Machine learning is important

Knowledge Discovery (KDD) Process

- This is a view from typical database systems and data warehousing communities
- Data mining plays an essential role in the knowledge discovery process

The “Dumb User” Perspective

Weka, RapidMiner, MSSAS, Clementine, SPSS, R, ...

Knowledge Discovery (KDD) Process

- Data cleaning
- Data integration
- Task-relevant data selection
- Data warehouse
- Data mining
- Pattern evaluation
- Data integration

Summary so far

- Different types of data
- Data preparation (ETL, cleaning, ...)
- Statistics/significance
- Visualisation
- Large data – algorithmics, fast counting
- Queries/reporting, OLAP
- Machine learning
- Business value

Today

- Stream data
- Ordered sequences
- Pattern Discovery in sequences
- (Web) Information Retrieval
- Social Network Analysis
Streams, time series

- Time
- Sequence order and position
- Continuously arriving data

- Use or lose ...

Wikipedia

- **Data Stream Mining** is the process of extracting knowledge structures from continuous, rapid data records. A data stream is an ordered sequence of instances that in many applications of data stream mining can be read only once or a small number of times using limited computing and storage capabilities. Examples of data streams include computer network traffic, phone conversations, ATM transactions, web searches, and sensor data. Data stream mining can be considered a subfield of data mining, machine learning, and knowledge discovery.

Software

- **RapidMiner**: free open-source software for knowledge discovery, data mining, and machine learning also featuring data stream mining, learning time-varying concepts, and tracking drifting concept (if used in combination with its data stream mining plugin (formerly: concept drift plugin))

Literature


- In many data stream mining applications, the goal is to **predict the class or value of new instances in the data stream** given some knowledge about the class membership or values of previous instances in the data stream. Machine learning techniques can be used to learn this prediction task from labeled examples in an automated fashion.

- In many applications, the distribution underlying the instances or the rules underlying their labeling may change over time, i.e. the goal of the prediction, the class to be predicted or the target value to be predicted, may change over time. This problem is referred to as **concept drift**.

- **MOA (Massive Online Analysis)**: free open-source software specific for mining data streams with concept drift. It contains a prequential evaluation method, the EDDM concept drift methods, a reader of ARFF real datasets, and artificial stream generators as SEA concepts, STAGGER, rotating hyperplane, random tree, and random radius based functions. MOA supports bi-directional interaction with **Weka (machine learning)**.
Literature on Stream Mining


Characteristics of Data Streams

- Data Streams
  - Data streams—continuous, ordered, changing, fast, huge amount
  - Traditional DBMS—data stored in finite, persistent data sets

- Characteristics
  - Huge volumes of continuous data, possibly infinite
  - Fast changing and requires fast, real-time response
  - Data stream captures nicely our data processing needs of today
  - Random access is expensive—single scan algorithm (can only have one look)
  - Store only the summary of the data seen thus far
  - Most stream data are at pretty low-level or multi-dimensional in nature, needs multi-level and multi-dimensional processing

Stream Data Applications

- Telecommunication calling records
- Business: credit card transaction flows
- Network monitoring and traffic engineering
- Financial market: stock exchange
- Engineering & industrial processes: power supply & manufacturing
- Sensor, monitoring & surveillance: video streams, RFIDs
- Web logs and Web page click streams
- Massive data sets (even saved but random access is too expensive)

DBMS versus DSMS

- Persistent relations
- One-time queries
- Random access
- "Unbounded" disk store
- Only current state matters
- No real-time services
- Relatively low update rate
- Data at any granularity
- Assume precise data
- Access plan determined by query processor, physical DB design

- Transient streams
- Continuous queries
- Sequential access
- Bounded main memory
- Historical data is important
- Real-time requirements
- Possibly multi-GB arrival rate
- Data at fine granularity
- Data stale/imprecise
- Unpredictable/variable data arrival and characteristics

Architecture: Stream Query Processing

- SDMS (Stream Data Management System)
- Continuous Query
- Results
- Multiple streams
- Scratch Space (Main memory and/or Disk)

Challenges of Stream Data Processing

- Multiple, continuous, rapid, time-varying, ordered streams
- Main memory computations
- Queries are often continuous
  - Evaluated continuously as stream data arrives
  - Answer updated over time
- Queries are often complex
  - Beyond element-at-a-time processing
  - Beyond stream-at-a-time processing
  - Beyond relational queries (scientific, data mining, OLAP)
- Multi-level/multi-dimensional processing and data mining
  - Most stream data are at low-level or multi-dimensional in nature
Processing Stream Queries

- **Query types**
  - One-time query vs. continuous query (being evaluated continuously as stream continues to arrive)
  - Predefined query vs. ad-hoc query (issued on-line)

- **Unbounded memory requirements**
  - For real-time response, main memory algorithm should be used
  - Memory requirement is unbounded if one will join future tuples

- **Approximate query answering**
  - With bounded memory, it is not always possible to produce exact answers
  - High-quality approximate answers are desired

- **Data reduction and synopsis construction methods**
  - Sketches, random sampling, histograms, wavelets, etc.

Methodologies for Stream Data Processing

- **Major challenges**
  - Keep track of a large universe, e.g., pairs of IP address, not ages

- **Methodology**
  - Synopses (trade-off between accuracy and storage)
  - Use synopsis data structure, much smaller (O(\log N) space) than their base data set (O(N) space)
  - Compute an approximate answer within a small error range (factor \(c\) of the actual answer)

- **Major methods**
  - Random sampling
  - Histograms
  - Sliding windows
  - Multi-resolution model
  - Sketches
  - Randomized algorithms

Stream Data Processing Methods (1)

- **Random sampling (but without knowing the total length in advance)**
  - Reserve sampling: maintain a set of candidate tuples in the reservoir, which form a true random sample of the element seen so far in the stream.
  - As the data stream flow, every new element has a certain probability \(\alpha\) of replacing an old element in the reservoir.

- **Sliding windows**
  - Make decisions based only on recent data of sliding window size \(w\)
  - An element arriving at time \(t\) expires at time \(t + w\)

- **Histograms**
  - Approximate the frequency distribution of element values in a stream
  - Partition data into a set of contiguous buckets
  - Equal-width (equal value range for buckets) vs. V-optimal (minimizing frequency variance within each bucket)

- **Multi-resolution models**
  - Popular models: balanced binary trees, micro-clusters, and wavelets

Approximate Query Answering in Streams

- **Sliding windows**
  - Only over sliding windows of recent stream data
  - Approximation but often more desirable in applications

- **Batched processing, sampling and synopses**
  - **Batched** if update is fast but computing is slow
    - Compute periodically, not very timely
  - **Sampling** if update is slow but computing is fast
    - Compute using sample data, but not good for joins, etc.

- **Synopsis data structures**
  - Maintain a small synopsis or sketch of data
  - Good for querying historical data

- **Blocking operators, e.g., sorting, avg, min, etc.**
  - **Blocking** if unable to produce the first output until seeing the entire input

Projects on DSMS (Data Stream Management System)

- **Research projects and system prototypes**
  - **STREAM** (Stanford): A general-purpose DSMS
  - Cougar (Cornell): sensors
  - Aurora (Brown/MIT): sensor monitoring, dataflow
  - Hancock (AT&T): telecom streams
  - Nagios (OGI/Wisconsin): Internet XML databases
  - OpenCQ (Georgia Tech): triggers, incr. view maintenance
  - Tapestry (Xerox): pub/sub content-based filtering
  - Telegraph (Berkeley): adaptive engine for sensors
  - Tradebot (www.tradebot.com): stock tickers & streams
  - Tribeca (Bellcore): network monitoring
  - MAIDS (UIUC/NCSA): Mining Alarming Incidents in Data Streams

Stream Data Mining vs. Stream Querying

- **Stream mining**—A more challenging task in many cases
  - It shares most of the difficulties with stream querying
  - But often requires less “precision”, e.g., no join, grouping, sorting
  - Patterns are hidden and more general than querying
  - It may require exploratory analysis
  - Not necessarily continuous queries

- **Stream data mining tasks**
  - Multi-dimensional on-line analysis of streams
  - Mining outliers and unusual patterns in stream data
  - Clustering data streams
  - Classification of stream data
Concept drift

- In many applications, the distribution underlying the instances or the rules underlying their labeling may change over time, i.e., the goal of the prediction, the class to be predicted or the target value to be predicted, may change over time. This problem is referred to as concept drift.

Episode Rules

- Association rules applied to sequences of events.
- **Episode** – set of event predicates and partial ordering on them

Episode Rules

- **Association rules applied to sequences of events.**
- **Episode** – set of event predicates and partial ordering on them

Basics

- **Association rules describe how things occur together in the data**
  - E.g., "**IF** an alarm has certain properties, **THEN** it will have other given properties"

- **Episode rules describe temporal relationships between things**
  - E.g., "**IF** a certain combination of alarms occurs within a time period, **THEN** another combination of alarms will occur within a time period"

Basics

- **Association rules describe how things occur together in the data**
  - E.g., "**IF** an alarm has certain properties, **THEN** it will have other given properties"

- **Episode rules describe temporal relationships between things**
  - E.g., "**IF** a certain combination of alarms occurs within a time period, **THEN** another combination of alarms will occur within a time period"

Episodes

- **Partially ordered set of pages**
- **Serial episode** – totally ordered with time constraint
- **Parallel episode** – partially ordered with time constraint
- **General episode** – partially ordered with no time constraint
• Data:
  – Data is a set $R$ of events
  – Every event is a pair $(A, t)$, where
    • $A \in R$ is the event type (e.g., alarm type)
    • $t$ is an integer, the occurrence time of the event
  – Event sequence $s$ on $R$ is a triple $(s, T_s, T_e)$
    • $T_s$ is starting time and $T_e$ is ending time
    • $T_s < T_e$ are integers
    • $s = \langle (A_1, t_1), (A_2, t_2), \ldots, (A_n, t_n) \rangle$
    • $A_i \in R$ and $T_s \leq t_i < T_e$ for all $i=1, \ldots, n$
• Example alarm data sequence:
  – $s = \langle (D, 10), (C, 20), \ldots, (A, 150) \rangle$
  – $T_s$ (starting time) = 10 and $T_e$ (ending time) = 150
  – Note: There needs not to be events on every time slot!
• Episodes:
  – An episode is a pair $(V, \leq)$
    • $V$ is a collection of event types, e.g., alarm types
    • $\leq$ is a partial order on $V$
  – Given a sequence $S$ of alarms, an episode $\alpha = (V, \leq)$ occurs within $S$ if there is a way of satisfying the event types (e.g., alarm types) in $V$ using the alarms of $S$ so that the partial order $\leq$ is respected
  – Intuitively: episodes consist of alarms that have certain properties and occur in a certain partial order
• The most useful partial orders are:
  – Total orders
    • The predicates of each episode have a fixed order
    • Such episodes are called serial (or "ordered")
  – Trivial partial orders
    • The order of predicates is not considered
    • Such episodes are called parallel (or "unordered")
  – Complicated?
    • Not really, let’s take some clarifying examples

DAG for Episode

Basics

• Data:
  – Data is a set $R$ of events
  – Every event is a pair $(A, t)$, where
    • $A \in R$ is the event type (e.g., alarm type)
    • $t$ is an integer, the occurrence time of the event
  – Event sequence $s$ on $R$ is a triple $(s, T_s, T_e)$
    • $T_s$ is starting time and $T_e$ is ending time
    • $T_s < T_e$ are integers
    • $s = \langle (A_1, t_1), (A_2, t_2), \ldots, (A_n, t_n) \rangle$
    • $A_i \in R$ and $T_s \leq t_i < T_e$ for all $i=1, \ldots, n$

Basics

• Example alarm data sequence:
  – $s = \langle (D, 10), (C, 20), \ldots, (A, 150) \rangle$
  – $T_s$ (starting time) = 10 and $T_e$ (ending time) = 150
  – Note: There needs not to be events on every time slot!

Basics

• Episodes:
  – An episode is a pair $(V, \leq)$
    • $V$ is a collection of event types, e.g., alarm types
    • $\leq$ is a partial order on $V$
  – Given a sequence $S$ of alarms, an episode $\alpha = (V, \leq)$ occurs within $S$ if there is a way of satisfying the event types (e.g., alarm types) in $V$ using the alarms of $S$ so that the partial order $\leq$ is respected
  – Intuitively: episodes consist of alarms that have certain properties and occur in a certain partial order

Basics

• The most useful partial orders are:
  – Total orders
    • The predicates of each episode have a fixed order
    • Such episodes are called serial (or "ordered")
  – Trivial partial orders
    • The order of predicates is not considered
    • Such episodes are called parallel (or "unordered")
  – Complicated?
    • Not really, let’s take some clarifying examples

Course on Data Mining

Course on Data Mining

Course on Data Mining

Course on Data Mining
The name of the WINEPI method comes from the technique it uses: a sliding window.

Intuitively:
- A window is slided through the event-based data sequence.
- Each window "snapshot" is like a row in a database.
- The collection of these "snapshots" forms the rows in the database.

Complicated?
- Not really, let's take a clarifying example.

By definition, the first and the last windows on a sequence extend outside the sequence, so that the last window contains only the first time point of the sequence, and the last window only the last time point.

Example alarm data sequence:

D    C    A     B     D    A    B     C

The window width is 40 seconds, last point excluded.
The first/last window contains only the first/last event.

Formally, given a set \( E \) of event types, an event sequence \( S = (s, T, e) \) is an ordered sequence of events \( e \) such that \( e \) event \( s \) event \( e \), for all \( i = 1, \ldots, n \), and \( T \) is an event such that the window width is \( w \).

\[ \alpha \]

\[ \beta \]

\[ \gamma \]

\[ \delta \]

\[ \epsilon \]

\[ \zeta \]

\[ \eta \]

\[ \theta \]

\[ \iota \]

\[ \kappa \]

\[ \lambda \]

\[ \mu \]

\[ \nu \]

\[ \xi \]

\[ \omik \]

\[ \pi \]

\[ \rho \]

\[ \sigma \]

\[ \tau \]

\[ \upsilon \]

\[ \phi \]

\[ \chi \]

\[ \psi \]

\[ \omega \]

\[ \alpha' \]

\[ \beta' \]

\[ \gamma' \]

\[ \delta' \]

\[ \epsilon' \]

\[ \zeta' \]

\[ \eta' \]

\[ \theta' \]

\[ \iota' \]

\[ \kappa' \]

\[ \lambda' \]

\[ \mu' \]

\[ \nu' \]

\[ \xi' \]

\[ \omik' \]

\[ \pi' \]

\[ \rho' \]

\[ \sigma' \]

\[ \tau' \]

\[ \upsilon' \]

\[ \phi' \]

\[ \chi' \]

\[ \psi' \]

\[ \omega' \]
Mika Klemettinen and Pirjo Moen

WINEPI Approach

- When searching for the episodes, a frequency threshold (cf. support threshold with association rules) \( \min_f \) is used
- Episode \( \alpha \) is frequent if \( F(\alpha, \kappa, \text{win}) > \min_f \), i.e., "if the frequency of \( \alpha \) exceeds the minimum frequency threshold within the data sequence \( s \) and with window width \( \text{win} \)"
- \( F(s, \text{win}, \min_f) \): a collection of frequent episodes in \( s \) with respect to \( \text{win} \) and \( \min_f \)
- Apriori trick holds: if an episode \( \alpha \) is frequent in an event sequence \( s \), then all subepisodes \( \beta < \alpha \) are frequent

WINEPI Algorithm

- Input: A set \( R \) of event/alarm types, an event sequence \( s \) over \( R \), a set \( E \) of episodes, a window width \( \text{win} \), and a frequency threshold \( \min_f \)
- Output: The collection \( F(s, \text{win}, \min_f) \)
- Method:
  1. Compute \( C_i := \{ \alpha \in E | |\alpha| = i \} \);
  2. \( i := 1 \);
  3. While \( C_i \neq \emptyset \) do
  4. \( i^* \) compute \( F_i(s, \text{win}, \min_f) := \{ \alpha \in C_i | F(s, \kappa, \text{win}) \geq \min_f \} \);
  5. \( i := i + 1 \);
  6. \( i^* \) compute \( C_i := \{ \alpha \in E | |\alpha| = i \} \), and \( \beta \in F_i(s, \text{win}, \min_f) \) for all \( \beta \in E, |\beta| < |\alpha| \);
- \(* = \) database pass, \(* = \) candidate generation

Intuitively:

- WINEPI rules are like association rules, but with an additional time aspect:
- If events (alarms) satisfying the rule antecedent (left-hand side) occur in the right order within \( \text{win} \) time units, then also the rule consequent (right-hand side) occurs in the location described by \( s \), also within \( \text{win} \) time units

\[ \text{antecedent} \Rightarrow \text{consequent} \ \text{[window width] (f, c)} \]
• Parallel episodes:
  – For each candidate $\alpha$ maintain a counter $\alpha$.event_count: how many events of $\alpha$ are present in the window
  – When $\alpha$.event_count becomes equal to $|\alpha|$, indicating that $\alpha$ is entirely included in the window, save the starting time of the window in $\alpha$.inwindow
  – When $\alpha$.event_count decreases again, increase the field $\alpha$.freq_count by the number of windows where $\alpha$ remained entirely in the window

• Serial episodes: use a state automata

• Example alarm data sequence:

  By sliding the window, we'll get 11 windows ($U_1$ to $U_{11}$):

  Frequency threshold is set to 40%, i.e., an episode has to occur at least in 5 of the 11 windows

• Suppose that the task is to find all parallel episodes:
  – First, create singletons, i.e., parallel episodes of size 1 ($A, B, C, D$)
  – Then, recognize the frequent singletons (here all are)
  – From those frequent episodes, build candidate episodes of size 2: $AB, AC, AD, BC, BD, CD$
  – Then, recognize the frequent parallel episodes (here all are)
  – From those frequent episodes, build candidate episodes of size 3: $ABC, ABD, ACD, BCD$
  – When recognizing the frequent episodes, only ABD occurs in more than four windows
  – There are no candidate episodes of size four

• Episode frequencies and example rules with WINEPI:
  - $D$: 73%
  - $C$: 73%
  - $A$: 64%
  - $B$: 64%
  - $D \Rightarrow A \{40\}$ (55%, 75%)
  - $D \Rightarrow C \{45\}$
  - $D \Rightarrow B \{45\}$ (45%, 82%)
  - $A \Rightarrow C \{45\}$
  - $B \Rightarrow A \{45\}$
  - $C \Rightarrow B \{45\}$
  - $A \Rightarrow B \{45\}$
  - $D \Rightarrow A \{45\}$
• Data:
  – Alarms from a telecommunication network
  – 73,000 events (7 weeks), 287 event types
  – Parallel and serial episodes
  – Window widths (W): 10-120 seconds
  – Window movement = W/10
  – min_fr = 0.003 (0.3%), frequent: about 100 occurrences
  – 90 MHz Pentium, 32MB memory, Linux operating system. The data resided in a 3.0 MB flat text file

WINEPI: Experimental Results

<table>
<thead>
<tr>
<th>Window width (s)</th>
<th>Serial episodes #frequent</th>
<th>Parallel episodes #frequent</th>
<th>time (s)</th>
<th>#frequent</th>
<th>time (s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>16</td>
<td>31</td>
<td>10</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>31</td>
<td>63</td>
<td>17</td>
<td>9</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>57</td>
<td>117</td>
<td>33</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>60</td>
<td>87</td>
<td>186</td>
<td>56</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>80</td>
<td>145</td>
<td>271</td>
<td>95</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>245</td>
<td>372</td>
<td>139</td>
<td>21</td>
<td></td>
</tr>
<tr>
<td>120</td>
<td>359</td>
<td>478</td>
<td>189</td>
<td>22</td>
<td></td>
</tr>
</tbody>
</table>

WINEPI Approach

• One shortcoming in WINEPI approach:
  – Consider that two alarms of type A and one alarm of type B occur in a window
  – Does the parallel episode consisting of A and B appear once or twice?
  – If once, then with which alarm of type A?

Alternative approach to discovery of episodes

• No sliding windows
• For each potentially interesting episode, find out the exact occurrences of the episode
  • Advantages: easy to modify time limits, several time limits for one rule.
    "If A and B occur within 15 seconds, then C follows within 30 seconds"
  • Disadvantages: uses a lot of space

MINEPI Approach

• Formally, given a episode \( \alpha \) and an event sequence \( S \), the interval \([t_s, t_e]\) is a minimal occurrence \( \alpha \) of \( S \):
  – If \( \alpha \) occurs in the window corresponding to the interval
  – If \( \alpha \) does not occur in any proper subinterval

The set of minimal occurrences of an episode \( \alpha \) in a given event sequence is denoted by \( \text{mo}(\alpha) \):

\[
\text{mo}(\alpha) = \{ [t_s, t_e] | [t_s, t_e] \text{ is a minimal occurrence of } \alpha \}
\]
Pattern Discovery

1. Choose the language (formalism) to represent the patterns (search space)
2. Choose the rating for patterns, to tell which is “better” than others
3. Design an algorithm that finds the best patterns from the pattern class, fast.


MINEPI Approach

- Informally, a MINEPI episode rule gives the conditional probability that a certain combination of events (alarms) occurs within some time bound, given that another combination of events (alarms) has occurred within a time bound.
- Formally, an episode rule is \( \beta [\text{win}_1] \Rightarrow \alpha [\text{win}_2] \).
- \( \beta \) and \( \alpha \) are episodes such that \( \beta \prec \alpha \) (\( \beta \) is a subepisode of \( \alpha \)).
- If episode \( \beta \) has a minimal occurrence at interval \([t, t_s]\) with \( t_s - t \leq \text{win}_1 \), then episode \( \alpha \) occurs at interval \([t, t'_s]\) for some \( t'_s \) such that \( t'_s - t \leq \text{win}_2 \).

Patterns: AT

IUPAC: W H AT

Patterns: [AT][ACT]AT
Cluster of co-expressed genes, pattern discovery in regulatory regions

Expression profiles

600 basepairs

Retrieve

Upstream regions

Find patterns over-represented within cluster


Binomial or hypergeometric distribution

Background - ALL upstream sequences

Cluster: π occurs 3 times

P(3,6,0.2) is probability of having ≥3 matches in 6 sequences

5 out of 25, p = 0.2  P(π,3,6,0.2) = 0.0989

Eukaryotic genome can be thought of as six Levels of DNA structure.

The loops at Level 4 range from 0.5kb to 100kb in length.

If these loops were stabilized then the genes inside the loop would not be expressed.

DNA determines function (?)
YGR128C + 100

101 Sequences relative to ORF start

Sequence patterns: the basis of the SPEXS

SPEXS: general algorithm

1. $S$ = input sequences ( $|S|$=n )
2. $e$ = empty pattern, $e$.pos = {1,...,n}
3. enqueue(order, e )
4. while $p$ = dequeue(order )
5. generate all allowed extensions $p'$ of $p$ (& $p'$:pos)
6. enqueue(order, $p'$, priority($p'$) )
7. enqueue(output, $p'$, fitness($p'$) )
8. while $p$ = dequeue(output )
9. Output p

Applications to biotechnology:
- Functional elements in proteins (2002: 32 cit)


These hits result in a PWM:

PWMA-based on all previous hits, here shown highest-scoring occurrences in blue

<table>
<thead>
<tr>
<th>Pattern</th>
<th>In cluster</th>
<th>Total in</th>
<th>Ratio</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>G.GATGAG.T</td>
<td>39</td>
<td>193</td>
<td>13.24</td>
<td>2.490e-33</td>
</tr>
<tr>
<td>TG.AAA.TTT</td>
<td>53</td>
<td>538</td>
<td>6.46</td>
<td>3.248e-31</td>
</tr>
<tr>
<td>TG.AAAA.TTT</td>
<td>45</td>
<td>333</td>
<td>8.86</td>
<td>1.696e-31</td>
</tr>
<tr>
<td>G.GATGAGTGT</td>
<td>61</td>
<td>1295</td>
<td>3.09</td>
<td>1.441e-19</td>
</tr>
<tr>
<td>+1: TG.AAA.TTT</td>
<td>89</td>
<td>3836</td>
<td>1.52</td>
<td>6.126e-12</td>
</tr>
<tr>
<td>+2: TG.AAAA.TTT</td>
<td>76</td>
<td>2190</td>
<td>2.27</td>
<td>1.654e-18</td>
</tr>
</tbody>
</table>


GPCR coupling

- Signal: Agonist
- Current perspective
- GPCRs
- G-protein
- Effector enzymes
- Intracellular messengers

GPCRs

- Sense and transduce a huge variety of:
  - hormones, neurotransmitters, peptides, photons, tastants, odorants, ions, peptidases, and pheromones
- Involved in practically all aspects of human physiology, and also pathophysiology
- Estimates of 800-2000 members of the superfamily in the human genome
Drugs Acting Upon GPCRs

- β-Adrenoceptor agonists/antagonists
  - Asthma, high blood pressure, anxiety
- L-DOPA (dopamine precursor)
  - Agent of choice for Parkinson’s disease
- Histamine H1 antagonists
  - Allergic and anaphylactic reactions, travel sickness
- Histamine H2 antagonists
  - Ulcer treatment
- Opioid agonists
  - Powerful analgesics
- 5-HT₁D antagonists
  - Migraine

Diseases Associated with GPCRs

- Colour blindness
  - Cone opsin receptor
- Hyperthyroidism
  - Thyrotropin receptor
- X-linked diabetes insipidus
  - V2 vasopressin receptor

Receptor-G Protein Coupling

- We know:
  - 20 G protein α subunits, these determine the functional class of the G protein heterotrimer. By sequence similarity these can be split into 4 groups.
  - From a functional perspective there are three well-recognised classes of G proteins
    - Gₛ stimulate adenylate cyclase (includes Gₐₒ)
    - Gᵢᵢ inhibit adenylate cyclase (includes Gᵢₒ, Gᵢₜ)
    - Gᵢᵢ₁ stimulate PI hydrolysis
  - Gᵢᵧ unknown function
- No easy way to predict from GPCR it’s coupling specificity to Gₛ, Gᵢᵢ₁, Gᵢᵢᵢ₁
- GPCRs may exhibit selective, or promiscuous coupling

Computational Approaches

Structural Analysis

We have only one crystal structure - 1f88

Our Computational Approach

- Using a new membrane topology prediction algorithm (designed specifically for GPCRs), we constrained our pattern search to the intracellular domains of ~100 receptor sequences with well-characterised, and non-promiscuous coupling (split into Gₛ, Gᵢᵢ and Gᵢᵢᵢ₁)

Receptor Match Positions

Croning, Vilo, Möller, ISMB 2001
**More Topics**

- **Information Retrieval**
- **Text Mining**
- **Web Mining**
- **Social Network Analysis**
  - friends, epidemiology, co-authoring, co-citation, espionage, ...
- **Graph Mining**

**Concepts**

**Information Retrieval** - the study of systems for representing, indexing (organising), searching (retrieving), and recalling (delivering) data.

**Information Filtering** - given a large amount of data, return the data that the user wants to see.

**Information Need** - what the user really wants to know; a query is an approximation to the information need.

**Query** - a string of words that characterizes the information that the user seeks

**Browsing** - a sequence of user interaction tasks that characterizes the information that the user seeks

**Documents**

- News, articles
- Laws, legal documents
- Scientific publications, patents
- E-mail
- Technical documents
- Books
- Encyclopaediae
- Dictionaries
- ...

**Information Retrieval**

- DB of indexed documents
- Query
- Find documents relevant to query
### More features

- Metadata
- Context
- Hypertext – xrefs
- Language
- Structured vs unstructured
- Semantics
- Tags
- ...

---

### Classic IR Goal

- Classic relevance
  - For each query \( Q \) and stored document \( D \) in a given corpus assume there exists relevance \( \text{Score}(Q, D) \)
  - \( \text{Score} \) is average over users \( U \) and contexts \( C \)
  - Optimize \( \text{Score}(Q, D) \) as opposed to \( \text{Score}(Q, D, U, C) \)
  - That is, usually:
    - Context ignored
    - Individuals ignored
    - Corpus predetermined


---

### The Notion of Relevance

- Data retrieval: semantics tied to syntax
- Information retrieval: ambiguous semantics
- Relevance:
  - Depends on the user
  - Depends on the context (task, time, etc.)
  - Corollary: The Perfect IR System does not exist

---

### Evaluation: First Quality, next Efficiency

- Precision - \( \frac{TP}{TP + FP} \)
- Recall - \( \frac{TP}{TP + FN} \)
- \( TP \): True Positives
- \( FP \): False Positives
- \( TN \): True Negatives
- \( FN \): False Negatives

---

### User Needs

- Need (Broder 2002)
  - Informational: want to learn about something \((-40\% / 65\%)\)
    - e.g., low hemoglobin
  - Navigational: want to go to that page \((-25\% / 18\%)\)
    - e.g., United Airlines
  - Transactional: want to do something (web-mediated) \((-35\% / 20\%)\)
    - e.g., Edinburgh weather
    - Downloads
    - Shop
    - E.g., Gray areas
    - Car rental

---

An Introduction to IR in Web Mining, P2P/OML, Beijing
Using the Context

Example: I want information about Santiago

- **Context**
  - Family in Chile
  - Catholic
  - Travelling to Cuba
  - Lives in Argentina
  - Located in Santo Domingo
  - Architect
  - Spanish movies fan
  - Baseball fan

- **Probable Answer**
  - Santiago de Chile
  - Santiago de Compostela
  - Santiago de Cuba
  - Santiago del Estero
  - Santiago de los Caballeros
  - Santiago Calatrava
  - Santiago Segura
  - Santiago Bernardo

Web search

- data mining university of tartu fall 2009
- Java
- Paris Hilton
- climate change

Corporate bias?
What does WWW look like?

Fig. 7. The bowtie structure of the Web: Plot (a) shows the 4 parts: IN, OUT, SCC, and TENDRILS [Broder et al. 2000]. Plot (b) shows recursive bowties: subgraphs of the WWW can each be considered a bowtie. All these smaller bowties are connected by the navigational backbone of the main SCC of the Web [Dill et al. 2001].
PageRank

- Let $E$ be the adjacency matrix of the graph, and $L$ the row-stochastic version of $E$
- Each row of $E$ is normalized so that it sums to 1
- Authority score defined by $p_{n+1} = L^T p_n$
- problematic if the graph is not strongly connected, So:
  $p_{n+1} = \alpha L^T p_n + (1-\alpha) I/n$
- where $I$ is the matrix with all entries equal to 1
- and $\alpha \in (0,1)$, common value $\alpha = 0.85$

Topics

- Information Retrieval
- Text Mining
- Web Mining
- Social Network Analysis
- – friends, epidemiology, co-authoring, co-citation, espionage, …
- Graph Mining

Skype Fast Facts

- +170 million monthly connected users
- 30 million concurrent users
- +300 million minutes of Skype video/day
- 42% of all calls include video
- Goal: reach 1 billion users daily
- Context: 2.1 billion Internet users

Social network of 9/11 hijackers
Running Example

Hijackers by Flight

<table>
<thead>
<tr>
<th>Flight 77 : Pentagon</th>
<th>Flight 11 : WTC 1</th>
<th>Flight 175 : WTC 2</th>
<th>Flight 93 : PA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Khalid Al-Midhar</td>
<td>Jamel al-Suqami</td>
<td>Marwan Al-Shehhi</td>
<td>Saeed Alghamdi</td>
</tr>
<tr>
<td>Majid Moqed</td>
<td>Waleed M. Alshahr</td>
<td>Fayez Ahmed</td>
<td>Ahmed Alhamadi</td>
</tr>
<tr>
<td>Nawaf Alhamzi</td>
<td>Waleed Alshahr</td>
<td>Ahmed Alghamdi</td>
<td>Ahmed Alhamadi</td>
</tr>
<tr>
<td>Salem Alhamzi</td>
<td>Mohamed Atta</td>
<td>Hamza Alghamdi</td>
<td>Ziad Jarrahi</td>
</tr>
<tr>
<td>Hans Hanjour</td>
<td>Abdulaziz Alomari</td>
<td>Mohmed Alshahr</td>
<td></td>
</tr>
</tbody>
</table>

Semantic search: "Most popular drink that is available on bars that are visited by my friends"

http://yury.name/webguide/02webguide.pdf
Semantic search: “Most popular drink that is available on bars that are visited by my friends”

ABSTRACT

We present a study of anonymized data capturing a month of high-level communication activities within the world of the Facebook Messenger Instant-Messaging service. The study examines the social aspects of large numbers of people, rather than the actions and characteristics of individuals. The dataset contains extensive properties of 2.4 billion conversations among 240 million people. From this data, we construct a notion of social networks where each node represents an individual, and each inter-person edge, the largest social network constructed and analyzed to date. We report on multiple aspects of the dataset and resulting graph. We find that the graph is well-connected and robust to scale reduction. We introduce a measure of social cohesion for individuals that are captured by “the degree of separation” and find that the average path length among Messenger users is 4.4. We also find that people tend to communicate more with each other when they have similar age, location, and gender. People who have shorter conversations also tend to have longer duration than conversations with the same number of messages.

Keywords: Social networks, Communication networks, User demographics, Large-scale, Online communication.

Figure 2: (a) Distribution of login duration. (b) Duration of times when people are not logged into the system (times between login and logout).
Nov 19, 2011

- Facebook has released two papers about their network:
  - The Anatomy of the Facebook Social Graph
  - Four Degrees of Separation
Possible questions in SNA

- Prestige
- Centrality
- Co-citation/co-occurrence
- Radius/diameter
- ...
Conclusions on course:

- Data Mining is a rich and diverse field
- Driven by data, curiosity, business value
- Algorithmics, Statistics, Visualisation, ...
- Decision support/actionable information