

Python: statsmodels package
Dealing with Random Variations

![Graph showing stock price movements over time with different alpha values]
Recap: Trend Analysis

• Allows to identify general trends (upward, downward)
• Overall approach:
  – eliminate all other components so that only the trend remains
• Method for factoring out seasonal variations
  – and compute deseasonalized time series
• Methods for eliminating with random variations (smoothing)
  – moving average
  – exponential smoothing
Time Series Prediction: Definition

http://xkcd.com/1245/
From Moving Averages to Autoregressive Models

• Recap moving average for smoothing
  – each value is replaced by the average of its surrounding ones

• Moving average for prediction
  – predict the average of the last n values
  – \( y_t = \frac{1}{n} \times (y_{t-1} + \ldots + y_{t-n}) \)

• Here: weights are uniform
  – advanced: weights are learned from the data
  – \( y_t = \delta_1 y_{t-1} + \delta_2 y_{t-2} + \ldots + \delta_n y_{t-n} + \beta + \varepsilon_t \)
  – just like linear regression learning
  – this is called an autoregressive model
    • i.e., regression trained on the time series itself
Autoregressive Models in RapidMiner / Python

- RapidMiner: only with a twist
  - generate windowed representation for learning first
  - learn linear model on top

```python
from statsmodels.tsa.ar_model import AR
lagged values/ lag variables
```

from statsmodels.tsa.ar_model import AR
Autoregressive Models
Autoregressive Models

• First observation:
  – we have learned a linear model using the lag values
  – but the prediction itself is not linear!

• Second observation:
  – periodicities are learned well

• Why?
  – e.g., given that we have a strong weekly trend
  – we will learn a high weight for $\delta_{t-7}$
  – multiple periodicities can also be learned
    • e.g., time series with weekly and monthly component
Extension of AR models

- **ARMA**
  - Fits an AR model
  - Fits a second model to estimate the errors made by the AR model
  - \( y_t = \delta_1 y_{t-1} + \delta_2 y_{t-2} + \ldots + \delta_p y_{t-p} + \beta + \gamma_1 \epsilon_{t-1} + \ldots + \gamma_q \epsilon_{q-1} \)

- **ARIMA**
  - Tries to predict a differenced model
    - i.e., the relative change of a time series instead of the absolute value
  - ARIMA models come with three parameters:
    - \( p \): number of terms in the AR part
    - \( q \): number of terms in the MA part
    - \( d \): number of times the time series is differenced
## Lag Variables for Nominal Prediction

<table>
<thead>
<tr>
<th>Date</th>
<th>Weather</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.</td>
<td>Sunny</td>
</tr>
<tr>
<td>2.1.</td>
<td>Cloudy</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Date</th>
<th>Weather-3</th>
<th>Weather-2</th>
<th>Weather-1</th>
<th>Weather</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1.</td>
<td>?</td>
<td>?</td>
<td>?</td>
<td>Sunny</td>
</tr>
<tr>
<td>2.1.</td>
<td>?</td>
<td>?</td>
<td>Sunny</td>
<td>Cloudy</td>
</tr>
<tr>
<td>3.1.</td>
<td>?</td>
<td>Sunny</td>
<td>Cloudy</td>
<td>Cloudy</td>
</tr>
<tr>
<td>4.1.</td>
<td>Sunny</td>
<td>Cloudy</td>
<td>Cloudy</td>
<td>Rainy</td>
</tr>
<tr>
<td>5.1.</td>
<td>Cloudy</td>
<td>Cloudy</td>
<td>Rainy</td>
<td>Cloudy</td>
</tr>
<tr>
<td>6.1.</td>
<td>Cloudy</td>
<td>Rainy</td>
<td>Cloudy</td>
<td>Sunny</td>
</tr>
<tr>
<td>7.1.</td>
<td>Rainy</td>
<td>Cloudy</td>
<td>Sunny</td>
<td>Sunny</td>
</tr>
<tr>
<td>8.1.</td>
<td>Cloudy</td>
<td>Sunny</td>
<td>Sunny</td>
<td>Sunny</td>
</tr>
<tr>
<td>9.1.</td>
<td>Sunny</td>
<td>Sunny</td>
<td>Sunny</td>
<td>Rainy</td>
</tr>
</tbody>
</table>
Lag Variables in Multivariate Series

- Also possible for multi-variate data

<table>
<thead>
<tr>
<th>Row No.</th>
<th>Date</th>
<th>Weather-2</th>
<th>Weather-1</th>
<th>Weather-0</th>
<th>Temperature-2</th>
<th>Temperature-1</th>
<th>Temperature-0</th>
<th>label</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>04.01.2013</td>
<td>sunny</td>
<td>cloudy</td>
<td>cloudy</td>
<td>23</td>
<td>24</td>
<td>28</td>
<td>cloudy</td>
</tr>
<tr>
<td>2</td>
<td>07.01.2013</td>
<td>cloudy</td>
<td>cloudy</td>
<td>cloudy</td>
<td>24</td>
<td>28</td>
<td>32</td>
<td>rainy</td>
</tr>
<tr>
<td>3</td>
<td>08.01.2013</td>
<td>cloudy</td>
<td>rainy</td>
<td>sunny</td>
<td>28</td>
<td>32</td>
<td>19</td>
<td>sunny</td>
</tr>
<tr>
<td>4</td>
<td>09.01.2013</td>
<td>cloudy</td>
<td>rainy</td>
<td>sunny</td>
<td>32</td>
<td>19</td>
<td>24</td>
<td>rainy</td>
</tr>
<tr>
<td>5</td>
<td>10.01.2013</td>
<td>rainy</td>
<td>sunny</td>
<td>rainy</td>
<td>19</td>
<td>24</td>
<td>25</td>
<td>cloudy</td>
</tr>
<tr>
<td>6</td>
<td>11.01.2013</td>
<td>sunny</td>
<td>rainy</td>
<td>cloudy</td>
<td>24</td>
<td>25</td>
<td>17</td>
<td>sunny</td>
</tr>
<tr>
<td>7</td>
<td>14.01.2013</td>
<td>rainy</td>
<td>cloudy</td>
<td>sunny</td>
<td>25</td>
<td>17</td>
<td>14</td>
<td>sunny</td>
</tr>
<tr>
<td>8</td>
<td>15.01.2013</td>
<td>cloudy</td>
<td>sunny</td>
<td>sunny</td>
<td>17</td>
<td>14</td>
<td>12</td>
<td>rainy</td>
</tr>
<tr>
<td>9</td>
<td>16.01.2013</td>
<td>sunny</td>
<td>sunny</td>
<td>rainy</td>
<td>14</td>
<td>12</td>
<td>26</td>
<td>sunny</td>
</tr>
<tr>
<td>10</td>
<td>17.01.2013</td>
<td>sunny</td>
<td>rainy</td>
<td>sunny</td>
<td>12</td>
<td>26</td>
<td>23</td>
<td>cloudy</td>
</tr>
<tr>
<td>11</td>
<td>18.01.2013</td>
<td>rainy</td>
<td>sunny</td>
<td>cloudy</td>
<td>26</td>
<td>23</td>
<td>24</td>
<td>cloudy</td>
</tr>
<tr>
<td>12</td>
<td>21.01.2013</td>
<td>sunny</td>
<td>cloudy</td>
<td>cloudy</td>
<td>23</td>
<td>24</td>
<td>28</td>
<td>cloudy</td>
</tr>
<tr>
<td>13</td>
<td>22.01.2013</td>
<td>cloudy</td>
<td>cloudy</td>
<td>cloudy</td>
<td>24</td>
<td>28</td>
<td>32</td>
<td>rainy</td>
</tr>
<tr>
<td>14</td>
<td>23.01.2013</td>
<td>cloudy</td>
<td>cloudy</td>
<td>rainy</td>
<td>28</td>
<td>32</td>
<td>19</td>
<td>sunny</td>
</tr>
<tr>
<td>15</td>
<td>24.01.2013</td>
<td>cloudy</td>
<td>rainy</td>
<td>sunny</td>
<td>32</td>
<td>19</td>
<td>24</td>
<td>rainy</td>
</tr>
<tr>
<td>16</td>
<td>25.01.2013</td>
<td>rainy</td>
<td>sunny</td>
<td>rainy</td>
<td>19</td>
<td>24</td>
<td>25</td>
<td>cloudy</td>
</tr>
<tr>
<td>17</td>
<td>28.01.2013</td>
<td>sunny</td>
<td>rainy</td>
<td>cloudy</td>
<td>24</td>
<td>25</td>
<td>17</td>
<td>sunny</td>
</tr>
</tbody>
</table>
Predicting with Exponential Smoothing

• Recap exponential smoothing
  – \( S_t = \alpha y_t + (1-\alpha)S_{t-1} \)
    • We can also understand \( S_t \) as a prediction of \( y_{t+1} \)
    • i.e., we predict the average of the last value and the last prediction

• By recursion, we can use exponential smoothing for prediction
  – i.e., predict one step into the future
    • then use this prediction as input to the next step
  – works OK for short forecasting windows
  – at some point, the predictions usually diverge
Predicting with Exponential Smoothing

The green line has some “delay”
Double Exponential Smoothing

• Smaller values for $\alpha$:
  – more cancellation of random noise, but
  – exponential smoothing takes longer to adapt to trend

• With a trend, the smoothed time series will rise/fall over time
  – $S_t = \alpha y_t + (1-\alpha)(S_{t-1} + b_{t-1})$
  – $b_t = \beta(S_t - S_{t-1}) + (1-\beta)b_{t-1}$

• Explanation:
  – $S_t - S_{t-1}$ describes the change of the estimate
  – $b$ is the exponentially smoothed time series of those changes

• $S$ is called level smoothing, $b$ is called trend smoothing
Double Exponential Smoothing: Example
Triple Exponential Smoothing

- Double exponential smoothing
  - Uses level and trend, but no seasonality

- Triple exponential smoothing (also known as Holt Winters Method)
  - Introduces seasonal component
    - \( S_t = \alpha (y_t - c_{t-L}) + (1-\alpha)(S_{t-1} + b_{t-1}) \)
    - \( b_t = \beta (S_t - S_{t-1}) + (1-\beta)b_{t-1} \)
    - \( c_t = \gamma (y_t - S_t) + (1-\gamma)c_{t-L} \)

Most recent value and its trend component

L is the cycle length of the seasonality
Triple Exponential Smoothing

• Cycle length L
  – counted in number of observations

• Examples:
  – weekly cycles, one observation = one day: 7
  – yearly cycles, one observation = one month: 12
  – hourly cycles, one observation = one second: 3600
Triple Exponential Smoothing
Holt Winters in RapidMiner and Python

- Parameters:
  - $\alpha$, $\beta$, $\gamma$
  - period length

- Python implemention:
  - can also estimate parameters
  - as to fit the given data best

- Both implementations:
  - have additive and multiplicative variant
  - multiplicative often works better

```python
from statsmodels.tsa.holtwinters import ExponentialSmoothing
```
Selecting an Exponential Smoothing Model

- Taken from Alan Wan, Forecasting Methods for Business

- No trend or seasonal pattern? → Single Exponential Smoothing Method
- Linear trend and no seasonal pattern? → Holt’s Trend Corrected Exponential Smoothing Method
- Both trend and seasonal pattern? → Holt-Winters Methods
- Use Other Methods
Missing Values in Series Data

• Remedies in non-series data:
  – replace with average, median, most frequent
  – Imputation (e.g., k-NN)
  – replace with most frequent
  – ...

• What happens if we apply those to time series?
Missing Values in Series Data

- Original time series
  - with missing values inserted
Missing Values in Series Data

- Replace with average
Missing Values in Series Data

• Alternatives
  – Linear interpolation
  – Replace with previous
  – Replace with next
  – K-NN imputation
    • Essentially: this is the average of previous and next
Missing Values in Series Data

- Linear interpolation plotted
Evaluating Time Series Prediction

- So far, our gold standard has been 10-fold cross validation
  - Divide data into 10 equal shares
  - Random sampling:
    - Each data point is randomly assigned to a fold
Evaluating Time Series Prediction

• Using Cross Validation?
Evaluating Time Series Prediction

• Variant 1
  – Use hold out set *at the end* of the training data
  – E.g., train on 2000-2015, evaluate on 2016

• Variant 2:
  – Sliding window evaluation
  – E.g., train on one year, evaluate on consecutive year
Wrap-up

• Time series data is data sequentially collected at different times

• Analysis methods discussed in this lecture
  – frequent pattern mining
  – trend analysis
  – different prediction methods
Questions?