Mining Complex Dynamic Data

Hans-Peter Kriegel
Irene Ntoutsi
Arthur Zimek
Ludwig-Maximilians-Univ. Munich, GERMANY

Myra Spiliopoulou
University of Magdeburg, GERMANY

Grigoris Tsoumakas
Aristotle University of Thessaloniki, GREECE

ECML PKDD Tutorial
Athens, Sept. 2011
Tutorial Structure

Block 1: Introduction
Block 2: Mining multi-label data
Block 3: Mining high-dimensional data
Block 4: Mining multi-relational data
Block 5: Outlook

Grigorios Tsoumakas
Arthur Zimek
Irene Ntoutsi
Myra Spiliopoulou
Tutorial Structure

Block 1: Introduction

Block 2: Mining multi-label data

Block 3: Mining high-dimensional data

Block 4: Mining multi-relational data

Block 5: Outlook

Grigorios Tsoumakas
Arthur Zimek
Irene Ntoutsi
Myra Spiliopoulou
Tutorial Presenters

**Myra Spiliopoulou**
Professor of Information Systems in the Faculty of Computer Science, Otto-von-Guericke-Univ. Magdeburg, and Chair of the Knowledge Management & Discovery (KMD) group. Co-founder of WebKDD workshop series, has co-organized the ECML-PKDD 2006 and NLDB’08, Tutorials Co-Chair of ICDM 2010 and Workshops Chair of ICDM 2011.

http://omen.cs.uni-magdeburg.de/itikmd

**Grigoris Tsoumakas**
Lecturer in the Department of Informatics of the Aristotle University of Thessaloniki and member of the Machine Learning and Knowledge Discovery (MLKD) group. His main research interests include ensemble methods and multi-label learning.

http://users.auth.gr/greg
Irene Ntoutsi

Postdoc researcher in the database systems and data mining group of Prof. Hans-Peter Kriegel at the Ludwig Maximilians University (LMU) Munich, Germany and scholar of the Alexander vor Humboldt Foundation. Her research interests include data mining over dynamic data with a current focus on high dimensional data.

http://www.dbs.ifi.lmu.de/cms/Irene_Ntoutsi

Arthur Zimek

Scientific assistant (lecturer) in the database systems and data mining group of Hans-Peter Kriegel at the Ludwig Maximilians University (LMU) Munich, Germany. His research interests include clustering (esp. clustering in high dimensional data) and outlier detection.

http://www.dbs.ifi.lmu.de/~zimek/
Motivation for the Tutorial

In recent years, many applications require mining from richer data types than conventional data(base) records:

- Social networks require the combination of activity recordings with content; recommendation engines require user ratings, customer transactions, item descriptions and user profiles; medical applications require a combination of different kinds of recordings on patients.

At the same time, mining tasks become more elaborate:

- Data are multi-faceted and adhere to many concepts; Data accumulate or form streams; they are dynamic and call for adaptation of the mining models.

In this tutorial, we discuss mining on complex data, putting the emphasis on learning and adaptation over streaming, dynamic data.
Block 2: Mining multi-label data

Multi-label data have become ubiquitous

- Text, music, video, biological data, web data

In several cases, such data come in the form of a stream

- News, emails, patents, organizational documents, urls

Learning from multi-label data streams inherits the challenges of multi-label learning and stream mining, while at the same time raising its own unique issues.

In Block 2, we discuss learning from multi-label data streams. We first introduce basic concepts and learning methods for static multi-label data and then discuss issues and approaches for streams of multi-label data.
Block 3: Mining high-dimensional data

Due to the advances in hardware and software more and more information is collected regarding our applications:

- Store all the details → high dimensional data
- Keep track of evolution → dynamic data

Knowledge extraction from high dimensional and dynamic data is challenging due to the so called “curse of dimensionality”.

In Block 3, we discuss methods for mining high dimensional dynamic data. We first discuss the challenges due to high dimensionality in static environments and then proceed to dynamic environments/ data streams.
Block 4: Mining multi-relational data

In a social network, people perform various kinds of activities, accessing different types of resources.

In database terms:

- Entity types: person, resource, opinion, ...
- Relation types: upload, download, annotate, reply, invite, ...

These data constitute a stream and reflect the evolution of the social network and of the entities in it.

In Block 4, we discuss methods for learning upon such a stream, and for visualizing network evolution. We also discuss a special case of particular importance for recommendation engines and further business applications: the evolution of customers.
Presentation Outline

☐ Block 1: Introduction

Block 2: Multi-label data
  - Static multi-label data
  - Dynamic multi-label data

Block 3: High-dimensional data

Block 4: Multi-relational data

Block 5: Conclusions and Outlook

(Grigoris Tsoumakas)
Multi-Label Data

Data with multiple binary target variables (labels)

For example, an article about Fukushima could be annotated with the following set of labels:

- {“nuclear crisis”, “Asia-Pacific”, “energy”, “environment”}

Main issues

- Exploit label structure to get improved performance
- Scale up to a large label dimension
Applications

(Semi) automated annotation of large object collections for information retrieval

- Text/web, image, video, audio, biology

Tag suggestion in Web 2.0 systems

Query categorization

Drug discovery

Direct marketing

Medical diagnosis

Multi-label data are ubiquitous
Notation

A $d$-dimensional input space, $\mathcal{X}$

- Numeric or nominal features

A set of $q$ output labels

- $\mathcal{Y} = \{\lambda_1, \lambda_2, \ldots, \lambda_q\}$

A multi-label dataset of $m$ training examples

- $S = \{(x_1, Y_1), (x_2, Y_2), \ldots, (x_m, Y_m)\}$
- where $x_i \in \mathcal{X}, Y_i \subseteq \mathcal{Y}$
Output Types – Learning Tasks (1/4)

Classification

- Produce a bipartition of the set of labels into a relevant (positive) and an irrelevant (negative) set

For example, given

- $\mathcal{Y} = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\}$
- An unobserved instance $\mathcal{x}$

produce a bipartition

- $P_x : \{\lambda_1, \lambda_4\}, N_x : \{\lambda_2, \lambda_3, \lambda_5\}$
Output Types – Learning Tasks (2/4)

Ranking

- Produce a ranking (total strict order) of all labels according to relevance to the given instance

For example, given

- $\mathcal{Y} = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\}$
- An unobserved instance $x$

produce a ranking

- $r(\lambda_3) < r(\lambda_2) < r(\lambda_4) < r(\lambda_5) < r(\lambda_1)$
- where $r(\lambda_i)$ denotes the position of label $\lambda_i$ in the ranking
Output Types – Learning Tasks (3/4)

Classification and Ranking

- Produce both a bipartition and a ranking of all labels
- Should be consistent: $\lambda_i \in P_x, \lambda_j \in N_x \Rightarrow r(\lambda_i) < r(\lambda_j)$

For example, given

- $\mathcal{Y} = \{\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5\}$
- An unobserved instance $x$

produce a bipartition and a ranking

- $P_x : \{\lambda_1, \lambda_4\}, N_x : \{\lambda_2, \lambda_3, \lambda_5\}$
- $r(\lambda_1) < r(\lambda_4) < r(\lambda_3) < r(\lambda_5) < r(\lambda_2)$
Output Types – Learning Tasks (4/4)

Probabilistic indexing

- Output the probability of relevance to each label

For example, given

- \( \mathcal{Y} = \{ \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5 \} \)
- An unobserved instance \( x \)

produce the probabilities

- \( p(\lambda_1|x) = 0.8, p(\lambda_2|x) = 0.2 \)
- \( p(\lambda_3|x) = 0.9, p(\lambda_4|x) = 0.4, p(\lambda_5|x) = 0.1 \)

a ranking can be obtained after tie-breaking
Evaluation Measures: A Taxonomy

Based on calculation

- Example-based are calculated separately for each test example and averaged across the test set
- Label-based are calculated separately for each label and then averaged across all labels

Based on the output of the learner

- Bipartition of the labels into relevant and irrelevant ones
- Ranking of the labels (example-based)
- Probability or score for each label
Example–Based Binary (1/2)

Notation

- $P_i$ is the set of predicted labels for instance $x_i$
- $Y_i$ the set of actual labels for instance $x_i$

Subset accuracy

- $\frac{1}{m} \sum_{i=1}^{m} I(P_i = Y_i)$ where $I(\text{true})=1$ and $I(\text{false})=0$

Hamming loss

- $\frac{1}{m} \sum_{i=1}^{m} \frac{|P_i \triangle Y_i|}{q}$ where $\triangle$ is the XOR operation
- Average binary classification error

Accuracy

- $\frac{1}{m} \sum_{i=1}^{m} \frac{|P_i \cap Y_i|}{|P_i \cup Y_i|}$
Example–Based Binary (2/2)

Information retrieval view

- Precision
  \[ \frac{1}{m} \sum_{i=1}^{m} \frac{|P_i \cap Y_i|}{|P_i|} \]

- Recall
  \[ \frac{1}{m} \sum_{i=1}^{m} \frac{|P_i \cap Y_i|}{|Y_i|} \]

- Harmonic mean
  \[ \frac{1}{m} \sum_{i=1}^{m} \frac{2|P_i \cap Y_i|}{|P_i| + |Y_i|} \]

Problems, when prediction set or relevant set is empty
Example-Based Ranking (1/2)

One-error

\[
\frac{1}{m} \sum_{i=1}^{m} I(\arg \min_{\lambda \in Y} r_i(\lambda) \notin Y_i)
\]

- Evaluates how many times the top-ranked label is not in the set of relevant labels of the example

Coverage

\[
\frac{1}{m} \sum_{i=1}^{m} \max_{\lambda \in Y_i} r_i(\lambda) - 1
\]

- Evaluates how many steps are needed, on average, to go down the label list to cover all relevant labels of the example
Example-Based Ranking (2/2)

Ranking loss

\[ \frac{1}{m} \sum_{i=1}^{m} \frac{1}{|Y_i||\overline{Y}_i|} |\{(\lambda_a, \lambda_b) : r_i(\lambda_a) > r_i(\lambda_b), (\lambda_a, \lambda_b) \in Y_i \times \overline{Y}_i\}| \]

- Evaluates the average fraction of label pairs that are mis–ordered for the instance

Average precision

\[ \frac{1}{m} \sum_{i=1}^{m} \frac{1}{|Y_i|} \sum_{\lambda \in Y_i} \frac{|\{\lambda' \in Y_i : r_i(\lambda') \leq r_i(\lambda)\}|}{r_i(\lambda)} \]

- Evaluates the average fraction of labels ranked above a relevant label which are also relevant labels
Label-Based Binary

Let $B(TP_j, FP_j, TN_j, FN_j)$

- be a binary evaluation measure calculated based on the above contingency table
- E.g. accuracy $= (TP_j + TN_j)/(TP_j + FP_j + TN_j + FN_j)$

Macro-averaging

- Ordinary averaging of a binary measure
- $B_{macro} = \frac{1}{q} \sum_{j=1}^{q} B(TP_j, FP_j, TN_j, FN_j)$

Micro-averaging

- Labels as different instances of the same global label
- $B_{micro} = B\left(\sum_{j=1}^{q} TP_j, \sum_{j=1}^{q} FP_j, \sum_{j=1}^{q} TN_j, \sum_{j=1}^{q} FN_j\right)$
Probabilities or Scores per Label

Bipartition can be obtained via thresholding

- Example/label–based binary measures

A ranking can be obtained after solving ties

- Example–based ranking measures

A vertical ranking can be computed

- Ranking measures calculated vertically (label–based)
- Typically: Mean average precision (macro–averaged)

Threshold independent label–based evaluation

- Area under a ROC or PR curve
Basic Approaches (used in data stream settings)

Problem transformation

- Binary relevance
- Methods that combine labels

Algorithm adaptation

- Multi-label decision trees
Binary Relevance

How it works

- Learns one binary classifier for each label
- Outputs the union of their predictions
- Can do ranking if classifier outputs scores

Complexity

- $O(qm)$

Limitation

- Does not consider label relationships
Label Powerset (1/4)

Each different set of labels in a multi-label training set becomes a different class in a new single-label classification task.

Given a new instance, the single-label classifier of LP outputs the most probable class (a set of labels).

<table>
<thead>
<tr>
<th>features</th>
<th>Label set</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>${\lambda_1, \lambda_4}$</td>
<td>1001</td>
</tr>
<tr>
<td>$x_2$</td>
<td>${\lambda_3, \lambda_4}$</td>
<td>0011</td>
</tr>
<tr>
<td>$x_3$</td>
<td>${\lambda_1}$</td>
<td>1000</td>
</tr>
<tr>
<td>$x_4$</td>
<td>${\lambda_2, \lambda_3, \lambda_4}$</td>
<td>0111</td>
</tr>
</tbody>
</table>
Label Powerset (2/4)

The number of classes influences LP's complexity

- It depends on the number of distinct labelsets that exist in the training set
- It is upper bounded by $\min(m, 2q)$
- It is usually much smaller, but still larger than $q$

LP cannot predict unseen labelsets

Limited training examples for many classes
### Label Powerset (3/4)

<table>
<thead>
<tr>
<th>Dataset</th>
<th>m</th>
<th>q</th>
<th>Bound</th>
<th>Actual</th>
<th>Diversity</th>
</tr>
</thead>
<tbody>
<tr>
<td>emotions</td>
<td>593</td>
<td>6</td>
<td>64</td>
<td>27</td>
<td>0.42</td>
</tr>
<tr>
<td>enron</td>
<td>1702</td>
<td>53</td>
<td>1702</td>
<td>753</td>
<td>0.44</td>
</tr>
<tr>
<td>hifind</td>
<td>32971</td>
<td>632</td>
<td>32971</td>
<td>32734</td>
<td>0.99</td>
</tr>
<tr>
<td>mediamill</td>
<td>43907</td>
<td>101</td>
<td>43907</td>
<td>6555</td>
<td>0.15</td>
</tr>
<tr>
<td>medical</td>
<td>978</td>
<td>45</td>
<td>978</td>
<td>94</td>
<td>0.10</td>
</tr>
<tr>
<td>scene</td>
<td>2407</td>
<td>6</td>
<td>64</td>
<td>15</td>
<td>0.23</td>
</tr>
<tr>
<td>tmc2007</td>
<td>28596</td>
<td>22</td>
<td>28596</td>
<td>1341</td>
<td>0.05</td>
</tr>
<tr>
<td>yeast</td>
<td>2417</td>
<td>14</td>
<td>2417</td>
<td>198</td>
<td>0.08</td>
</tr>
</tbody>
</table>
Label Powerset (4/4)

![Bar chart showing the number of label sets for different number of appearances.]

- Number of label sets vs. Number of appearances
  - 1: 4104
  - 2: 895
  - 3: 384
  - 4: 215
  - 5: 156
  - 6: 111
  - 7: 85
  - 8: 77
  - 9: 48
  - 10: 39

(c) Kriegel, Ntoutsi, Spiliopoulou, Tsoumakas, Zimek – Athens, Sept. 2011
Pruned Sets (Read et al., 2008)

Follows the transformation of Label Powerset, but also…

Prunes examples whose labelsets (classes) occur less times than a small user–defined threshold $p$ (e.g. 2 or 3)

- Deals with the large number of infrequent classes

Re–introduces pruned examples along with subsets of their labelsets that do exist more times than $p$

- Strategy A: Rank subsets by size/number of examples and keep the top $b$ of those
- Strategy B: Keep all subsets of size greater than $b$
Pruned Sets (Read et al., 2008)

$p=3$

<table>
<thead>
<tr>
<th>Labelset</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_1$</td>
<td>16</td>
</tr>
<tr>
<td>$\lambda_2$</td>
<td>14</td>
</tr>
<tr>
<td>$\lambda_2, \lambda_3$</td>
<td>12</td>
</tr>
<tr>
<td>$\lambda_1, \lambda_4$</td>
<td>8</td>
</tr>
<tr>
<td>$\lambda_3, \lambda_4$</td>
<td>7</td>
</tr>
<tr>
<td>$\lambda_1, \lambda_2, \lambda_3$</td>
<td>2</td>
</tr>
</tbody>
</table>

Strategy A, $b=2$

<table>
<thead>
<tr>
<th>Subsets</th>
<th>Size</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_2, \lambda_3$</td>
<td>2</td>
<td>12</td>
</tr>
<tr>
<td>$\lambda_1$</td>
<td>1</td>
<td>16</td>
</tr>
<tr>
<td>$\lambda_2$</td>
<td>1</td>
<td>14</td>
</tr>
</tbody>
</table>

Strategy B, $b=1$

<table>
<thead>
<tr>
<th>Subsets</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lambda_2, \lambda_3$</td>
<td>2</td>
</tr>
<tr>
<td>$\lambda_1$</td>
<td>1</td>
</tr>
<tr>
<td>$\lambda_2$</td>
<td>1</td>
</tr>
</tbody>
</table>
Multi–Label Decision Trees

Multi–Label C4.5 (Clare & King, 2001)

Classical entropy: \( \text{entropy}(D) = - \sum_{j=1}^{q} p(\lambda_j) \log_2 p(\lambda_j) \)

Multi–label entropy:

\[
\text{entropy}(D) = - \sum_{j=1}^{q} p(\lambda_j) \log_2 p(\lambda_j) + (1-p(\lambda_j)) \log_2 (1-p(\lambda_j))
\]

Predictive Clustering Trees (Vens et al., 2008)

Selects the best attribute based on reduction of variance

\[
\text{variance}(D) = \sum_{i=1}^{\left|D\right|} \frac{d(Y_i, \bar{Y})}{\left|D\right|}
\]
Resources for Learning from Multi–Label Data

Survey


The Mulan open source software (and datasets)

- [http://mulan.sourceforge.net](http://mulan.sourceforge.net)
Dynamic Multi-Label Data

Wikipedia (and other collaborative systems)
- Content of articles can be updated
- Categories are added to and deleted from documents
- Articles are added and deleted
- There is no fixed set of categories

Health
- The health-related habits (smoking, exercise, …) of persons change over time
- Over time a person may be suffering from zero, one or more diseases at the same time
- New diseases appear, outbreak of existing ones
Streaming Multi-Label Data

Applications

- Emails, news articles, Web 2.0 content

Multi-label learning challenges remain

Stream learning challenges (as in single-label data)

- Incremental learning under time and memory constraints
- Dealing with concept-drift
  - Is it the same in the case of multi-label data?

Issues

- Synthetic data construction
- Open-ended label space
Concept Drift

Single-label data

- Class priors might change over time
- \( p(X|c) \) might change over time

Multi-label data

- Label priors might change over time
  - But not necessarily all at the same time
- \( p(X|\lambda) \) might change over time
- Labelset priors (label correlations) may change over time
- \( p(X|Y) \) might change over time
Synthetic Multi–Label Data Streams

Why not use real–world multi–label data streams?

- Because there aren't any available
  - May require access to operational systems and their tweaking
  - May involve sensitive data (e–mail, medical records)
- Exception: Reuters Corpus Vol. 1 Ver. 2 (Lewis et al., 2004)
- What about concept–drift?
  - May require domain experts to tell us when concepts change in the data

Issues to consider

- Ensure prior probabilities of labels and dependencies between labels follow that of real–world static data
- Simulate concept–drift
Real World Multi-Label Data: Label Priors (1/2)

- **CAL500**
  - (Turnbull et al., 2008)
  - 374 labels (26)
  - 502 examples

- **Mediamill**
  - (Snoek et al., 2006)
  - 101 labels (4)
  - 43907 examples
Real World Multi-Label Data: Label Priors (2/2)

- **Yeast**
  - (Elisseeff & Weston, 2002)
  - 14 labels (4)
  - 2417 examples

- **Enron**
  - UC Berkeley Enron Email Analysis Project
  - 54 labels (3)
  - 1702 examples
The Approach of Qu et al. (2009)

Concept drift

- One moving hyperplane for each label
- This means that all labels drift simultaneously
  - A different speed of drift could be set though

Label priors

- Each label given 0.5 prior probability
- The average number of labels per example is $q/2$

Label dependencies

- Manual introduction of adhoc dependencies between just 4 labels via equations connecting hyperplanes' parameters
The Approach of Read et al. (2010)

Label priors

- A parameter, $z$, controls the average labels per example
- Label priors randomly assigned and scaled based on $z$ (this is different from what observed in real-world data)

Label dependencies

- A $q \times q$ probability matrix is filled with $p(\lambda_j | \lambda_k)$, $0 < j < k \leq q$, approximately equal to $p(\lambda_j)$ (unconditional independence)
- Some of these values are overridden with random probabilities (obeying laws of probability) to simulate unconditional dependencies
- Bayes rule is used to fill in the rest of the matrix
The Approach of Read et al. (2010)

Basic idea: features affect labels (and combinations)

- Top $q/2$ most probable subsets $S_i$ of labels are considered
  - This favors the Pruned Sets approach and is not realistic
- Attribute mapping $ζ[a] → S_a \mod q/2$

Process

- A label is selected based on prior probabilities
- More labels are added based on the dependencies matrix
- One positive and one negative example is generated from a single-label binary generator used as a parameter
- For each feature, if the labelset it "affects" is a subset of the required labelset, then the value of the positive example is used, otherwise the value of the negative one
The Approach of Read et al. (2010)

Example of the process

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>${l_1}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>${l_2}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>${l_3}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>${l_2, l_3}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>${l_1}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Concept drift

- Mixing different multi-label streams through a sigmoid, as for single-label data streams (Bifet et al., 2009)

- All labels that will drift, will drift simultaneously (!)

- The drift is expected to be very radical as two totally different streams are mixed together
Learning from Multi-Label Data Streams

Online category ranking (Crammer & Singer, 2003)
Dynamic Classifier Ensemble (Qu et al., 2009)
Multi-label Hoeffding Trees (Read et al., 2009)
Bayesian Online Learning (Zhang et al., 2010)
Multiple Windows (Spyromitros-Xioufis et al., 2011)
Online Category Ranking (Crammer & Singer, 2003)

The Multiclass Multilabel Perceptron (MMP) algorithm

- Maintains one linear perceptron per label
- When the produced ranking is wrong (according to a loss function like RankingLoss, IsError), weights get updated
- For positive (negative) labels, the magnitude of change depends on the number of negative (positive) labels ranked higher (lower) than them
- The change magnitude further depends on the actual loss

Things to point out

- Improved predictive performance compared to BR
- Linear with respect to $q$ (update complexity, memory)
- Requires online thresholding mechanism for bipartitions
Online Category Ranking (Crammer & Singer, 2003)

Additional points

- If a new label appears, a new perceptron can be initialized
- No explicit concept-drift handling
- Could serve as a strong baseline

Some related approaches

- Pairwise Multilabel Perceptron (Loza Mencía & Fürnkranz, 2008) trains one perceptron for each pair of labels
  - Improved predictions over MMP, quadratic complexity with $q$
- Calibrated Label Ranking (Fürnkranz et al., 2008) further trains one perceptron for each label as in BR
  - Improved predictions, outputs bipartitions, quadratic complexity with $q$ remains
Dynamic Classifier Ensemble (Qu et al., 2009)

Training

- Partition the stream into sequential chunks $S_1, S_2, \ldots, S_n$ (the most recent) with the same chunk size
- Train a multi-label classifier from each chunk
- Keep the last $n$ chunks/models

Given a new instance

- Find $k$ nearest neighbors in the latest chunk $S_n$
- Compute accuracy of $n$ models in the nearest neighbors
- Use this accuracy as weight in a weighted voting process
Dynamic Integration of Classifiers (Tsymbal et al., 2008)

Training

- Partition the stream into sequential chunks $S_1, S_2, \ldots, S_n$ (the most recent) with the same chunk size
- Train a single-label classifier from each chunk
- Keep the best $n$ chunks/models based on their accuracy in the latest chunk $S_n$

Given a new instance

- Find $k$ nearest neighbors in the latest chunk $S_n$
- Compute accuracy of $n$ models in the nearest neighbors
- Use this accuracy as weight in a weighted voting process
Dynamic Classifier Ensemble (Qu et al., 2009)

Combines

- The ensemble based approach of (Tsymbal et al., 2008) for handling concept drift (more or less)
- An improved version of binary relevance proposed in that paper (Qu et al, 2009)

Message

- State-of-the-art ensemble methods that maintain models trained over different time periods in order to deal with concept drift (Tsymbal et al, 2008; Wang et al., 2003) are readily applicable to multi-label data streams, and could serve as strong baselines
  - But, again, assumes that all labels drift simultaneously
  - Can handle the appearance of new labels
Multi-Label Hoeffding Trees (Read et al., 2010)

The multi-label Hoeffding tree combines

- The Hoeffding tree of (Domingos & Hulten, 2000)
- The multi-label decision tree of (Clare & King, 2001)

Furthermore, Read et al. (2010) consider training a multi-label learner at the leaves of the tree

- In particular, they argue for the Pruned Sets algorithm
- Combinations are calculated based on the first $N$ examples, during which period only single labels are considered
Multi-Label Hoeffding Trees (Read et al., 2010)

In addition, Read et al. (2010) combine the tree with the ADWIN bagging single-label ensemble approach (Bifet et al., 2009)

- Online bagging with model replacement based on accuracy upon concept-drift detection
- Again, simultaneously all labels are considered

Incremental Thresholding

- As some of the used methods, especially ensembles, output a probability vector for the labels, thresholding was required to obtain a bipartition
- The approach followed, incrementally updated (increased/decreased) a threshold, based on the predicted bipartition and feedback on the actual one
Multi-Label Hoeffding Trees: Experiments – Methods

HT: Multi-label Hoeffding tree with majority labelset at the leaves

HT–PS: Multi-label Hoeffding tree with PS classifier at the leaves ($p=1$, $b=1$, Naive Bayes), label combinations determined by first 1000 examples

HT–PSA: Multi-label Hoeffding tree with PS classifier at the leaves ($p=1$, $b=1$, Naive Bayes), ADWIN monitoring number of label combinations and resetting PS on change

BBR: Bagging of ten BR classifiers with Hoeffding tree as base classifier

BAG HT–PSA: ADWIN bagging of ten HT–PSA models
Multi-Label Hoeffding Trees: Experiments – Datasets

Random Tree Generator (Domingos & Hulten, 2000)

- Nominal attributes with 5 values, tree depth of 5, leaves starting at level 3 and a 0.15 chance of leaves thereafter

Random Radial Basis Function (RBF) generator

- Normally distributed hypersphere of examples surrounding 50 centers with varying densities
- Drift is introduced by moving the centroids with constant speed initialized by a drift parameter

Two types of multi-label stream per generator

- Simple: 10 attributes, $z=1.5$, $q=8$
- Complex: 100 attributes, $z=5$, $q=30$
Multi-Label Hoeffding Trees: Results

Macro F1

<table>
<thead>
<tr>
<th></th>
<th>HT</th>
<th>HT-PS</th>
<th>BRR</th>
<th>HT-PSA</th>
<th>BAG HT-PSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>RTS</td>
<td>0.30 ± 0.02</td>
<td>0.37 ± 0.01</td>
<td>0.17 ± 0.01</td>
<td>0.32 ± 0.01</td>
<td><strong>0.59 ± 0.08</strong></td>
</tr>
<tr>
<td>RRBFS</td>
<td>0.25 ± 0.00</td>
<td>0.35 ± 0.00</td>
<td>0.32 ± 0.01</td>
<td>0.34 ± 0.02</td>
<td><strong>0.46 ± 0.02</strong></td>
</tr>
<tr>
<td>RTC</td>
<td>0.07 ± 0.03</td>
<td>0.05 ± 0.00</td>
<td>DNF</td>
<td>0.11 ± 0.02</td>
<td><strong>0.13 ± 0.00</strong></td>
</tr>
<tr>
<td>RRBFC</td>
<td>0.05 ± 0.02</td>
<td><strong>0.15 ± 0.02</strong></td>
<td>DNF</td>
<td>0.12 ± 0.02</td>
<td><strong>0.12 ± 0.00</strong></td>
</tr>
<tr>
<td>RRBFS 0.0001</td>
<td>0.36 ± 0.04</td>
<td>0.40 ± 0.03</td>
<td>0.16 ± 0.01</td>
<td>0.55 ± 0.04</td>
<td><strong>0.69 ± 0.04</strong></td>
</tr>
<tr>
<td>RRBFS 0.001</td>
<td>0.36 ± 0.03</td>
<td>0.43 ± 0.04</td>
<td>0.15 ± 0.01</td>
<td>0.49 ± 0.03</td>
<td><strong>0.70 ± 0.05</strong></td>
</tr>
<tr>
<td>RRBFC 0.0001</td>
<td>0.06 ± 0.02</td>
<td>0.09 ± 0.01</td>
<td>DNF</td>
<td>0.12 ± 0.01</td>
<td><strong>0.20 ± 0.00</strong></td>
</tr>
<tr>
<td>RRBFC 0.001</td>
<td>0.06 ± 0.03</td>
<td>0.09 ± 0.02</td>
<td>DNF</td>
<td>0.12 ± 0.03</td>
<td><strong>0.13 ± 0.00</strong></td>
</tr>
<tr>
<td>IMDB</td>
<td>0.06</td>
<td>0.08</td>
<td>DNF</td>
<td>0.08</td>
<td>0.07</td>
</tr>
<tr>
<td>MMILL</td>
<td>0.03</td>
<td>0.07</td>
<td>0.03</td>
<td>0.03</td>
<td>0.06</td>
</tr>
<tr>
<td>TMC2007</td>
<td>0.08</td>
<td><strong>0.38</strong></td>
<td>0.25</td>
<td>0.16</td>
<td>0.17</td>
</tr>
</tbody>
</table>
Multiple Windows (Spyromitros–Xioufis et al., 2011)

Maintains two fixed-size separate moving windows for each label, one for positive and one for negative examples

A binary classifier ($k$NN) is trained on the union of the positive and the negative window

Correlations between labels are ignored but multiple concept drift is explicitly treated by retaining only the most recent examples for each label

Class imbalance is tackled by controlling the ratio of positive and negative examples

Essentially, a Binary Relevance approach
  - But BR methods that exploit correlations can be used
Multiple Windows (Spyromitros–Xioufis et al., 2011)

Training

- No training is involved since \( k \)NN is used as base classifier
- The positive and negative windows of each label are updated
- A space efficient implementation is described

Prediction

- NN search is applied to the union of positive and negative examples of each label
- A time efficient implementation is described
- A confidence score is given for each label
- Bipartition is taken by applying an incremental thresholding method (described next)
Multiple Windows (Spyromitros–Xioufis et al., 2011)

An incremental version of the PCut (proportional cut) thresholding method:

Every $n$ instances:

- A threshold that would most accurately approximate the observed frequency of that label in the last $n$ instances is calculated

The calculated thresholds are used on the next batch of $n$ instances

This type of thresholding allows adaptation to changing label priors

Experimental results show substantial gains in terms of Macro $F_1$ when this type of thresholding is applied
Summary and Conclusions

Multi-label data

- Ubiquitous complex type of data
- Requires techniques that exploit label structure and scale to large number of labels

Multi-label data streams, further requires

- Techniques for incremental learning under time/memory constraints and for dealing with concept-drift

Issues of interest, being currently partly addressed

- Synthetic data generation
- Adaptation to multi-label concept-drift
- Class imbalance, thresholding
References (Block 2) (1/4)

A. Bifet, G. Holmes, B. Pfahringer, R. Kirkby, R. Gavalda, “New ensemble methods for evolving data streams”, Proc. 15th ACM SIGKDD Int. Conf. on Knowledge discovery and data mining (KDD '09), 2009.


References (Block 2) (2/4)


References (Block 2) (3/4)

J. Read, A. Bifet, G. Holmes, B. Pfahringer, “Efficient Multi-label Classification for Evolving Data Streams”, Working Paper, 04/2010, University of Waikato


References (Block 2) (4/4)


H. Wang, W. Fan, P.S. Yu, J. Han, “Mining Concept-Drifting Data Streams using Ensemble Classifiers”, Proc. 2003 ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'03)

End of Block 2

Thank you!

Questions?
Presentation Outline

☑ Block 1: Introduction
☑ Block 2: Multi-label data
☑ Block 3: High-dimensional data

- static data
  - the “curse of dimensionality”
  - general strategies
  - exemplary approaches

- dynamic data

☑ Block 4: Multi-relational data
☑ Block 5: Conclusions and Outlook

(arthur zimek)

(c) kriegel, ntoutsi, spiliopoulou, tsoumakas, zimek – athens, sept. 2011
The “curse of dimensionality”

one buzzword for many problems –

First aspect: *Optimization Problem*

 “[The] curse of dimensionality […] is a malediction that has plagued the scientists from earliest days.” (Bellman 1961)

- The difficulty of any global optimization approach increases exponentially with an increasing number of variables (dimensions).

- General relation to clustering: fitting of functions (each function explaining one cluster) becomes more difficult with more degrees of freedom.

- Direct relation to subspace clustering: number of possible subspaces increases dramatically with increasing number of dimensions.
The “curse of dimensionality”

Second aspect: Concentration effect of $L_p$-norms

In (Beyer et al. 1999, Hinneburg et al. 2000) it is reported that the ratio of $(\text{Dmax}_d - \text{Dmin}_d)$ to $\text{Dmin}_d$ converges to zero with increasing dimensionality $d$

- $\text{Dmin}_d = \text{distance to the nearest neighbor in } d \text{ dimensions}$
- $\text{Dmax}_d = \text{distance to the farthest neighbor in } d \text{ dimensions}$

Formally:

$$\forall \varepsilon > 0 : \lim_{d \to \infty} P\left[\text{dist}_d \left( \frac{\text{Dmax}_d - \text{Dmin}_d}{\text{Dmin}_d}, 0 \right) \leq \varepsilon \right] = 1$$

- Distances to near and to far neighbors become more and more similar with increasing data dimensionality (loss of relative contrast or concentration effect of distances).
- This holds true for a wide range of data distributions and distance functions, but…
The “curse of dimensionality”

From bottom to top: minimum observed value, average minus standard deviation, average value, average plus standard deviation, maximum observed value, and maximum possible value of the Euclidean norm of a random vector. The expectation grows, but the variance remains constant. A small subinterval of the domain of the norm is reached in practice. [Figure and caption: (Francois et al. 2007)]

- The observations stated in (Beyer et al. 1999, Hinneburg et al. 2000) are valid within clusters but not between different clusters as long as the clusters are well separated (Bennett et al. 1999, Francois et al. 2007, Houle et al. 2010).

- This is not the main problem for subspace clustering, although it should be kept in mind for range queries.
The “curse of dimensionality”

Third aspect: Relevant and Irrelevant attributes

- A subset of the features may be relevant for clustering
- Groups of similar (“dense”) points may be identified when considering these features only

Different subsets of attributes may be relevant for different clusters
- Separation of clusters relates to relevant attributes (helpful to discern between clusters) as opposed to irrelevant attributes (indistinguishable distribution of attribute values for different clusters).
The “curse of dimensionality”

Effect on clustering:

- Usually the distance functions used give equal weight to all dimensions
- However, not all dimensions are of equal importance
- Adding irrelevant dimensions ruins any clustering based on a distance function that equally weights all dimensions
The "curse of dimensionality"
The “curse of dimensionality”

Fourth aspect: *Correlation among attributes*

- A subset of features may be correlated
- Groups of similar (“dense”) points may be identified when considering this correlation of features only

- Different correlations of attributes may be relevant for different clusters
The “curse of dimensionality”

Other strange things happen: Shrinking volume of hyperspheres

\[ V_d (r) = \frac{\pi^{\frac{d}{2}} r^d}{\Gamma\left(\frac{d}{2} + 1\right)} \]

small change of \( \varepsilon \) in a range query
\[ \Rightarrow \]
enormous change in data space coverage
The “curse of dimensionality”

Why not feature selection?

- (Unsupervised) feature selection is global (e.g. PCA)
- We face a local feature relevance/correlation: some features (or combinations of them) may be relevant for one cluster, but may be irrelevant for a second one
The "curse of dimensionality"

- Use feature selection before clustering

Projection on first principal component

PCA

DBSCAN
The “curse of dimensionality”

- Cluster first and then apply PCA

DBSCAN

Projection on first principal component

PCA of the cluster points

(c) Kriegel, Ntoutsi, Spiliopoulou, Tsoumakas, Zimek – Athens, Sept. 2011
General Strategies

adapt distance computations

- full-dimensional secondary distance measures (e.g. SNN)
- adaptations to subspaces

select subspaces

- random selection
- directed:
  - top-down search
  - bottom-up search
  - constrained search
General Strategies

Secondary distance measures: a derived distance measure on top of some primary distance

popular example: shared nearest neighbor (SNN) information:

- assess the set of \( s \) nearest neighbors for two objects \( x \) and \( y \) in terms of some primary distance (Euclidean, Manhattan, cosine...)

- derive overlap of neighbors (common objects in the NN of \( x \) and \( y \))

\[
\text{SNN}_s(x, y) = |\text{NN}_s(x) \cap \text{NN}_s(y)|
\]

- similarity measure:

\[
\text{simcos}_s(x, y) = \frac{\text{SNN}_s(x, y)}{s}
\]

- cosine of the angle between membership vectors for NN(x) and NN(y)
General Strategies

distance measures based on SNN:

\[ d_{\text{invs}}(x, y) = 1 - \text{simcos}_s(x, y) \]

\[ d_{\text{acos}}(x, y) = \arccos(\text{simcos}_s(x, y)) \]

\[ d_{\text{ln}}(x, y) = -\ln(\text{simcos}_s(x, y)) \]

- \( d_{\text{invs}} \): linear inversion
- \( d_{\text{acos}} \): penalizes slightly suboptimal similarities more strongly
- \( d_{\text{ln}} \): more tolerant for relatively high similarity values but approaches infinity for very low similarity values
General Strategies

discussion and performance evaluation of SNN (Houle et al. 2010)

160 / 320 relevant dimensions, primary distance: Euclidean
General Strategies

adaptations to subspaces (cluster-based)

- typical: partitioning clustering (k-means-type)
- eigensystem of point-sets (provisional clusters)

example: ORCLUS (Aggarwal&Yu 2000)
General Strategies

adaptations to subspaces (instance-based)

• eigensystem of a point $p$ based on its $\varepsilon$-neighborhood in Euclidean space
• in eigenvalue matrix $E_p$, replace large eigenvalues by 1, small eigenvalues by $\kappa \gg 1$
• adapted eigenvalue matrix yields a correlation similarity matrix for point $p$: $V_p E'_p V_p^T$

• effect on distance measure:

• distance of $p$ and $q$ w.r.t. $p$: $\sqrt{(p-q) \cdot V_p \cdot E'_p \cdot V_p^T \cdot (p-q)^T}$
• distance of $p$ and $q$ w.r.t. $q$: $\sqrt{(q-p) \cdot V_q \cdot E'_q \cdot V_q^T \cdot (q-p)^T}$

(c) Kriegel, Ntoutsi, Spiliopoulou, Tsoumakas, Zimek – Athens, Sept. 2011
General Strategies

adaptations to subspaces (instance-based)

- symmetry of the distance measure by choosing the maximum:

\[
\max \left\{ \sqrt{(p-q) \cdot V_p \cdot E'_p \cdot V_p^T \cdot (p-q)^T}, \sqrt{(q-p) \cdot V_q \cdot E'_q \cdot V_q^T \cdot (q-p)^T} \right\} \leq \epsilon
\]

example: 4C (Böhm et al. 2004), similar: PreDeCon (Böhm et al. 2004a)
General Strategies

selection of subspaces: restrict similarity–search/distance–computations to some subset of attributes

- random selection: perform analysis in several randomly chosen subspaces, possibly combine analysis–results

- directed:
  - top–down search for the most appropriate subspace (“projected clustering”)
  - bottom–up search (usually Apriori–like): test combinations of 1–, 2–, 3–, etc. –dimensional subspaces (finding all clusters in all subspaces – “subspace clustering”)
  - constrained (MultiView, Orthogonal) : given a clustering in some subspace, look for clusters in different (or even orthogonal) subspaces (e.g. Bickel&Scheffer (2004), Günnemann et al. (2009), Cui et al. (2010), see also the Tutorial of Müller et al., 2010, 2011)
General Strategies

top–down search

Cluster–based approach:
  - Learn the subspace of a cluster in the entire $d$-dimensional feature space
  - Start with full–dimensional clusters
  - Iteratively refine the cluster memberships of points and the subspaces of the cluster

Instance–based approach:
  - Learn for each point its subspace preference in the entire $d$-dimensional feature space
  - The subspace preference specifies the subspace in which each point “clusters best”
  - Merge points having similar subspace preferences to generate the clusters
General Strategies

key problem for top–down search: *how to learn the subspace preference of a cluster or a point?*

Most approaches rely on the so–called “locality assumption”

- The subspace is usually learned from the local neighborhood of cluster representatives/cluster members in the entire feature space:
  - Cluster–based approach: the *local neighborhood* of each cluster representative is evaluated in the \( d \)-dimensional space to learn the “correct” subspace of the cluster
  - Instance–based approach: the *local neighborhood* of each point is evaluated in the \( d \)-dimensional space to learn the “correct” subspace preference of each point

- Learning the subspace: usually variance/covariance–analysis (e.g., PCA)

- *The locality assumption*: the subspace preference can be learned from the *local neighborhood* in the \( d \)-dimensional space

(elaborated in Kriegel et al. 2009)
**General Strategies**

**bottom-up search:**

If the cluster criterion implements the downward closure, one can use any bottom-up frequent itemset mining algorithm, to find subspaces where clusters are present.

- Start with 1-dim subspaces and merge them to compute higher dimensional ones
  - If the cluster-criterion holds for any $k$-dimensional subspace $S$, then it also holds for any $(k-1)$-dimensional projection of $S$
  - Use the reverse implication for pruning:
    - If the cluster-criterion does not hold for a $(k-1)$-dimensional projection of $S$, then the criterion also does not hold for $S$

- Apply any frequent itemset mining algorithm
  (usually Apriori – Agrawal& Srikant, 1994)
General Strategies

Downward-closure property:

If $C$ is a dense set of points in subspace $S$, then $C$ is also a dense set of points in any subspace $T \subset S$.

$p$ and $q$ density-connected in $\{A,B\}$, $\{A\}$ and $\{B\}$

$p$ and $q$ not density-connected in $\{B\}$ and $\{A,B\}$

MinPts = 4
General Strategies

Downward-closure property:

the reverse implication does not hold necessarily

pruning-criterion:
no cluster in subspace A

no cluster in any superspace of A
General Strategies

Search-space:

4-itemset \{A,B,C,D\}  4D subspace (1,1,1,1)

(c) Kriegel, Ntoutsi, Spiliopoulou, Tsoumakas, Zimek – Athens, Sept. 2011
General Strategies

The key limitation: *global density thresholds*

- Usually, the cluster criterion relies on density
- In order to ensure the downward closure property, the density threshold must be fixed
- Consequence: the points in a 20–dimensional subspace cluster must be as dense as in a 2–dimensional cluster
- This is a rather optimistic assumption since the data space grows exponentially with increasing dimensionality (drastic change of volume in $\varepsilon$–ranges)
- Consequences:
  - A strict threshold will most likely produce only lower dimensional clusters
  - A loose threshold will most likely produce higher dimensional clusters but also a huge amount of (potentially meaningless) low dimensional clusters (elaborated in Kriegel et al. 2009)
Exemplary Approaches

• adapt distance computations
  • ABOD (Kriegel et al. 2008) (angle–variance instead of pure distances)
  • SOD (Kriegel et al. 2009a) (SNN–based neighborhood, compute subspace properties)

• select subspaces
  • random: PINN (Vries et al. 2010) (random projections to apply LOF (Breunig et al. 2000) on high–dim data, provable margin of error)
  • directed:
    • top–down:
      • PreDeCon (Böhm et al. 2004a)
      • PROCLUS (Aggarwal et al. 1999)
    • bottom–up:
      • CLIQUE (Agrawal 1998)
      • SUBCLU (Kailing et al. 2004)
Exemplary Approaches

top–down subspace search: PROCLUS (Aggarwal et al. 1999)

- *k*-medoid cluster model: a cluster is represented by its medoid
  - Iterative refinement of randomly determined initial cluster medoids
    - Each point is assigned to the nearest medoid (where the distance to each medoid is based on the corresponding subspaces of the medoids)
    - To each cluster, a subspace (of relevant attributes) is assigned based on the cluster members
  - After convergence:
    Points that have a large distance to their nearest medoid are classified as noise
- Integration of a proper distance function into *k*-medoid clustering
- Cluster–based locality assumption
Exemplary Approaches

bottom-up subspace search: CLIQUE (Agrawal 1998)

- Cluster model
  - Each dimension is partitioned into $\xi$ equi-sized intervals called units
  - A $k$-dimensional unit is the intersection of $k$ 1-dimensional units (from different dimensions)
  - A unit $u$ is considered dense if the fraction of all data points in $u$ exceeds the threshold $\tau$ (Apriori-style mining of possibly dense higher dimensional units)
  - A cluster is a maximal set of connected dense units

(c) Kriegel, Ntoutsi, Spiliopoulou, Tsoumakas, Zimek – Athens, Sept. 2011
Exemplary Approaches

bottom–up subspace search: SUBCLU (Kailing et al. 2004)

- Cluster model:
  - Density–based cluster model of DBSCAN (Ester et al. 1996)
  - Clusters are maximal sets of density–connected points
  - Density connectivity is defined based on core points
  - Core points have at least $MinPts$ points in their $\varepsilon$–neighborhood

  - Detects clusters of arbitrary size and shape (in the corresponding subspaces)
  - Apriori–based search in all possibly interesting subspaces

- Downward–closure property holds for sets of density–connected points
End of Block 3–static data

Thank you!

Questions?
Presentation Outline

☑ Block 1: Introduction
☑ Block 2: Multi-label data

Block 3: High-dimensional data
  - static data
  - dynamic data
    - full-dimensional stream clustering
    - challenges due to high-dimensionality
    - subspace stream clustering

Block 4: Multi-relational data

Block 5: Conclusions and Outlook
Dynamic data/ data streams

- Data evolve over time as new data arrive and old data become obsolete.

- Static algorithms are not appropriate for such kind of data
  - (Huge amounts of) Data is accumulated over time
  - The generative distribution might change over time
  - One-pass access to the data
  - Arrival at a rapid rate

- We distinguish between
  - Dynamic data arriving at a low rate
  - Possible infinite sequence of elements arriving at a rapid rate (data streams)
Challenges & requirements for stream clustering: full dimensional case

- Traditional clustering methods require access upon the whole dataset
  - Online maintenance of clustering

- The underlying population distribution might change: drifts/shifts of concepts
  - One clustering model might not be adequate to capture the evolution

- The role of outliers and clusters are often exchanged in a stream
  - Clear and fast identification of outliers
A taxonomy of approaches (& representative methods)

<table>
<thead>
<tr>
<th>Static clustering</th>
<th>Dynamic/Stream clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Partitioning methods</strong></td>
<td></td>
</tr>
<tr>
<td>• k–Means</td>
<td>• single pass k–Means</td>
</tr>
<tr>
<td>• k–Medoids</td>
<td>• STREAM k–Means</td>
</tr>
<tr>
<td><strong>Density–based methods</strong></td>
<td>• CluStream</td>
</tr>
<tr>
<td>• DBSCAN</td>
<td></td>
</tr>
<tr>
<td>• OPTICS</td>
<td>• DenStream</td>
</tr>
<tr>
<td><strong>Grid–based methods</strong></td>
<td>• incDBSCAN *</td>
</tr>
<tr>
<td>• STING</td>
<td>• incOPTICS *</td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• DStream</td>
</tr>
</tbody>
</table>

(*) These methods require access to the raw data (this access might be limited though)
Partitioning methods

- Goal: Construct a partition of a set of objects into k clusters
  - e.g. k-Means, k-Medoids

- Two types of methods:
  - Adaptive methods:
    - Single pass k-Means (Farnstrom et al, 2000)
    - STREAM k-Means (O’Callaghan et al, 2002)
  - Online summarization – offline clustering methods:
    - CluStream (Aggarwal et al, 2003)
Single pass k–Means (Farnstrom et al, 2000)

- An extension of k–Means for streams
  - The iterative process of static k–Means cannot be applied to streams
  - Use a buffer that fits in memory and apply k–Means locally in the buffer

- Stream is processed in batches $B_1, B_2$, ..., each batch fits in memory
  - Apply k–Means on $B_1$ (retain only the k centers)
  - Whenever the buffer is full (i.e. batch $B_i$):
    - Apply k–Means over $B_i$ and the k previous centers
      - Each center is treated as a point, weighted with the number of points it compresses
    - Clear buffer, retain only the k weighted centers
CluStream (Aggarwal et al, 2003)

The stream clustering process is separated into:

- an **online** micro-cluster component, that summarizes the stream locally as new data arrive over time
  - Micro-clusters are stored in disk at snapshots in time that follow the pyramidal time frame.

- an **offline** macro-cluster component, that clusters these summaries into global clusters
  - Clustering is performed upon summaries instead of raw data
CluStream: the micro–cluster summary

The micro–cluster summary for a set of d–dimensional points \((X_1, X_2, \ldots, X_n)\) arriving at time points \(T_1, T_2, \ldots, T_n\) is defined as:

\[
\text{CFT} = (\text{CF}_2^x, \text{CF}_1^x, \text{CF}_2^t, \text{CF}_1^t, n)
\]

- Easy calculation of basic measures to characterize a cluster:
  - Center: \(\frac{\text{CF}_1^x}{n}\)
  - Radius: \(\sqrt{\frac{\text{CF}_2^x}{n} - \left(\frac{\text{CF}_1^x}{n}\right)^2}\)

- Important properties of micro–clusters:
  - Incremental: \(\text{CFT}(C_1 \cup C_p) = \text{CFT}(C_1) + p\)
  - Additivity: \(\text{CFT}(C_1 \cup C_2) = \text{CFT}(C_1) + \text{CFT}(C_2)\)
  - Subtractive: \(\text{CFT}(C_1 - C_2) = \text{CFT}(C_1) - \text{CFT}(C_2), C_1 \supseteq C_2\)
CluStream: the algorithm

A fixed number of \( q \) micro-clusters is maintained over time

- Online micro-cluster maintenance as new points arrive
  - Find the closest micro-cluster \( clu \) for the new point \( d \)
    - If \( d \) is within the max-boundary of \( clu \), \( d \) is absorbed by \( clu \)
    - Otherwise, a new cluster is created with \( d \)
  - The number of micro-clusters should not exceed \( q \)
    - Delete most obsolete micro-cluster or merge the two closest ones

- Periodic storage of micro-clusters snapshots into disk
  - At different levels of granularity depending upon their recency

- Offline macro-clustering
  - Input: A user defined time horizon \( h \) and the number of macro-clusters \( k \)
  - Locate the valid micro-clusters during \( h \)
  - Apply \( k \)-Means upon these micro-clusters \( \rightarrow k \) macro-clusters
Density based methods

Clusters as regions of high density surrounded by regions of low density (noise)

- Density is measured locally, in the $\epsilon$-neighborhood of each point
  - e.g. DBSCAN, OPTICS

Very appealing for streams

- No assumption on the number of clusters
- Discovering clusters of arbitrary shapes
- Ability to handle outliers and noise

But, they miss a clustering model (or it is too complicated)

- Clusters are represented by all their points

Solution: Describe clusters as set of summaries

- DenStream (Cao et al, 2006)
DenStream (Cao et al, 2006)

- The online–offline rationale is followed:
  - **Online** summarization as new data arrive over time
    - Core, potential core and outlier micro–clusters
  - **Offline** clustering over the summaries to derive the final clusters
    - A modified version of DBSCAN over the summaries

- Data are subject to ageing according to the exponential ageing function (damped window model)
  - \( f(t) = 2^{-\lambda t}, \lambda > 0 \)
  - \( \lambda \) the decay rate
  - higher \( \lambda \), less important the history data
DenStream: summarizing the stream

The micro-cluster summary at time $t$ for a set of $d$-dimensional points $(p_1, p_2, \ldots, p_n)$ arriving at time points $T_1, T_2, \ldots, T_n$ is defined as:

$$MC = (CF^1, CF^2, w)$$

- Center: $c = \frac{CF^1}{w}$
- Radius: $r = \sqrt{\frac{CF^2}{w} - \left(\frac{CF^1}{w}\right)^2}$

- A micro-cluster summary $c_p$ can be maintained incrementally
  - If a new point $p$ is added to $c_p$: $c_p = (CF^2+p^2, CF^1+p, w+1)$
  - If no point is added to $c_p$ for time interval $\delta t$:
    - $c_p = (2^{-\lambda \delta t}CF^2, 2^{-\lambda \delta t}CF^1, 2^{-\lambda \delta t}w)$
DenStream: core, potential core & outlier summaries

- **Core (or dense) micro-clusters**
  - \((w \geq \mu) \& (r \leq \varepsilon)\)
  
  \(\varepsilon: \text{the radius threshold}\)
  \(\mu: \text{the weight threshold}\)

- **But, in an evolving stream, the role of clusters and outliers often interchange:**
  - Should provide opportunity for the gradual growth of new clusters
  - Should promptly get rid of the outliers

- **Potential core micro-clusters**
  - \((w \geq \beta^{*}\mu) \& (r \leq \varepsilon), 0 < \beta \leq 1\)

- **Outlier micro-clusters**
  - \((w < \beta^{*}\mu) \& (r \leq \varepsilon), 0 < \beta \leq 1\)
DenStream: the algorithm (online step)

A group of $p$–micro–clusters and $o$–micro–clusters is maintained over time

- When a new point $d$ arrives
  - Find its closest $p$–micro–cluster $pclu$
    - If the updated radius of $pclu \leq \epsilon$, merge $d$ to $pclu$
  - o.w. find its closest $o$–micro–cluster $oclu$
    - If the updated radius of $oclu \leq \epsilon$, merge $d$ to $oclu$
    - Check if $oclu$ can be upgraded to a $p$–micro–cluster (if now $w \geq \beta \mu$)
  - o.w., create a new $o$–micro–cluster with $d$

- Periodic $p$– and $o$–micro–clusters update due to ageing
  - Delete a $p$–micro–cluster when $w < \beta \mu$
  - Delete an $o$–micro–cluster when $w < \xi$ (expected weight based on its creation time)
    - The longer an $o$–micro–cluster exists, the higher its weight is expected to be
  - The number of $p$– and $o$–micro–clusters is bounded
DenStream: the algorithm (offline step)

- Upon request, apply a variant of DBSCAN over the set of online maintained p–micro–clusters

- Core–micro–cluster
- Directly density reachable
  - $c_p$ is directly density reachable if:
    1. $c_q$ is a c–micro–cluster
    2. $\text{dist}(c_p, c_q) \leq 2\varepsilon$ (i.e., they are tangent or intersecting)

- Density reachable
- Density connected
Grid based methods

- A grid structure is used to capture the density of the dataset.
  - A cluster is a set of connected dense cells
  - e.g. STING

- Appealing features
  - No assumption on the number of clusters
  - Discovering clusters of arbitrary shapes
  - Ability to handle outliers

- In case of streams
  - The grid cells “constitute” the summary structure
  - Update the grid structure as the stream proceeds
  - DStream (Chen & Tu, 2007)
Dstream (Chen & Tu, 2007)

- The online–offline rationale is followed:
  - **Online** mapping of the new data into the grid
  - **Offline** computation of grid density and clustering of dense cells

Data are subject to ageing (damped window model)
  - \( D(x, t) = \lambda^{t-T(x)} \), \( T(x) \) is arrival time of point \( x \), \( t \) the current time point
  - \( 0 < \lambda < 1 \), the decay factor

The density of a cell \( g \) at time \( t \):

\[
D(g, t) = \sum_{x \in E(g, t)} D(x, t)
\]
DStream: Summarizing the stream into the grid

- The characteristic vector of a grid cell \( g \) is defined as:
  \[
  (t_g, t_m, D, \text{label}, \text{status})
  \]

- The grid density can be updated incrementally:
  \[
  D(g, t_n) = \lambda^{t_n-t_l} D(g, t_l) + 1
  \]
  \( t_n \): the new record arrival time
  \( t_l \): the last record arrival time \((t_n > t_l)\)

- The density of a cell changes over time
  - Dense grid cells: \( D(g, t) \geq D_m \)
  - Transitional grid cells: \( D_l \leq D(g, t) < D_m \)
  - Sparse grid cells: \( D(g, t) < D_l \)
Clustering over high dimensional dynamic data

- Challenges due to the dynamic nature of data
  - See previous discussion on full dimensional dynamic data

- Challenges due to high dimensionality
  - See Arthur’s part on high dimensional static data

⇒ Both point & dimensions associated with a cluster might evolve over time
## A taxonomy of approaches (& representative methods)

<table>
<thead>
<tr>
<th>Static clustering</th>
<th>Dynamic/Stream clustering</th>
<th>Subspace clustering</th>
<th>Subspace stream clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Partitioning methods</strong></td>
<td>• k–Means</td>
<td>• 1pass k–Means</td>
<td>• PROCLUS</td>
</tr>
<tr>
<td></td>
<td>• k–Medoids</td>
<td>• STREAM k–Means</td>
<td>• HPStream</td>
</tr>
<tr>
<td><strong>Density-based methods</strong></td>
<td>• DBSCAN</td>
<td>• DenStream</td>
<td>• PreDeCon</td>
</tr>
<tr>
<td></td>
<td>• OPTICS</td>
<td>• incDBSCAN</td>
<td>• inPreDeCon</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• incOPTICS</td>
<td></td>
</tr>
<tr>
<td><strong>Grid-based methods</strong></td>
<td>• STING</td>
<td>• DStream</td>
<td>• CLIQUE</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>• DucStream</td>
</tr>
</tbody>
</table>

(*) These methods require access to the raw data (this access might be limited though)
HPStream (Aggarwal et al, 2005)

- A partitioning method that maintains $k$ subspace clusters over time
- Solution: Continuous refinement of the set of projected dimensions and data points as the stream progresses
  - Extension of PROCLUS (Aggarwal et al, 1999) to streams
    - PROCLUS is not applicable since it requires multiple passes over the data
- Data are subject to ageing according to the exponential ageing function (damped window model)
  - $f(t) = 2^{-\lambda t}$, $\lambda > 0$ (higher $\lambda$, less important the history data)
HPStream: the fading cluster structure

A fading cluster structure at time $t$ for a set of $d$-dimensional points $C=\{p_1, p_2, \ldots, p_n\}$ arriving at $T_1, T_2, \ldots, T_n$ is defined as:

$$FC(C, t) = (CF^1(C, t), CF^2(C, t), W(t))$$

- It can be maintained online:
  - Additivity: $FC(C_1 \cup C_2, t) = FC(C_1, t) + FC(C_2, t)$
  - Temporal multiplicity: $FC(C, t+\delta t) = 2^{-\lambda \delta t} \ast FC(C, t)$

- Moreover, the cluster relevant dimensions are modeled:
  - Cluster bit vector: $B(C) = <v_1, v_2, \ldots, v_d>$
    - $v_i = 1$ if the $i^{th}$ dimension is *relevant* to $C$, o.w., $v_i = 0$
HPStream: Projected subspace and projected distance

- **Projected subspaces**
  - For each cluster C, compute the spread along each dimension j
    
    $$ r_j = \sqrt{FC^2(C, t)_j \cdot W(t) - \left(\frac{FC^1(C, t)_j}{W(t)}\right)^2} $$
  
    - Out of the total k*d values, choose the k*l dimensions with the least radii
    - Update the entries in B(C) accordingly for each cluster C

- **Projected distance**
  - Manhattan segmental distance between point x and the centroid of C:
    
    $$ d(C, x) = \sum_{j \in B(C): v_j = 1} |C_j - x_j| $$
    
    $$ \frac{1}{|B(C): v_j = 1|} $$

(c) Kriegel, Ntoutsi, Spiliopoulou, Tsoumakas, Zimek – Athens, Sept. 2011
HPStream: the algorithm

A fixed number of $k$ projected clusters is maintained over time

- **Initialization step**
  - Full dimensional $k$-Means over the initial points $\rightarrow k$ clusters, compute their projected subspaces
  - Repeat until convergence: Re-assign points to clusters based on projected distance. Re-compute the projected subspaces.

- **Online maintenance of clusters as a new point $X$ arrives**
  - Temporarily add $X$ to each cluster $C$ and update its projected subspace $B(C)$
  - Find the closest cluster $clu$ to $X$
    - If $X$ is within the boundary of $clu$, $X$ is absorbed by $clu$
    - o.w., a new cluster is created with $X$
  - If the total number of clusters $> k$, delete the less recent cluster

The boundary is defined by the avg radius along the projected dimensions in the cluster scaled by a factor $\tau$
incPreDeCon (Kriegel et al, 2011)

- A density-based method for maintaining density based subspace clusters over time
- Idea: Update only the affected, due to new points, part of the current clustering
  - Extension of PreDeCon (Boehm et al, 2004a) to dynamic data
- Incremental algorithm
  - Requires some access to raw data but this access is limited to a subset of the data
  - Can be applied to low arrival rate streams
  - Produces exact clustering results.
  - Supports single and batch insertions
PreDeCon (Boehm et al, 2004): basic notions

- Dimension preference vector for a point
  \[ \bar{w}_p = (w_1, w_2, \ldots, w_d) \]
  \[ w_i = \begin{cases} 
  1 & \text{if } \text{VAR}_i > \delta \\
  \kappa & \text{if } \text{VAR}_i \leq \delta 
  \end{cases} \]
  \[ \text{VAR}_{A_i}(N_\varepsilon(p)) = \frac{\sum_{q \in N_\varepsilon(p)} (\text{dist}((\pi A_i(p), \pi A_i(q)))^2}{|N_\varepsilon(p)|} \]

- Preference weighted distance function
  \[ \text{dist}_p(p, q) = \sqrt{\sum_{i=1}^{d} \frac{1}{w_i} \cdot (\pi A_i(p) - \pi A_i(q))^2} \]
  \[ \text{dist}_{\text{pref}}(p, q) = \max\{\text{dist}_p(p, q), \text{dist}_q(q, p)\} \]

- Preference weighted core points
  \[ \text{CORE}_{\text{pref}}(p) \iff \text{PDIM}(N_\varepsilon(p)) \leq \lambda \land |N_\varepsilon^{\bar{w}_o}(p)| \geq \mu \]
incPreDeCon: Affected core points

- Observation: A subspace preference cluster can be revealed through any of its preference weighted core points.

- Affected core points

\[ \mathcal{N}_\varepsilon(q) \xrightarrow{\text{VAR}_{A_i}(\mathcal{N}_\varepsilon(q))} \bar{w}_q \xrightarrow{\text{PDIM}(\mathcal{N}_\varepsilon(q))} \mathcal{N}_\varepsilon^{\bar{w}}(q) \]

\[ \text{CORE}_{\text{pref}}^\text{den}(p) \Leftrightarrow \text{PDIM}(\mathcal{N}_\varepsilon(p)) \leq \lambda \wedge |\mathcal{N}_\varepsilon^{\bar{w}_p}(p)| \geq \mu \]

- Effect in core property
  - non-core \(\rightarrow\) core
  - core \(\rightarrow\) non-core
  - core \(\rightarrow\) core, under different preferences

(c) Kriegel, Ntoutsi, Spiliopoulou, Tsoumakas, Zimek – Athens, Sept. 2011
incPreDeCon: Affected points & reorganization

- Due to the change in the core property of q, points that are preference weighted reachable from q might be affected:
  - q: core $\rightarrow$ non-core: any density connectivity relying on q is destroyed
  - q: non-core $\rightarrow$ core: new density connectivity might arise
  - q: core $\rightarrow$ core (different dimension preferences): both different cases can occur

- From where to start restructuring?
  - Any changes are initiated by points in AFFECTEDCORE
  - If q is an affected core point, seeds points are any core point q’ in its preferred neighborhood (UPDSEED).
  - Apply the expand procedure of PreDeCon using the points in the UPDSEED.
DUCstream (Gao et al, 2005)

- A grid based method for detecting evolving clusters
  - Extension of CLIQUE (Agrawal et al, 1998) to streams

- Basic idea: Monitor the density of units over time and update the clusters

- Data are processed in batches $B_1, B_2, ..., B_t, ...$, $|B_i| = m$

- Density of unit
  - $\text{rel\_den}(u, t) = \frac{\text{den}(u, t)}{m \times t}$
    - If $\text{rel\_den}(u, t) \geq \gamma$, $u$ is dense at $t$

- Local density of a unit
  - $\text{loc\_den}(u, t) = \frac{\text{den}(u, t)}{m \times (t - i - 1)}$
    - If $\text{loc\_den}(u, t) \geq \gamma$, $u$ is candidate for dense and we should monitor it
DUCstream: the algorithm

The algorithm maintains a list L of local dense units over time

- Initialization: Apply CLIQUE on B₁
- When a new chunk Bₜ arrives at t
  - map_and_maintain(L, Bₜ)
    - Map Bₜ → grid
    - For each affected unit u: If u is already in L, update it; otherwise, if it is a local dense unit start monitoring it (add it in L)
    - Decide on Qₐ, Qₜ
  - update_clustering(Rₜ₋₁, Qₐ, Qₜ)
    - Incremental update of the previous clustering Rₜ₋₁ based on the addition of new dense units (Qₐ) and the deletion of old dense units (Qₜ)
      - due to Qₐ: creation, absorption, mergence
      - due to Qₜ: removal, reduction, split

Qₐ: the added dense units
Qₜ: the deleted dense units
Block 3: High-dimensional data: lessons learned

“curse of dimensionality” – a couple different problems in high-dimensional data. Strategies:

• adapt similarity-measures (e.g. SNN or subspace similarity)

• select subspaces (e.g. bottom-up, top-down)

additional challenges in dynamic data:

• updates may change structures learned so far

• stream data only available for a single pass: adapt/select subspace with one glimpse at the data

methods esp. for dynamic data just begun to be explored – limitations:

• a constant number of clusters over time and a constant number of projected dimensions per cluster (HPStream)

• requirement for (limited) access to raw data (incPreDeCon)
End of Block 3—dynamic data

Thank you!

Questions?
References (Block 3) (1/7)

C. C. Aggarwal, J. Han, J. Wang, P. S. Yu: A framework for clustering evolving data streams. VLDB, 2003.

C. C. Aggarwal, J. Han, J. Wang, P. S. Yu: A framework for projected clustering of high dimensional data streams. VLDB, 2004.


References (Block 3) (2/7)


F. Cao, M. Ester, W. Qian, A. Zhou: Density–Based Clustering over an Evolving Data Stream with Noise. SDM, 2006.


References (Block 3) (4/7)


References (Block 3) (5/7)


References (Block 3) (6/7)


Presentation Outline

☑ Block 1: Introduction
☑ Block 2: Multi-label data
☑ Block 3: High-dimensional data

Block 4: Multi-relational data
  - Social data in evolving networks
  - An evolving warehouse

Block 5: Conclusions and Outlook

(Myra Spiliopoulou)
Multi-relational data in a social network

Actors in a social network: People

Objects in a social network:
- Pictures
- Documents
- Videos
- Tags
- Comments, ...

Actions in a social network:
- uploading, sharing
- inviting
- ranking, agreeing, disagreeing
- replying, commenting
- linking, ...

Lin et al. (FacetNet: WWW’08, METAFACT: KDD’09) use the terms “facets” and “relations” (among facets)
Why monitor social (multi-relational) data?

Crisis management

- Short term tasks, e.g. first aid & support for helpers
- Long term tasks, e.g. understanding disease outbreaks, establishing emergency routines etc

Business

- Sales planning & formulation of recommendations
- Advertising and company PR (including blogs)
- Opinion monitoring

Contact to the citizen

- Polls and opinion monitoring

Detection and monitoring of trends (in market, science, politics, ...)

(c) Kriegel, Ntoutsi, Spiliopoulou, Tsoumakas, Zimek – Athens, Sept. 2011
Learning and adaptation of communities

Community Detection in Evolving Social Graphs
Classification of methods for community detection in evolving graphs

- Community detection in snapshot graphs and mapping across successive structures
  - based on Clique Percolation and community overlap \cite{Palla07}
  - based on Mutual Awareness expansion and interaction correlation \cite{Lin07}
  - based on the membership of core nodes \cite{Wang08}

- Evolutionary community detection
  - Traditional clustering techniques in an evolutionary setting \cite{Chakrabarti06}
  - Spectral clustering \cite{Tang08}
  - Non-negative matrix/tensor factorization
  - Identifying graph stream segments and community structure in them

- Incremental community detection
  - based on dynamic modularity maximization \cite{Gorke10}
  - density-based, DENGRAH \cite{Falkowski08}

- Mixed methods combining evolutionary community detection and community mapping
- Community detection on graph segment approximations and community mapping \cite{Yang09}
- Identification of temporally smooth local communities and community mapping \cite{Kim09}
- Community factorization \cite{Chi07}
- FacetNet \cite{Lin08}
- MetaFac \cite{Lin09}
- GraphScope \cite{Sun07}
- Stream-Group \cite{Duan09}

(c) Kriegel, Ntoutsi, Spiliopoulou, Tsoumakas, Zimek – Athens, Sept. 2011
Learning on a (multi-relational) evolving social network

1. Two aspects of model quality (Chakrabarti et al., KDD’06):
   - **Snapshot cost** \( CS \)
     - captures quality of current clustering
   - **Temporal cost** \( CT \)
     - captures similarity to previous clustering

2. Model learning as optimization problem for

\[
\text{Cost}(\xi) = a \times CS(\xi) + \beta \times CT(\xi)
\]

- Find a sequence of models that minimizes global cost (Chakrabarti et al., KDD’06)
- Build a model that minimizes cost wrt previous model (Chi et al., KDD’07)
Learning on a (multi-relational) evolving social network

1. Two aspects of model quality (Chakrabarti et al., KDD’06):
   - **Snapshot cost**: $CS(\xi)$ captures quality of current clustering
   - **Temporal cost**: $CT(\xi)$ captures similarity to previous clustering

2. Model learning as optimization problem for

   $Cost(\xi) = a \times CS(\xi) + \beta \times CT(\xi)$

   - Find a sequence of models that minimizes global cost (Chakrabarti et al., KDD’06)
   - Build a model that minimizes cost wrt previous model (Chi et al., KDD’07)
Learning a model of smoothly evolving multifaceted communities

FacetNet (Lin et al., WWW’08)
- Captures multiple types of entities in matrices
- Expresses snapshot cost and temporal cost as KL–divergence
- Matrix factorization

METAFAc (Lin et al., KDD’09)
- Introduces “facet”, “relation”, “multi–relational hypergraph”
- Captures multiple relations among entities with multiple tensors
- Tensor factorization

ContexTour (Lin et al., SDM’10)
- Captures multiple relations among entities with matrices
- Snapshot cost as approximation error (of model that explains data)
- Temporal cost as approximation error (of model that explains transition of old clustering to new one)

\[ D(Y;X) = \sum_{ij} Y_{ij} \log \frac{Y_{ij}}{X_{ij}} - Y_{ij} + X_{ij} \]
Evolving communities: mission accomplished?

Multi-relational social data can be modeled as a tensor (or multiple tensors).

Tensor factorization maps the entities and their relations in a latent space. A mixture model is learned.

✔ It explains the data at each timepoint.
✔ It enforces smooth transitions from one timepoint to the next.
✔ The approach scales.

BUT

✔ Works for the a posteriori study of all entities.
   Problems when unknown entities arrive.

❖ How to “read” the results and “see” the trends?
Showing evolution in a social network

Approach I: Summary–based

- Learn and adapt a model of the social network
- Summarize the data of each community into (a set of) variable(s)
- Show how the summaries are evolving

(Mei & Zhai, KDD’05): Evolution of the “strength” of the topics associated with the Asia tsunami of Dec. 2004

Figure 7: Absolute strength life cycles in CNN data
Dynamic topic modeling with PLSA

- A “topic” is a latent variable, it is associated with all words in the vocabulary but the strength of a word in a topic varies.
- A “topic” is described by the strongest words, after removing the non-discriminative ones.
- A human-understandable text (a “theme”) can be derived.
- An indicator of a theme’s importance is its strength at each timepoint – the number of documents adhering to it.

EXAMPLE: Asia tsunami of 2004
- personal experience
- aids
- statistics
- ...

EXAMPLE: Hurricanes Katrina & Rita
- government response to Katrina
- aid and donation for Katrina
- government response to Rita
- storm Rita
Showing topic evolution on texts (Mei & Zhai, KDD’05)

Topics on the *evolution graph*

- One set of nodes per timepoint
- A node is a *theme* (is derived from a topic)
- An edge connects a theme of timepoint $i$ to the best matching theme at timepoint $i+1$.

The inspection of the evolution graph gives clues on theme evolution.

*Figure 6: Theme evolution graph for Asia Tsunami*
(I) Summary–based visualization of evolution graphs for communities

CODYM (Falkowski et al., Web Intelligence‘06)
- Original graph: nodes are users; edges denote interactions
- Evolution graph: nodes are clusters; links denote similarity

TIMEFALL (Ferlez et al., ICDE‘08)
- Original graph is bipartite: nodes are individuals and resources; edges denote events associating an individual to a resource
- Evolution graph: nodes are profiles; links denote similarity

TopicTable (Gohr et al., KDIR’10)
- Original graph: nodes are users, tags & documents; edges denote postings in a social tagging platform
- Evolution threads: one thread per topic
(I) Summary-based: evolution graphs for communities

**TopicTable** (Gohr et al., KDIR’10)

- Social tagging environment: stream of user postings, partitioned in time intervals
- Topics-under-a-tag as latent variables: Probabilistic model learning and adaptation (adaptive PLSA)

Visualization of a topic as a current:

- the topic descriptor consists of the words most dominant in the topic; new words are in boldface
- the river’s width reflects the similarity of the descriptor at t to t+1
- strength of the topic as radius of the circle in the mid of the river
Showing evolution in a social network

Approach II: Object-based

- Learn a model of the social network at each timepoint and adapt it from one timepoint to the next
- Depict objects that are important
- Show how the surroundings of the important objects change

(Ahmed & Xing, PNAS 2009): Evolving network of 100 US senators, analyzed with the method TESLA
Visualizing evolving communities – ContexTour (Lin et al, SDM’10)

At each timepoint, draw one view per context (e.g. community view, content view)

Select important entities by ranking on
- importance score: conditional probability of entity given cluster
- total importance: importance score over all clusters

then perform biased sampling

Compute similarity of entities, link entities that are more similar than a threshold, and display the resulting entity network

Superimpose a *contour-map* over the displayed entity network; it depicts the density distribution of *all* entities
Visualizing evolving communities – ContexTour (Lin et al, SDM’10)

At each timepoint, draw one view per context: sample important entities, link them, draw the network and place it in the contour–map of all entities

*Contour–Map*:

- Kernel functions over all sampled entities: estimate joint density distribution of all kernels
- Assign each non-sampled entity to one of the closest sampled entities: reverse–kNN query

(c) Kriegel, Ntoutsi, Spiliopoulou, Tsoumakas, Zimek – Athens, Sept. 2011
Visualizing evolving communities – ContexTour (Lin et al, SDM’10)

Across timepoints, draw one view per context, thereby preserving the mental map of the observer.

1. At timepoint $t$, minimize:

$$\sum_{i<j} \frac{1}{s_{ij}} \left( \|X_i - X_j\| - \frac{1}{s_{ij}} \right)^2 + \sum_i (X_i - X'_i)^2 \quad \text{Eq.12}$$

2. Assign clusterID and color to each entity, depending on the entity’s soft clustering scores.

- Compute transition probabilities between clusters of $t'$, $t$
- Map $c$ (of time $t$) to $c'$ (of earlier time $t'$) if the id of $c'$ has not been assigned and $c$ delivers the maximum transition probability for $c'$. If there is no $c'$, then create new clusterId for $c$.
Visualizing evolving communities – ContexTour (Lin et al, SDM’10)

Across timepoints, draw one view per context:

Preserving the location of important objects from one timepoint to the next.
Showing evolution: Summary and an open issue

Summary–based methods: work best
when data can be summarized to topics

Methods depicting important objects: work best
when objects have names and
there are not many new objects at each timepoint

What to show?

- A video
- A series of pictures – how many?
- One picture – which objects should be in it?
First part of Block 4 is over ...

Questions thus far?
Presentation Outline

☑ Block 1: Introduction
☑ Block 2: Multi-label data
☑ Block 3: High-dimensional data

Block 4: Complex multi-relational data
  ▪ Evolving communities
  ▪ Evolving customers

Block 5: Conclusions and Outlook
Customers are changing ... 

Insights won during the Netflix competition:

Y. Koren (KDD’09 & CACM’10), Abstract:
“Customer preferences for products are drifting over time. Product perception and popularity are constantly changing as new selection emerges. Similarly, customer inclinations are evolving, leading them to even redefine their taste.

[...] Within the ecosystem intersecting multiple products and customers, many different characteristics are shifting simultaneously, while many of them influence each other and often those shifts are delicate and associated with a few data instances. This distinguishes the problem from concept shift explorations, where mostly a single concept is tracked.”
Does model updating pay off on *live data*?

Dias et al. (RecSys’08):
Effect of model learning and of model change on the *impact* of a recommendation engine

- Study on consumer goods (standardized, repeated consumption)
- Three recommenders:

  *In–Store Recommender I (hand–crafted):*
  up to 8 non–personalized recommendations; items from the same categories as in the basket

  *In–Store Recommender II (model–based):*
  replaces 2 out of the 8 recommendations with personalized ones

  *Checkout Recommender:*
  recommends 3 items that are usually purchased, but are not in the current basket; chooses items with max probability given basket
Dias et al. (RecSys’08) – Impact of the recommenders on:

1. **Shopper penetration**

   proportion of shoppers who accepted at least one recommendation over all shoppers who bought at least one item from LeShop counting up to and including the month in consideration.

Findings:

- Penetration increases with time
- Penetration is affected positively by model updating

“The increase in penetration resulting from a model update was on average 0.26%.”

Figure 4: Penetration of our recommender systems over time. The red dotted vertical lines indicate the time points at which the model files were updated.
Dias et al. (RecSys’08) – Impact of the recommenders on:

2. Direct extra revenue

- total amount of money spent by shoppers on recommended items
- computed on a monthly basis
- normalized by LeShop’s total monthly turnover

Findings:

✓ The 3 recommenders achieve a D.E.R. of ca. 0.30% – a large value.
→ Model updating is imperative.

“An important lesson to take away from this analysis is that it is imperative to keep the model files updated [...]. We see that the performance of the model-based systems fall-off very rapidly until they are updated again using the latest data.”
Learning customer evolution

Approach I: Learning on timestamped data

Approach II: Learning on a drifting data stream
Koren (KDD’09): Automating the detection of change

Time-changing baseline predictors:

✧ Capturing the evolution of items’ popularity
✧ Capturing the evolution of users’ rating style

\[ b_{ui}(t) = \mu + b_u(t) + b_i(t) \]

Time-changing factor model:

✧ Capturing the evolution of users

Temporal dynamics at neighborhood models
Static learning in latent factor space (Koren, KDD’09)

Users and items are mapped to a joint latent factor space of dimensionality $f$:

- A user $u$ is associated with a vector $p_u$ in $\mathbb{R}^f$.
- An item $i$ is associated with a vector $q_i$ in $\mathbb{R}^f$.
- A rating could be predicted by the rule $\hat{r}_{ui} = q_i^T p_u$, but biases must be taken into account.

Vectors $p_u$, $q_i$ are learned by minimizing the regularized square error

$$
\min_{q^*, p^*} \sum_{(u,i,t) \in K} (r_{ui} - q_i^T p_u)^2 + \lambda(\|q_i\|^2 + \|p_u\|^2)
$$

where $\lambda$ controls the extend of regularization.
Static learning in latent factor space (Koren, KDD’09)

Users and items are mapped to a joint latent factor space of dimensionality $f$:

- a user $u$ is associated with a vector $p_u$ in $\mathbb{R}^f$
- an item $i$ is associated with a vector $q_i$ in $\mathbb{R}^f$
- a rating is predicted by the rule

$$\hat{r}_{ui} = b_{ui} + q_i^T p_u + \frac{1}{|R(u)|} \sum_{j \in R(u)} y_j$$

where the baseline predictor

$$b_{ui} = \mu + b_u + b_i$$

captures the deviation of ratings of $u$ and for $i$ from average $\mu$, $R(u)$ are the items rated by $u$ and $y_i$ (for $i$ in $R(u)$) is a second set of factor vectors.
Time–changing baseline predictors (Koren, KDD’09)

A time–changing baseline predictor has the form

\[ b_{ui}(t) = \mu + b_u(t) + b_i(t) \]

The time–changing baseline predictor for items is

\[ b_i(t) = b_i + b_i,Bin(t) \]

for which the timeline is partitioned in bins.

The time–changing baseline predictor for users must capture drift in a user’s ranking behavior, as well as sudden shifts and periodic phenomena.

\[ b_u(t) = b_u + \text{???)} \]
Time-changing baseline predictors (Koren, KDD’09)

A time-changing baseline predictor has the form

\[ b_{ui}(t) = \mu + b_u(t) + b_i(t) \]

Time-changing baseline predictors for users include:

- \( b_u^{(1)}(t) = b_u + a_u \text{sign}(t - t_u) |t - t_u|^{\beta} \)

- \( b_u^{(2)}(t) \): extends the above by partitioning the time interval of u’s ratings in \( k_u \) bins and learning \( k_u \) kernels.

- \( b_u^{(3)}(t) = b_u + a_u \text{sign}(t - t_u) |t - t_u|^{\beta} + b_{u,t} \)

- \( b_u^{(4)}(t) \): extends \( b_u^{(2)}(t) \) with \( b_{u,t} \)
Time–changing baseline predictors (Koren, KDD’09)

Periodicity with respect to day of week may affect both the ratings for an item and the rating behavior of a user:

\[ b_i(t) = b_i + b_{i,\text{Bin}(t)} \oplus b_{i,\text{period}(t)} \]

\[ b_u^{(3)}(t) = b_u + a_u \text{sign}(t - t_u) \left| t - t_u \right|^\beta + b_{u,t} \oplus b_{u,\text{period}(t)} \]

The merit of an item \( i \) at time \( t \) is not truly user–independent, because different users apply different rating scales and tend to change them over time, too. Hence:

\[ b_{ui}(t) = \mu + b_u(t) + b_i(t) \times c_u(t) \]
Learning with a time-changing factor model

Time-aware prediction rule:

\[ \hat{r}_{ui}(t) = b_{ui}(t) + q_i^T \left( p_u(t) + \left| R(u) \right|^{-\frac{1}{2}} \sum_{j \in R(u)} y_j \right) \]

\[ p_u(t) = \begin{cases} p_{u1} + a_{u1} \text{dev}_{u}(t) + p_{u1,t} \\ \vdots \\ p_{uk} + a_{uk} \text{dev}_{u}(t) + p_{uk,t} \\ \vdots \\ p_{uf} + a_{uf} \text{dev}_{u}(t) + p_{uf,t} \end{cases} \]
Solution goes beyond the Netflix competition

“Matrix factorization models map both users and items to a joint latent factor space of dimensionality $f$, such that ratings are modeled as inner products in that space.” (Koren, KDD’09)

✔ Extends to tensor factorization $\iff$ captures multiple entities.

✔ Extends to multi-tensors $\iff$ captures multiple relations (users rate items, reply to tweets, assign tags).

✔ Covers stationary portion, linear change and periodicity.

BUT

☐ Parameters must be set.

☢ All the data have been used.

Problems when unknown entities arrive.

☢ How to adapt to incoming data?
Learning customer evolution

Approach I: Learning on timestamped data

Approach II: Learning on a drifting data stream
Classification $\rightarrow$ Stream Classification $\rightarrow$ Concept Drift

A simple definition of the conventional classification problem:

- $D$ is a set of training examples of the form $(x, y)$ from a population $A$
- $x$ is a vector of attribute values over the feature space $F$
- $y$ is a discrete class label from a set of labels $C$

**Objective:** Induce a model $\xi_{A\rightarrow C}$ that predicts the class label $y$ for future examples $x$ (with high quality).
Classification → Stream Classification → Concept Drift

An adapted definition for the stream classification problem:

- $D$ is a growing set of training examples of the form $(x, y)$ from a (most likely evolving) population $A$
- $x$ is a vector of attribute values over the feature space $F$
- $y$ is a discrete class label from a set of labels $C$

**Objective:** Induce a model $\xi_{A \rightarrow C}$ on the basis of the examples seen thus far, use it to predict labels for new examples, and then incorporate the new examples in the model, as soon as their labels arrive.
Classification $\rightarrow$ Stream Classification $\rightarrow$ **Concept Drift**

Some (more or less) conventional manifestations of drift:

- Change in the speed of the whole stream, without change in the data distribution
- Change in the speed of part of the stream – of the examples adhering to label $y$ in $C$.
- Arrival of examples whose labels are not in $C$ – novel classes
- Change in the relationship between a variable in feature space $F$ and the target variable.
Customer evolution as a stream mining problem


(Siddiqui et al., SSDBM’09)
Customer evolution as a stream classification problem

Simplest scenario: Set of classes $C=\{\text{single, two, 3+}\}$

Running Example (Siddiqui et al., SSDBM’09)

Fig 2: Schema of the multi-table stream.
Classification $\rightarrow$ Stream Classification $\rightarrow$ Concept Drift

Drift on multi-relational data:

- Examples re-appear – with a different label than before.
- There are relations among entities: a change in one entity may affect the label of another.
- There are relations among entities: a change in a relation may affect the label of an entity.

$\rightarrow$ Do not overwrite entities.

$\rightarrow$ Allow for the appearance of new entities & the re-appearance of old ones.

$\rightarrow$ Remember changes in label.

$\rightarrow$ Model learning when entities change their content and label.
Preparing the data for multifaceted stream classification

(Siddiqui et al., SSDBM’09)

Expand the Customer stream:
- Add (feature,value)-pairs to accommodate the content of each associated entity and relation.
- Summarize numerical values into four features (avg,max,min,count)
Preparing the data for multifaceted stream classification

Running Example (Siddiqui et al., SSDBM’09)

<table>
<thead>
<tr>
<th>Transaction</th>
<th>Product</th>
<th>Customer</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>TID</td>
<td>CID</td>
<td>PID</td>
<td>Time</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Target:
- single
- two
- 3+
- ?
- two

(186)
Tree Induction for Perennial Objects (Siddiqui et al, SSDBM’10)

Core classifier is based on VFDT_c (Gama et al., KDD’03):

- Tree induction algorithm for conventional streams that
- extends VFDT (Domingos & Hulten, KDD 2000), which
- at each moment, grows the best tree, provided that its gain is better than that of the second best wrt the Hoeffding bound,
- deals with concept drift by maintaining alternative trees,
- keeps “sufficient statistics” (wrt splitting criterion) in each leaf,
- uses kNN classifier at each leaf to assign an example to a class.

Assume \( n \) independent observations of a variable and let \( \overline{r} \) be the observed mean. If \( R \) is the range of the variable and \( r \) is the true mean, then with probability \( 1 - \delta \) it holds that

\[
\overline{r} \in [r - \varepsilon, r + \varepsilon], \varepsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2n}}
\]
Tree Induction for Perennial Objects (Siddiqui et al, SSDBM’10)

Alternative trees for fast adaptation to drift:

- Alternative trees are grown as in the core classifier.
- A window slides over the propositionalized data: *individuals* are forgotten, if they have not been seen for a while.
- The lifetime of an alternative tree is determined by
  * its quality
  * its support (number of individuals it describes)

Alternative trees re-juvenate if they get new support.
Customer evolution as a stream mining task – easy?

Evolving multi-relational data can be expressed as a stream.

✔ Tree induction builds upon conventional stream classification.

✔ Relational stream mining methods available also for clustering (Siddiqui et al), association rules & novelty detection (Appice, Malerba et al)

✔ Methods for forgetting individuals and for incorporating new examples

✔ Adaptation to drift

BUT

⚠ The propositionalization of multi-relational data scales poorly.
Evolution of customers and other multi-relational data

Learning one model on timestamped data:
- excellent results, but requires all the data;
  re-learning needed when new data arrive

Learning and adapting the model at each timepoint:

I. Methods based on matrix/tensor factorization,
   learn a new model at each timepoint, enforcing a smooth
   evolution of the old model
   + mathematically solid methods
   + scale well to the stream of instances of each relation
   - problems with new entities

II. Multi-relational stream mining methods
   + allow for entities and relations to constitute streams
   + cope with new types of drift
   - propositionalization step scales poorly
Block 4 References – Evolution in social networks


Y. Chi, X. Song, D. Zhou, K. Hino, B. Tseng. “Evolutionary Spectral Clustering by Incorporating Temporal Smoothness”, ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD’07), San Jose, CA, Aug. 2007.


C. Tanipathananandh, T. Berger–Wolf, D. Kempe. A Framework for Community Identification in Dynamic Social Networks. 13th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD’07), San Jose, CA, Aug. 2007.
Block 4 References – Evolution & visualization

T. Falkowski, J. Bartelheimer, M. Spiliopoulou. Mining and visualizing the evolution of subgroups in social networks. 2006 IEEE/WIC/ACM Int. Conf. on Web Intelligence (WI’06), 52–58, Dec. 06.


Q. Mei, C. Zhai. Discovering evolutionary theme patterns from text – an exploration of temporal text mining. 11th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD'05), 198–207, Chicago, IL, Aug. 2005.

Block 4 References – Recommenders and model updating


Block 4 References – Multi-relational streams + core methods

M. Ceci, A. Appice, C. Loglisci, C. Caruso, F. Fumarola, C. Valente, D. Malerba: Relational Frequent Patterns Mining for Novelty Detection from Data Streams. MLDM 2009:427–439

P. Domingos, G. Hulten “Mining High Speed Data Streams”, ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD’2000), Washington DC, Aug. 2000.


J. Gama, R. Rocha, P. Medas “Accurate Decision Trees for Mining High Speed Data Streams”, ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD’03), Aug. 2003.


Z. F. Siddiqui, M. Spiliopoulou. “Combining multiple interrelated streams for incremental clustering”, 21st Int. Conf. on Scientific and Statistical Database Management (SSDBM’09), New Orleans, USA, June 2009.

End of Block 4

Thank you!

Questions?
Presentation Outline

☑ Block 1: Introduction
☑ Block 2: Multi-label data
☑ Block 3: High-dimensional data
☑ Block 4: Multi-relational data

Block 5: Conclusions and Outlook
Dynamic multi-label data in the junction of two research fields

**Data Streams**
- Incremental learning under time and memory constraints
- Dealing with concept-drift

**Multi-Label Data**
- Exploit label structure to improve predictive performance
- Scale up to large number of labels

Label-specific concept drift + Open-ended label space

Simulations & testing with synthetic data:
How to capture real-world data properties?
Dynamic high-dimensional data: the next steps

- Changes in concept (incl. # clusters)
- Feature space may change

Data Streams
- Incremental learning under time and memory constraints
- Change detection
- Noise handling

High-dimensional data
- Curse of dimensionality
- Efficient subspace selection
- Similarity measures

Change detection & monitoring
- Dimension evaluation
- Finding the right datasets for testing
Dynamic multi-relational data: dealing with evolving networks and people

Data Streams
- Incremental learning under time and memory constraints
- Dealing with drift

Multi-relational data
- Learning complex concepts
- Visualizing complex models

New forms of concept drift + Streams of interrelated entities

Scalable methods
- Adaptation to new forms of drift
- Synthetic data for simulations
KD for people evolution: Multi-rel, high-dim, multi-label

Some forms of people evolution are well-known, and are exploited successfully (e.g. for ads, insurances, recommendations, personalized services)

- Commonplace I:
  Teenagers behave more like teenagers.
  Married adults behave more like married adults.

- Commonplace II:
  When singles get married, they start behaving like married people.
Group evolution as a cluster tracing problem

For the evolution of complex concepts, adaptive learners are needed:

- Which groups of people (if any) have changed their attitude to nuclear energy after the Japan tsunami?
- Which groups of people have changed their attitude to vegetables at the beginning of the EHEC outbreak? Is this change now reversed?

Q1: How is each cluster of similar individuals evolving?

Q2: How to find points of change / drift in the life of each cluster?
Acknowledgements

Cordial thank you to our students and collaborators:

- Aristotle University
  Eleftherios Spyromitros-Xioufis

- U. Magdeburg
  Zaigham Faraz Siddiqui
  Alex Hinneburg & Andre Gohr (U. Halle)

Funding institutions:

- Irene Ntoutsi
  [http://www.humboldt-foundation.de](http://www.humboldt-foundation.de)

- M. Spiliopoulou and the KMD team at U. Magdeburg
  DFG, project IMPRINT (2011- ) on “Incremental Mining for Perennial Objects”
  DAAD, “Recommendation engines for the Web 2.0” cooperation with U. Thessaloniki
  EU, FP7 project UBIPOL 248010
Thank you very much!

QUESTIONS