Learning from Evolving Data

Joao Gama
University of Porto, PORTUGAL
Myra Spiliopoulou
University of Eindhoven, NETHERLANDS
Ernestina Menasalvas
Technical Univ. of Madrid, SPAIN
Athena Vakali
University of Thessaloniki, GREECE

and the Chairs of the HaCDAIS Workshop:
Mykola Pechenizkiy, Eindhoven Univ. of Technology, NETHERLANDS
Indre Zliobaite, Eindhoven Univ. of Technology, NETHERLANDS

Goal for the Tutorial

Point out the new challenges introduced by evolving data like resource aware learning, change detection, novelty detection, multi-horizons analysis, and reasoning about the learning process.

- We elaborate on important application areas where data evolution must be taken into account
- We discuss the impact of evolution on economical data, and on understanding social networks.
- We investigate how learning under constraints (time, storage capacity and other resources) is affected by data evolution;
- We identify applications that require model learning over complex data (as in Customer Relationship Management or Social Tagging)
- Present appropriate adaptive learning methods.

Tutorial Presenters (or HaCDAIS Workshop Co-chairs?)

Joao Gama
Is a researcher at LIAD, University of Porto, working at the Machine Learning group. His main research interest is in Learning from Data Streams. He has published more than 80 articles. He served as Co-chair of ECML 2005, DS09, KDD09 and a series of Workshops on KDSS and Knowledge Discovery from Sensor Data with ACM SIGKDD. He is author of a recent book on Knowledge Discovery from Data Streams. [http://www.liad.up.pt/~jgama/]

Athena Vakali
Associate Professor at the Department of Informatics of Aristotle University, Thessaloniki, Greece. Her main research interest is Web usage mining with emphasis on social Web data and she published more than 100 papers in refereed journals and Conferences. [http://owinds.csd.auth.gr/~avakali/]

Mykola Pechenizkiy
Assistant Professor at the Department of Computer Science, Eindhoven University of Technology, the Netherlands. He has broad research interests in data mining and its application to various (adaptive) information systems serving industry, commerce, medicine and education. He has been organizing several workshops and conferences in these areas. [http://www.win.tue.nl/~mpechen/]

Indre Zliobaite
Postdoctoral Researcher at the Department of Computer Science, Eindhoven University of Technology, the Netherlands. She received her PhD from Vilnius University, Lithuania. Her main research interests include data mining under concept drift and context-aware prediction. She publishes in intelligent data analysis and pattern recognition venues. [http://zliobaite.googlepages.com]

Tutorial Structure

Block 1: Introduction

- Mining and adapting classifiers
- Dealing with concept drift
- Novelty detection

Block 2: Supervised learning on streams

- Adapting clusters
- Probabilistic models
- Learning on complex data
Tutorial Structure

Block 4: Mining evolving social data
- Structures and Models
- Community Detection in Evolving Social Graphs
- Applications of Evolving Community Detection
Block 5: Mining under resource constraints
- Introduction
- Approaches
Block 6: Conclusions and Outlook

Presentation Agenda  (tentative)

Block 1: Introduction
- Block 2: Supervised learning on streams
  (João Gama, Mykola Pechenizkiy, Indre Zliobaitė)
- Block 3: Unsupervised learning on streams
  (M. Spiliopoulou)
- Tiny Break
- Block 4: Mining evolving social data (Athena Vakali)
- Block 5: Mining under resource constraints (Ernestina Menasalvas)
- Block 6: Conclusions and Outlook

Presentation Outline

Block 1: Introduction
- Block 2: Supervised learning on streams
  (João Gama)
- Mining and adapting classifiers
- Novelty detection
- Dealing with concept drift in AIS (M. Pechenizkiy and I. Žliobaitė)
Block 3: Unsupervised learning on streams
Block 4: Mining evolving social data
Block 5: Mining under resource constraints
Block 6: Conclusions and Outlook

Evolving Data: Illustrative Example

Wind Power Generation
At the end of 2009, the worldwide capacity of wind powered generators was 159.2 gigawatts (GW). In 2020 the penetration of renewable energies should be 20% (EC recommendation).

Goal:
- Given wind velocity and direction predict the power produced by a set of turbines for a time horizon

Predicting Wind Power

Numerical weather prediction

Numerical weather prediction uses current weather conditions as input into mathematical models of the atmosphere to predict the weather.

The atmosphere is a fluid. NWP uses the state of the fluid at a given time and the equations of fluid dynamics and thermodynamics to estimate the state of the fluid at some time in the future.

Usual NWP:
- Predictions for every hour in the next 24, 48, 72 hours
- Predictions are delivered every day

Prediction and observation...
Challenges from Evolving Data

It is impractical to store and use all the historical data for training:

- It would require infinite storage and running time

Requires Incremental learning

There may be concept-drift in the data, meaning, the underlying concept of the data may change over time.

Uncertainty and reliability of predictions

Novel classes may emerge from unlabelled examples in the stream.

We might be interested in:

- prediction / forecast for different time horizons
- modeling for different time granularities
- mine evolution of the decision models based on the changes observed in a sequence of windows

Stream Classification

Construct a classification model based on past records

Use the model to predict labels for new data

- Single Classifiers
- Ensemble of Classifiers
Stream Classification

Processing each example:
- Small constant time
- Fixed amount of main memory
- Single scan of the data
- Without (or reduced) revisit old records.

Processing examples at the speed they arrive
Decision Models at anytime
Ability to detect and react to concept drift
Ideally, produce a model equivalent to the one that would be obtained by a batch data mining algorithm.

The Very Fast Decision Tree

Mining High-Speed Data Streams, P. Domingos, G. Hulten; KDD00

The base idea:
A small sample can often be enough to choose the optimal splitting attribute
- Collect sufficient statistics from a small set of examples
- Estimate the merit of each attribute
- Use Hoeffding bound to guarantee that the best attribute is really the best.
- Statistical evidence that it is better than the second best

How many examples are enough?

Entropy as Splitting Criteria

How to measure the ability of an attribute to discriminate between classes?
Many measures. Entropy is a popular one: \[ H(x) = -\sum p_i \log_2 p_i \]

Growing a decision tree is guided by reducing the entropy, that is the randomness or difficulty to predict the class.

Hoeffding bound

Suppose we have made \( n \) independent observations of a random variable \( r \) whose range is \( R \).
The Hoeffding bound relates the mean in the sample with the mean in the population:
With probability \( 1-\delta \)
- The true mean of \( r \) is in the range \( \hat{\mu} - \epsilon \) to \( \hat{\mu} + \epsilon \)

- Independent of the probability distribution generating the examples.

Entropy: Sufficient Statistics

Each leaf stores sufficient statistics to evaluate the splitting criterion
- For each attribute
  - If nominal
    - Counter for each observed value per class
  - If continuous
    - Binary tree with counters of observed values
    - Discretization: e.g. 10 bins over the range of the variable
    - Univariate Quadratic Discriminant (UFTD)
Classifying Test Examples
VFDT like algorithms can classify test examples at any time.
To classify a test example:
- The example traverse the tree from the root to a leaf
- It is classified using the information stored at this leaf.
VFDT like algorithms store more information:
- The distribution of attribute values per class.
- Required by the splitting criteria.
- Information collected from hundreds (or thousand's) of examples!

Can we use this information?
- Functional Leaves

Classification strategies
Accurate Decision Trees for mining high-speed Data Streams, J. Gama, R. Rocha: KDD03
Two classification strategies:
- The standard strategy use ONLY information about the class distribution: P(yi)
- A more informed strategy, use the sufficient statistics P(xj | yi)
- Naive Bayes Classifier: P(yi | X) = log(P(yi)) + \sum log (P(xj | yi))
VFDT stores sufficient statistics of hundred of examples in the leaves.

Illustrative Evaluation
Training time / Memory used
Properties of VFDT like Algorithms
Low variance models:
- Stable decisions with statistical support.
- No need for pruning.
- No need for pruning.
Low overfitting:
- Examples are processed only once.
- Decisions are taken using different set of examples.
Convergence: VFDT becomes asymptotically close to that of a batch learner. The expected disagreement is (∆ ≤ p) where p is the probability that an example falls into a leaf.

VFDT like algorithms: Multi-Time-Windows
A multi-window system: each node (and leaves) receive examples from different time-windows of examples.

Useful for change detection:
- Compare distributions at different nodes.
- The RS algorithm.
Automatic adaptation to change by growing the tree

The training set:

Concept 1 + Concept 2

Projecting the final tree in the instance space:

Ensembles of Classifiers

H. Wang, W. Fan, P. S. Yu, and J. Han, “Mining Concept-Drifting Data Streams using Ensemble Classifiers”, KDD’03.

Train $K$ classifiers from $K$ chunks

For each subsequent chunk

• train a new classifier
• test other classifiers against the chunk
• assign weight to each classifier
• select top $K$ classifiers

Ensembles for Mining Skewed Data Streams

J. Gao, B. Ding, W. Fan, J. Han, P. Yu: Classifying Data Streams with Skewed Class Distributions and Concept Drifts. IEEE Internet Computing 12(6): 37-49 (2008)

Ubiquitous Skewed Distributions

• Examples from the interesting classes are scarce
  - Credit card frauds, network intrusions

Existing Stream Classification Algorithms used to be evaluated on balanced data

Problems:

• Ignore minority examples
• The cost of misclassifying minority examples is usually huge

References

Supervised learning

Mining High Speed Data Streams, by P. Domingos, G. Hulten, SIGKDD 2000.


Incremental rule learning and border examples selection from numerical data streams, by D. Agrawal, R. Agarwal, and S. Nath. SIGKDD 2000.


Mining and Learning from high speed data streams, by J. Gama, R. Rocha, P. Medas, SIGKDD 2005.


The Dynamic Weighted Majority (DWM) maintains:

• An ensemble of base learners,
• Predicts using a weighted-majority vote of these experts, and
• Dynamically creates and deletes experts in response to changes in performance.

Experts can use the same algorithm for training and prediction:

• They are created at different time steps so they use different training set of examples.
• The final prediction is obtained as a weighted vote of all the experts.

• The learning element of DWM, first predicts the classification of the training example.

The weights of all the experts that misclassified the example are decreased

Learning from Skewed Data

Learning from batches of examples over time:

• Insufficient positive examples

STEP 1 – Sampling

• Accumulate positive examples

STEP 2 – Generate an Ensemble

• Sample negative examples from the last batch

Ensembles of Classifiers

H. Wang, W. Fan, P. S. Yu, and J. Han, “Mining Concept-Drifting Data Streams using Ensemble Classifiers”, KDD’03.

Train $K$ classifiers from $K$ chunks

For each subsequent chunk

• train a new classifier
• test other classifiers against the chunk
• assign weight to each classifier
• select top $K$ classifiers

Koller and M. Maloof, Using additive expert ensembles to cope with Concept drift, Proc. 22nd ICML, 2005

The Dynamic Weighted Majority (DWM) maintains:

• An ensemble of base learners,
• Predicts using a weighted-majority vote of these experts, and
• Dynamically creates and deletes experts in response to changes in performance.

Experts can use the same algorithm for training and prediction:

• They are created at different time steps so they use different training set of examples.
• The final prediction is obtained as a weighted vote of all the experts.

• The learning element of DWM, first predicts the classification of the training example.

• The weights of all the experts that misclassified the example are decreased

Learning from Skewed Data

Learning from batches of examples over time:

• Insufficient positive examples

STEP 1 – Sampling

• Accumulate positive examples

STEP 2 – Generate an Ensemble

• Sample negative examples from the last batch
Consider an example in which:

- A small portion of the training stream is not used for the creation of the micro-clusters. This portion of the training stream is referred to as the horizon fitting stream segment.
- The remaining portion of the training stream is used for the creation and maintenance of the class-specific micro-clusters.

Consider an example in which:

- The current clock time is \( t_0 \), and
- A horizon of length \( h \) in order to find the micro-clusters in the time period \( (t_0, h, t_0 + t) \).

**Classification on Demand**

How do we find the most effective horizon for classification at a given moment in time?

- A small portion of the training stream is not used for the creation of the micro-clusters. This portion of the training stream is referred to as the horizon fitting stream segment.
- The remaining portion of the training stream is used for the creation and maintenance of the class-specific micro-clusters.

Consider an example in which:

- The current clock time is \( t_0 \), and
- A horizon of length \( h \) in order to find the micro-clusters in the time period \( (t_0, h, t_0 + t) \).

**Classification on Demand**

On demand classification can dynamically select the appropriate window of past training data to build the classifier.

A supervised micro-cluster for a set of:

- d-dimensional points \( X_i \) \( (i=1, ..., n) \)
- with time stamps \( T_i \) \( (i=1, ..., n) \)
- belonging to the class \( C \)

is defined as the (2d+4) tuple \( (CF^1_i, CF^2_i, CF^1, CF^2, n, C) \),

wherein \( CF^2 \) and \( CF^1 \) each correspond to a vector of \( d \) entries.

**Classification on Demand**

On Demand Classification of Data Streams Charu C. Aggarwal, Jiawei Han, Jianyong Wang, Philip S. Yu KDD 04

On demand classification can dynamically select the appropriate window of past training data to build the classifier.

A supervised micro-cluster for a set of:

- d-dimensional points \( X_i \) \( (i=1, ..., n) \)
- with time stamps \( T_i \) \( (i=1, ..., n) \)
- belonging to the class \( C \)

is defined as the (2d+4) tuple \( (CF^1_i, CF^2_i, CF^1, CF^2, n, C) \),

wherein \( CF^2 \) and \( CF^1 \) each correspond to a vector of \( d \) entries.

**Properties of micro-clusters**

Each micro-cluster has a unique id

Micro clusters are additive

Incremental update of micro-cluster

- When a new data point is available:
  - Update the nearest neighbor of the same class
  - If the diameter increases to a threshold value
  - Start a new micro-cluster with a single point

Micro clusters are subtractive

- \( CF^2 - CF^1 \) summarizes the stream in the interval

**Classification on Demand**

On Demand Classification of Data Streams Charu C. Aggarwal, Jiawei Han, Jianyong Wang, Philip S. Yu KDD 04

On demand classification can dynamically select the appropriate window of past training data to build the classifier.

A supervised micro-cluster for a set of:

- d-dimensional points \( X_i \) \( (i=1, ..., n) \)
- with time stamps \( T_i \) \( (i=1, ..., n) \)
- belonging to the class \( C \)

is defined as the (2d+4) tuple \( (CF^1_i, CF^2_i, CF^1, CF^2, n, C) \),

wherein \( CF^2 \) and \( CF^1 \) each correspond to a vector of \( d \) entries.

**Classification on Demand**

On Demand Classification of Data Streams Charu C. Aggarwal, Jiawei Han, Jianyong Wang, Philip S. Yu KDD 04

On demand classification can dynamically select the appropriate window of past training data to build the classifier.

A supervised micro-cluster for a set of:

- d-dimensional points \( X_i \) \( (i=1, ..., n) \)
- with time stamps \( T_i \) \( (i=1, ..., n) \)
- belonging to the class \( C \)

is defined as the (2d+4) tuple \( (CF^1_i, CF^2_i, CF^1, CF^2, n, C) \),

wherein \( CF^2 \) and \( CF^1 \) each correspond to a vector of \( d \) entries.
Classification on Demand

Evaluation
- Once the micro-clusters for a particular time horizon have been determined, they are used to determine the classification accuracy of that particular horizon.
- The process is executed periodically in order to adjust for the changes which have occurred in the stream in recent time periods.
- For this purpose, we use the horizon fitting stream segment.
- A data point is classified in the class of the nearest neighbor micro-cluster.
- Evaluate for all time horizons in the geometric time frame.

Illustrative Example

Predictions at different granularities

Analysis of Web click streams
- Raw data at low levels:
  - seconds, web page addresses, user IP addresses, ...
- Analysts want: changes, trends, unusual patterns, at reasonable levels of details
  - E.g., Average clicking rate in North America sports in the last 15 minutes is 40%
  - Higher than that in the last 24 hours.

Analysis of power consumption streams
- Raw data:
  - Power consumption flow for every household, every minute
- Patterns one may find: average hourly power consumption surges up 30% for manufacturing companies in Chicago in the last two hours than that of the same day a week ago.

A Tilted Time-Frame Model

Up to 7 days:
- 7 days, 24 hours, 15 minutes, 25 minutes

Up to a year:
- 31 weeks, 31 days, 24 hours, 6 am

Logarithmic (exponential) scale:

Presentation Outline

- Block 1: Introduction
- Block 2: Supervised learning on streams
  - Mining and adapting classifiers
  - Novelty detection
- Block 3: Unsupervised learning on streams
- Block 4: Mining evolving social data
- Block 5: Mining under resource constraints
- Block 6: Conclusions and Outlook

Novelty Detection

Classification problems where the full set of class labels is unknown.

Automatic identification of unforeseen phenomena embedded in a large amount of normal data.

Novelty is a relative concept with regard to our current knowledge:
- It must be defined in the context of a representation of our current knowledge.
- Specially useful when novel concepts represent abnormal or unexpected conditions.
- Expensive to obtain abnormal examples.
- Probably impossible to simulate all possible abnormal conditions.
In real problems, as time goes by:
- The distribution of known concepts may change.
- New concepts may appear.

By monitoring the data stream, emerging concepts may be discovered.

Emerging concepts may represent:
- An extension to a known concept (Extension).
- A novel concept (Novelty).


One-class classification


Three layer network:
- The nr. of neurons in the output layer is equal to the input layer.
- The network is trained to reproduce the input at the output layer.

To classify a test example $x$:
- Propagate $x$ through the network and let $y$ be the corresponding output.
- If $\Sigma (x_i - y_i)^2 < \text{Threshold}$ Then the example is considered from class normal.
- Otherwise, $x$ is a counter-example of the normal class.

Nearest Neighbor for One-class Classification

Nearest neighbour for novelty detection (Tax, 2001)

If $d_1 / d_2 > 1 \rightarrow$ reject.

One-class classification

- Model knowledge about a single profile.
- New examples may be identified as members of that profile or not.

Methods based on Frequencies:
- A pattern is surprising if the frequency of its occurrence differs substantially from that expected by chance, given the previously seen data. (TARZAN; Keogh et al., 2002)

Methods based on decision structure:
- Considers decisions taken by each unit in a decision structure.
- In a stable state, the contribution of each unit is likely to remain constant. Changes in the participation of decision units may indicate a conceptual change.

OLLINDA

Cluster-based novelty detection. Spinoza, Carvalho, Gama, SAC 08.

Initial Phase: Supervised, batch mode:
- Start by modeling the normal condition.
- Learns a partial model about what is known.

Based on a set of classified examples.

Second Phase: Process stream of unlabelled examples:
- For each incoming example:
  - If it is explained by the current model: classify the example and discard.
  - If it is not explained: Store it in a short-term memory.
  - Time to Time:
    - Find clusters in the examples stored in the short-term memory.
    - Clusters far away from existing ones: Novel concept.
    - Clusters closed to existing ones: Extend known concepts.
Bibliography on Novelty Detection

- "Online Novelty Detection on Temporal Sequences," by Junshui Ma, Simon Perkins, in the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD) 2003.
- "On-line novelty detection and online clustering based on global and local knowledge," by Yunyue Zhu, Dennis Shasha, in the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD) 2003.
- "OLINDDA: a cluster-based approach for detecting novelty and concept drift in data streams," by E. Spinosa, A. Carvalho, J. Gama, SAC 2007, 499-502.
- "Evaluating novelty detection in data streams," by J. Gama, H. W. L. L. M. de Carvalho, in the ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD) 2005.
- "Classification and Novel Class Detection in Data Streams with Active Mining," by M. Masud, J. Gao, L. Khan, J. Han, B. Thuraisingham, in the SIAM International Conference on Data Mining (SDM) 2005.

Software

- **VFML**
  - A very fast machine learning toolkit for mining high-speed data streams and very large data sets.
- **MOA**
  - A framework for learning from a data stream. Includes tools for evaluation and a collection of machine learning algorithms. Related to the WEKA project, also written in Java, while scaling to more demanding problems.
- **Rapid Miner**
  - [http://rapid-i.com/](http://rapid-i.com/)
  - The Data Stream plugin provides operators for data stream mining and for learning drifting concepts.
- **KNIME**
  - [http://www.knime.org/](http://www.knime.org/)

Presentation Outline

- **Block 1: Introduction**
- **Block 2: Supervised learning on streams**
  - Mining and adapting classifiers
  - Novelty detection
    - Dealing with concept drift in AIS
  - Concept detection in AIS
- **Block 3: Unsupervised learning on streams**
- **Block 4: Mining evolving social data**
- **Block 5: Mining under resource constraints**
- **Block 6: Conclusions and Outlook**

Concept Drift: Application Perspective

CD refers to non-stationary supervised learning problems but there are different types of CD and different types of applications.

Personal recommenders, spam filters, fraud detection, navigation are affected by drifts coming from different sources.
Motivation
View CD research from an application perspective
What is the match between the mainstream CD research assumptions and properties of the applications?
Identify promising future research directions from the application perspective

We will talk about
Why changes appear in different applications?
What are the properties of CD application tasks?
How the application tasks can be categorized in terms of these basic properties?

What is Concept Drift?
The closed world assumption in data mining
• learn a model from examples described by a finite set of features
In reality some important properties are not observed
• hidden variables that influence the concept
Hidden variables may change over time
• concepts learned at one time can become inaccurate
• possible changes in the characteristic properties of the concept
Concept Drift
• changes in the hidden context that can induce more or less radical changes in the target concept
Virtual concept drift – changes due to population drift

Desired Properties of a System Handling Concept Drift
Adapting to concept drift asap
• must have assumptions of what and how may change
Being robust to noise and distinguishing it from concept drift
• e.g. occasionally wrong selection or rating of an item, clicking a link, connection failure (mobile computing)
Elasticity
• discouraging brittleness

Being capable to recognize and react to reoccurring contexts
• such as seasonal differences

Properties of the tasks
DATA task (detection, classification, prediction, ranking)
• type (time series, relational, mix)
organization (stream/batches, data re-access, missing)
DRIFT change type (sudden, gradual, incremental, reoccurring)
• source (adversary, interests, population, complexity)
expectation (unpredictable, predictable, identifiable)
DECISIONS and GROUND TRUTH
• labels (real time, on demand, fixed lag, delay)
decision speed (real time, analytical)
ground truth labels (soft, hard)
costs of mistakes (balanced, unbalanced)
Change types

- sudden
- gradual
- incremental
- recurring

Landscape of applications

<table>
<thead>
<tr>
<th>Types of apps</th>
<th>Monitoring/control</th>
<th>Personal assistance/personalization</th>
<th>Management and planning</th>
<th>Ubiquitous applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Security, Police</td>
<td>Fraud detection, insider trading detection, adversary actions detection</td>
<td>Product or service recommendations, including complimentary search</td>
<td>Paste for budget planning</td>
<td>Paste for location based services, related ads, mobile apps</td>
</tr>
<tr>
<td>Finance, Banking, Telecommunication, Credit Scoring, Insurance, Direct Marketing, Retail, Advertising, e-Commerce</td>
<td>Monitoring &amp; management of customer segments, bankruptcy prediction</td>
<td>Demand prediction, response rate prediction, budget planning</td>
<td>Paste for location based services, related ads, mobile apps</td>
<td>Paste for virtual reality simulations</td>
</tr>
<tr>
<td>E-learning, Professional, Children, e-Learning, Entertainment, Media</td>
<td>Gaming the system, Drop out prediction</td>
<td>Paste for budget planning</td>
<td>Paste for virtual reality simulations</td>
<td>Paste for virtual reality simulations</td>
</tr>
</tbody>
</table>

STREAMS/SENSORS

Online mass flow prediction in CFB boilers

data collected from a typical experimentation with CFB boiler

- asymmetric nature of the outliers
- short consumption periods within feeding stages

Pechenizkiy et al. 2009

CFB Boiler Optimization

Online mass flow prediction in CFB boilers
Food sales prediction: utility of Belgium milk in Sep. 2009

Challenges in food sales prediction (Zliobaite et al., 2009)

Reoccuring and suddent drift in food sales

Antibiotic Resistance Prediction (Tsymbal et al., 2008)

Antibiotic Resistance Prediction
predict the sensitivity of a pathogen to an antibiotic based on data about the antibiotic, the isolated pathogen, and the demographic and clinical features of the patient.

How Antibiotic Resistance Happens

It was an in-sulit moment through the hospital kitchen that Albert was first approached by a member of the Antibiotic Resistance.
Recommender Systems

Lessons learnt from Netflix:

Temporal dynamics is important
Classical CD approaches may not work

(Koren, SIGKDD 2009)

We Know What You Ought To Be Watching This Summer

Something Happened in Early 2004…

Are movies getting better with time?

Multiple sources of temporal dynamics

Both items and users are changing over time

Item-side effects:
  • Product perception and popularity are constantly changing
  • Seasonal patterns influence items’ popularity

User-side effects:
  • Customers ever redefine their taste
  • Transient, short-term bias; anchoring
  • Drifting rating scale
  • Change of rater within household

Categorization of CD Handling Strategies

Evolving Methods vs. Informed Methods
  • Evolving: adapt the learner at regular intervals without considering whether changes have really occurred
    - instance selection and instance weighting
  • Informed: modify the model only after a change was detected
    - used in conjunction with a detection model

Training set manipulation vs. model manipulation
  • Training set:
    • Instance weighing and selection; windowing vs. filtering
  • Model manipulation:
    • VFDT and similar approaches

Outlook to Handling Concept Drift

From general methods to more specific application oriented problems like
  • delayed labeling
  • label availability
  • cost–benefit trade off of the model update

Changing the focus to
  • change description
  • prediction reoccurring contexts and
  • meta learning in addition to change detection

Current situation

Where CD research is heading to?
Where should it go to?

Come in the afternoon to HaCDAIS workshop, listen what other presenters say and share your experience and vision
Applications of unsupervised learning on evolving data

---

References (Block 1) 2/3

Gama et al. (ICDM'03).
Martinez et al. (2006).

Mei & Zhai (KDD'05)

Gama, Menasalvas, Spiliopoulou, Vakali

---

References (Block 2) 1/3


Drift, Information Fusion, Special Issue on Applications of Ensemble Methods, 9(1), pp. 56-67


Yehuda Koren, CACM’10

Drift, Information Fusion, Special Issue on Applications of Ensemble Methods, 9(1), pp. 56-67

---

Presentation Outline

- Block 1: Introduction
- Block 2: Supervised learning on streams
- Block 3: Unsupervised learning on streams (Myra Spiliopoulou)
  - Adapting clusters
  - Probabilistic models
  - Learning on complex data
- Block 4: Mining evolving social data
- Block 5: Mining under resource constraints
- Block 6: Conclusions and Outlook

---

Applications of unsupervised learning on evolving data

---

End of Block 2

Thank you!

Questions?

---

Questions?

Thank you!
Modeling evolving data for unsupervised learning

Data stream model of computation (Guha et al, 2003)

A stream is a sequence of data points \(x_1, x_2, ..., x_i, ...\) that arrive in increasing order of the index \(i\).

An algorithm operating upon a stream is subject to memory constraints must minimize the number of passes over the data.

A learning algorithm operating upon a stream must maintain a good model of the encountered data subject to constraints on time and space.

Maintaining a good model upon all the data?

IF the data generating process is not stationary

THEN we want a good model on the most recent data.

⇒ We must forget the oldest data.

IF the data generating process is stationary

THEN we do not need to remember the oldest data.

⇒ We may as well forget the oldest data.

Unsupervised model adaptation across the time axis

Stream Clustering
Spectral Clustering
Tensor-based clustering
Multi-relational clustering

Dynamic probabilistic models

Topic Discovery & Monitoring
Discovery & Monitoring of (e.g.):
• customer segments
• communities
• vehicle fleets

Tensor based

One stream & static data

Multi-relational clustering

One stream

Dynamic probabilistic models

Tensor based

Presentation Outline

Block 1: Introduction
Block 2: Supervised learning on streams
Block 3: Unsupervised learning on streams
• Adapting clusters
• Probabilistic models
• Learning on complex data
Block 4: Mining evolving social data
Block 5: Mining under resource constraints
Block 6: Conclusions and Outlook

Some core ideas in stream clustering

• Online and offline components
• Summarization of data instances
• Working with snapshots
• Reporting on cluster changes
CluStream (Aggarwal et al., 2003)

Clustering framework for evolving data streams:

- Online and offline components
- Summarization of the information on the stream
- Multiple time windows (snapshots)
- Reporting on cluster changes

**CluStream (Aggarwal et al., 2003)**

Let a stream of $d$-dimensional data points $X_1, X_2, ..., X_i, ...$ that arrive in increasing order of the index $i$.

**Micro-cluster**

- a set of data points
- arrived at $t_i^1, t_i^2, ..., t_i^n$
- retaining only the last $\alpha + 1$ snapshots of order $j$ at any timepoint.

**Pyramidal time frame** for time snapshots:

- $\alpha$ be a positive integer parameter.
- A snapshot of order $j$ is taken at clock value $T$, where $T \mod \alpha = 0$.

Online component for micro-cluster maintenance

For given snapshot parameter $\alpha$ and number of micro-clusters $q$:

- When enough data instances have arrived:
  - Build the first $q$ clusters
- When a new data instance $x$ arrives:
  - IF $x$ is close enough to the centroid of a micro-cluster $M$ AND $x$ falls within the maximum boundary of $M$
    - THEN assign it to that micro-cluster
  - ELSE IF $x$ should become a new micro-cluster
    - THEN either delete an old micro-cluster or merge two old ones
- At each snapshot:
  - Delete the data of the least recent snapshot
  - Compute the most recent snapshot, omitting redundant computations

Offline component for macro-cluster creation:

For given time-horizon $h$ and number of clusters $K$:

1. Find the snapshots belonging to this horizon.
2. Identify the micro-clusters to be considered
3. Re-compute the vectors of each micro-cluster, as they were within the desired snapshots
4. Build the clusters:
   - Pick micro-cluster centroids as initial seeds with probability proportional to micro-cluster size
   - Assign each micro-cluster to its closest seed
   - Re-compute the seeds as weighted centroids

DenStream (Cao et al., 2007)

Density-based stream clustering

- online and offline components
- summarization of data instances
- working with snapshots

Time horizon with an ageing function

**CluStream (Aggarwal et al., 2003)**

Let a stream of $d$-dimensional data points $X_1, X_2, ..., X_i, ...$ that arrive in increasing order of the index $i$.

Micro-cluster over

- a set of data points

**Pyramidal time frame** for time snapshots:

- $\alpha$ be a positive integer parameter.
- A snapshot of order $j$ is taken at clock value $T$, where $T \mod \alpha = 0$.

**Micro-clusters** have unique IDs

**DenStream (Cao et al., 2007)**

Density-based stream clustering

- robust to noise
- clusters need not be spheres

- online and offline components
- summarization of data instances

Micro-clusters instead of data points

- working with snapshots
- Time horizon with an ageing function
DenStream (Cao et al., 2007)

**Time horizon:**
- Damped time window \([0, \text{now}]\)
- Ageing function that assigns exponentially decreasing weights \(f(t) = 2^{-\lambda t}, \lambda > 0\)

**Micro-cluster:**
- Group of proximal data points \(p_i, p_2, \ldots, p_n\)
- That arrived at \(t_i, t_2, \ldots, t_n\)
- With joint weight \(w = \sum f(t_i - t_z)\) at timepoint \(t\)
- With center computed as weighted average of data points
- And radius computed as weighted distance of points from center

DenStream (Cao et al., 2007)

**Core-micro-cluster:**
- Its weight is more than a threshold \(\mu\)
- Its radius is less than a threshold \(\epsilon\)

**Potential core-micro-cluster:**
- Its weight is more than a threshold fragment of \(\mu, \beta, \min (0, 1)\)
- Its radius is less than \(\epsilon\)

**Outlier micro-cluster:**
- Its weight is less than \(\beta\mu\)
- Its radius is less than \(\epsilon\)

DenStream (Cao et al., 2007)

**Online component for micro-cluster maintenance (1/2)**

For given thresholds \(\mu, \beta, \epsilon\) and at each new data instance \(x\):

- Identify closest p-micro-cluster
  - If the updated radius does not exceed the threshold:
    - Merge \(x\) to the p-micro-cluster
  - ELSE
    - Identify closest o-micro-cluster
      - If the updated radius does not exceed the threshold:
        - Merge \(x\) to o-micro-cluster
      - ELSE
        - If the updated o-micro-cluster weight exceeds threshold:
          - Promote it to p-micro-cluster
        - ELSE
          - Turn \(x\) to a (singleton) o-micro-cluster

**Online component for micro-cluster maintenance (2/2)**

For given thresholds \(\mu, \beta, \epsilon\) and every \(T_{\mu, \beta}\) periods:

- For each p-micro-cluster:
  - Recompute its weight and eventually degrade it to an o-micro-cluster
- For each o-micro-cluster:
  - Recompute its weight
    - If weight is larger than threshold, promote it to a p-micro-cluster
    - Else if it is lower than threshold \(\xi\), delete it.

DenStream (Cao et al., 2007)

**Offline component for clustering:**

DBSCAN

- Modified notion of density, using either
  - Max distance of points from micro-cluster center
  - Average distance of points from micro-cluster center

ClusTree (Kranen et al., 2009)

**Anytime stream clustering:**

- Online and offline components

- Summarization of the information on the stream

- Working with snapshots
- Exploitation of available time for quality improvement
Anytime stream mining (Kranen et al., '09)

Principle of anytime algorithms:
- Deliver a model at any time
- Improve the model if more time is available

Model adaptation whenever an instance arrives

Model refinement whenever time permits

Anytime stream clustering with ClusTree (Kranen et al.'09)

Model: Tree structure

Online component:
- At each data point arrival:
- 1. Push data point down the tree as far as time permits.
- 2. Take similar data points on the way down.
- 3. Grow the tree further if time permits.

Activities in a node / micro-cluster:
- Insert a new data point
- Pull a data point down the subtree
- Keep a buffer of data points (hitchhikers), to be pulled down the subtree
- Aggregate data points
- Split - create subtree - for model refinement

A data point:
- traverses the tree towards a leaf node,
- takes (max two) hitchhikers in its way down the tree,
- leaves a hitchhiker at a node, if their ways do not match anymore.

Cluster monitoring: What is a cluster?

All objects of a region
- The dimensions and the metric are invariant.

All objects satisfying a function
- The dimensions are invariant.

A set of objects

Some core ideas in stream clustering

- Online and offline components
- Summarization of data instances
- Working with snapshots
- Reporting on cluster changes
MONIC (Spiliopoulou et al., 2006)

1. Partition the time axis into timepoints \( t_1, \ldots, t_m \).
2. Cluster (adaptively?) the data at each snapshot.
3. Compare the sets of clusters (need not be of equal strength)

**Matching model:**
- How to track a given cluster?
- When is a new cluster a mutation of an old one?

**Transition model:**
- Is an old cluster associated with exactly one new cluster?
- How did the cluster change with respect to other clusters?
- What kind of internal changes did the cluster experience?

MONIC – Matching Model (Spiliopoulou et al., 2006)

For two given clusterings \( \xi, \xi' \) (built at \( t_1, t_2, t_1 < t_2 \)) and for threshold \( \tau \):

1. For each \( X \) in \( \xi \), compute
   \[
   \text{overlap}(X, Y) = \frac{\sum_{i \in X} q_p(i, t_i) \cap q_p(X, t_i)}{\sum_{i \in X} q_p(i, t_i) + \sum_{i \in Y} q_p(X, t_i)}
   \]
2. Choose \( Y_{best} \) with max overlap to \( X \)
3. If overlap\( (X, Y_{best}) \) does not exceed \( \tau \), then discard \( Y_{best} \)

MONIC – Transition model

Cluster Transitions

- **External** (w.r.t. the whole clustering)
  - survival
  - split into multiple clusters
  - absorption from a cluster
  - disappearance
  - appearance of a new cluster

- **Internal** (w.r.t. the cluster itself)
  - size shrink / expand
  - compactness more compact / more diffuse
  - location no change

MONIC – Lifetime of clusters and clusterings

**Lifetime of a cluster \( X \):**
- Number of adjacent timepoints where \( X \) has survived
- Since lifetime
- Lifetime under internal transitions
- Lifetime with absorptions

**Survival ratio of a clustering:**
- Portion of clusters that survive at the next timepoint

**Passforward ratio of a clustering:**
- Portion of clusters that survive or get absorbed at the next timepoint

MONIC on the ACM H2.8:

- Knowledge discovery
  - spatial
  - association rules
- Cluster knowledge discovery
  - spatial
  - association rules

Presentation Outline

- **Block 1:** Introduction
- **Block 2:** Supervised learning on streams
- **Block 3:** Unsupervised learning on streams
  - Adapting clusters
    - Probabilistic models
    - Learning on complex data
- **Block 4:** Mining evolving social data
- **Block 5:** Mining under resource constraints
- **Block 6:** Conclusions and Outlook
Dynamic topic model according to Blei & Lafferty (2006)

Static model with Dirichlet priors $\alpha$ (documents), $\beta$ (topics)

$$N \quad z \quad w \quad \theta \quad \alpha \quad \beta$$

$\alpha$ (documents), $\beta$ (topics)

$K$ number of topics (fixed)

Method inferring $\alpha$, $\beta$

Learning on the whole vocabulary

$A$

$k$th topic at slice $i$ evolves smoothly from $k$th topic of slice $i-1$

Online LDA (AlSumait et al., 2008)

Underpinnings:
- Gibbs sampling for the estimation of
  - document-specific topic proportions $q$
  - topic-specific word proportions $f$

Time axis is discretized – sliding window of size $d$ slices
The model generated at slice $i/d$ is used as prior for LDA for the stream portion $S$ in slice $i$

LDA thread
- L. AlSumait et al. On-line LDA: Adaptive Topic Models for Mining Text Streams with Applications to Topic Detection and Tracking (ICDM’08)
- R. Nallapati et al. Multiscale Topic Tomography (KDD’07)

PLSA thread
- Qiaozhu Mei “Contextual Text Mining” – PhD thesis (2009), papers since KDD’05
- A. Gohr et al. Topic Evolution in a Stream of Documents (SDM’09)

Online LDA (AlSumait et al., 2008)

Underpinnings:
The model generated at slice $i/d$ is used as prior for LDA for the stream portion $S$ in slice $i$
- Stream $S$ introduces new words.
- Parameters of topic $k$ are determined from its past distributions:

$$\beta_k' = B_k^{-1} \omega^\delta$$

Evolution matrix of topic $k$ has $V$ columns, a column is $\phi_i$

Vector of weights for $\theta$, $i=[d, ...]$
Online LDA (AlSumait et al., 2008)
Evaluation for document modeling
- Comparison to an LDA method that remembers all data.

Online LDA (AlSumait et al., 2008)
Evaluation on topic evolution
- Interesting findings on the evolution of NIPS topics

Adaptive PLSA (Gohr et al., 2009)
Underpinnings:
Time axis is discretized – sliding window of size $\delta$ slices
The model generated at slice $t-1$ is used as basis for PLSA adaptation for the stream portion $\delta$ in slice $t$.
- Old documents are forgotten, new documents are folded-in.
- Old words are forgotten, new words are folded-in.

Adaptive PLSA (Gohr et al., 2009)
Evaluation for document modeling
- Comparison to a PLSA method that re-learns from scratch
- Stream of SIGIR (2000–2007)

Adaptive PLSA (Gohr et al., 2009)
Topic threads in SIGIR (2000–2007)

Contextual text mining with generative models (Q. Mei)
Temporal evolution of topics over a stream $D$ of weblog documents
Let $d_i$ be the document arriving at timepoint $t_i$, $2 \leq t_i \leq T$.
Assume $K$ global topics $\theta_1, \theta_2, \ldots, \theta_K$ and background topic $\theta_B$.
The likelihood of word $w$ in document $d_i$ at time $t_i$ is:
$$p(w|d_i,t_i) = \lambda_\theta p(w|\theta_0) + (1 - \lambda_\theta) \sum_{k=1}^{K} p(w|\theta_k)$$
Topic evolution on a text stream (Mei et al., 2005)
Stream of hurricane news: K = 5 topics after removal of non-discriminative words (background model), labeling of the topics and weighting the strength of the words in each topic label at each timepoint.

Presentation Outline

- Block 1: Introduction
- Block 2: Supervised learning on streams
  - Adapting clusters
  - Probabilistic models
- Block 3: Unsupervised learning on streams
  - Learning on complex data
- Block 4: Mining evolving social data
- Block 5: Mining under resource constraints
- Block 6: Conclusions and Outlook

Unsupervised model adaptation across the time axis

Stream Clustering
Dynamic probabilistic models
Spectral Clustering
Tensor-based clustering
Multi-relational clustering

One stream
One stream & static data
Multiple streams

Example: Recommendations over users, items and tags


Example: Recommendations over users, items and tags

Clustering over multiple, correlated streams

Challenges:
1. Multiple streams (and some *almost* static data)

<table>
<thead>
<tr>
<th>Users</th>
<th>Tags</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>Postings</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

2. Correlated streams of different speeds

<table>
<thead>
<tr>
<th>Customers</th>
<th>Transactions</th>
<th>Items</th>
</tr>
</thead>
</table>

3. Streams of permanent objects, not of data points:
- Data objects may show up more than once.
- New objects may show up at any time.
- Old objects might be forgotten (in *some* applications).

Incremental clustering + Stream clustering on complex data

How to model data and their relations?
- Data on Tensor
- Data on Cube (multi-relational)

• Paradigm covers graphs
• Solid theoretical underpinnings and mathematical groundwork
• Combines with advances on generative models

Modeling temporal smoothness (Chi et al., ‘07)

1. Two aspects of model quality (Chakrabarti et al., ‘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 cost (Chakrabarti et al., ‘06)
   - Build a model that minimizes cost w.r.t previous model (Chi et al., ‘07)

Evolutionary clustering (Chakrabarti et al., ’06), (Chi et al., ‘07)

1. Two aspects of model quality (Chakrabarti et al., ’06):
   - Snapshot cost \( CS \)
   - Temporal cost \( CT \)

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 cost (Chakrabarti et al., ’06)
   - Build a model that minimizes cost w.r.t. previous model (Chi et al., ’07)

Evolutionary clustering (Chakrabarti et al., ‘06), (Chi et al., ‘07)

1. Two aspects of model quality (Chakrabarti et al., ‘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 cost (Chakrabarti et al., ’06)
   - Build a model that minimizes cost w.r.t. previous model (Chi et al., ‘07)
One topic thread towards tensor-based stream clustering

Evolutionary clustering (with temporal smoothness)

- Incremental tensor-based clustering

MetaFac (Lin et al., KDD’09)

1. Community-related data modeled on multiple tensors
   - Facet: set of objects of the same type (tensor mode)
   - Relation: interaction among facets - at least binary
   - Multi-relational hypergraph: facets & relations among them
2. Metagraph factorization for community discovery
3. Generalization for evolving data:
   - Given a metagraph $G$ and a timestamped sequence of data tensors defined on $G$, find a non-negative core tensor and corresponding factors: facets for each timepoint $t$
   - assuming consistent interactions in a community

More on tensor-based clustering for community discovery in Block 4
Mining a multi-table data set (Kroegel, 2003)

Non-naive solution: Propositionalization

- Specify a target table $T$
- For each tuple $x$ in $T$ and tupleset $T_x$ in a table $T'$
  - build four columns per numerical attribute in $T'$ and fill in the min, max, avg and count of values for $x$ in $T_x$
  - build one column per value of nominal attribute in $T'$ and fill it as extensions to $T$
- Mine the extended table $T$

Propositionalization in the example dataset

Mining a multi-table data stream

Challenges:

- Dealing with the past
  - Which data to forget ...
  - for each table?
  - Which attribute values to forget?

- Dealing with the future
  - How many nominal values may show up per attribute?
  - How long to wait until a tuple shows up?

Preprocessing at each mining run: Tuple ranking

For each visible tuple $x$ of target stream $T$:

- For each stream $T'$ associated with $x$ expand $x$ with the tuples inside the window
- For each stream $T'$ associated with $x$, cache
  - weight each tuple in $T'$ with the number of references to it
  - assign a bonus to tuples that are already in the cache
  - apply a decay function to reduce the weight of older tuples
  - update the cache with the top-rank tuples, fetching them from secondary storage if necessary

Preprocessing: Tuple expansion

Target stream $T$

Tuple concatenation

Other stream

generate columns for avg, count, max, min

generate one column for each attribute value

Components:

- Typification of streams with respect to oblivion
  - window-based
  - cache-based
- Tuple ranking
  - Management of windows and caches
- Tuple expansion with data from multiple streams
- Incremental clustering algorithm (Siddiqui & Spiliopoulou, DS'09)
- Incremental tree induction algorithm (__, SSDBM'10)

Stream preparation at timepoint $t$

window-based

cache-based

Preprocessing: Tuple expansion

Target stream $T$

Tuple concatenation

Other stream

generate columns for avg, count, max, min

generate one column for each attribute value

Components:

- Typification of streams with respect to oblivion
  - window-based
  - cache-based
- Tuple ranking
  - Management of windows and caches
- Tuple expansion with data from multiple streams
- Incremental clustering algorithm (Siddiqui & Spiliopoulou, DS'09)
- Incremental tree induction algorithm (__, SSDBM'10)
Preprocessing: Tuple expansion

- Target stream \( T \)
- Other stream \( T' \)

Tuple concatenation

- Generate one column for each attribute value up to \( r_A \) columns
- If value is no longer referenced, then free column
- If \( r_A \) is exceeded, then cluster values of similar tuples together

Tuple summarization

- Generate columns for avg, count, max, min
- Stream preparation at timepoint \( t \)
  - Window-based
  - Cache-based

Mining a multi-table stream (Siddiqui et al., 2009)

Components:
- Typification of streams with respect to oblivion
- Tuple ranking
- Management of windows and caches
- Tuple expansion with data from multiple streams
- Incremental clustering algorithm (Siddiqui & Spiliopoulou, DS'09)
- Incremental tree induction algorithm (__, SSDBM’10)

Components:
- Window-based
- Cache-based

Tree Induction on PKDD’99 competition data (Siddiqui et al., 10)

Impact of remembered objects on quality over time:

- Stream Clustering
- Dynamic probabilistic models
- Spectral Clustering
- Tensor-based clustering
- Multi-relational clustering
- One stream & static data
- Tensor based

New book “SOCIAL NETWORK DATA ANALYTICS”, ed. Charu Aggarwal:
Chapter 6 “EVOLUTION IN SOCIAL NETWORKS: A SURVEY” by Myra Spiliopoulou

References (Block 3) (1/3)

C. Aggarwal, J. Han, J. Wang, P. Yu. “A Framework for Clustering Evolving Data Streams”, 29th Int. Conf. on Very Large Data Bases (VLDB’03), Berlin, Germany, 2003.
F. Cao, M. Ester, W. Qian, A. Zhou. “Density-Based Clustering Over an Evolving Data Stream with Noise”, SIAM Int. Conf. on Data Mining (SDM’06), 2006.

References (Block 3) (2/3)

Q. Mei. 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.
Social data which are evolving in fast rates.

Users participate massively in Web 2.0 applications such as:

- social networking sites (e.g. Facebook, LinkedIn)
- blogs, microblogs (e.g. Twitter)
- social bookmarking/tagging systems (e.g. Delicious, Digg)

Social data refer to different types of actors and interactions:

- users
- content
- metadata

that are associated via various types of activities:

- interaction
- relationship

Motivation for Mining Social Data

- The availability of massive sizes of data gave new impetus to data mining.
  - e.g. more than 400 million active Facebook users, sharing on average more than 25 billion pieces of content each month (Facebook Statistics 2010)
- Mining social web data can act as a barometer of the users' opinion.
- Non-obvious results may emerge.

Various mining techniques are/can be used for applications such as recommender systems, automatic event detectors, etc.
Motivation for Mining Evolving Social Data

- Analysis of social data referring to a given time period is of great importance. However, as social data are evolving rapidly, the analysis of the evolution of social data over time is deemed crucial.

Applications:
- Identifying over time the events that affect social interactions
- Tracking posts in a micro-blogging website to identify floods, fires, riots, or other events and inform the public
- Highlighting trends in users' opinions, preferences, etc.
- Companies can track customers' opinions and complaints in a timely fashion to make strategic decisions
- Tracking the evolution of groups (communities) of users or resources, finding changes in time and correlations
- Develop better personalized recommender systems to improve user experience
- Scientists can more easily identify and relate social phenomena

Structures and models

The network model as an obvious choice...

- Nodes represent entities/objects and edges represent relations
- Different types of nodes and edges
- Weighted/unweighted
- Directed/undirected

In a weighted network each edge $(i,j)$ has an arithmetic value (weight) $w_{ij}$ indicating its significance

Presentation Outline

- Block 1: Introduction
- Block 2: Supervised learning on streams
- Block 3: Unsupervised learning on streams
- Block 4: Mining evolving social data (Athena Vakali)
  - Structure and Models
  - Community Detection in Evolving Social Graphs
  - Applications of Evolving Community Detection
- Block 5: Mining under resource constraints
- Block 6: Conclusions and Outlook

Structures for static social data

- Hypergraph: generalization of a graph where an edge connects more than one nodes
- Folksonomy: lightweight knowledge representation emerging from the use of a shared vocabulary to characterize resources – tripartite hypergraph (technorati, $\text{web2.0}$)
- Projection on bipartite & unipartite graphs for simplicity (Yu Yeung)
  - e.g. tag-tag network where two tags are connected if they have been assigned to the same resource
- Graph's structure can be encoded in an adjacency matrix $A$ if $G$ is unweighted ($A_{ij} = 1$, $(i,j) \in E$) or a similarity matrix $M$ if $G$ is weighted ($M_{ij} = w_{ij}$, $(i,j) \in E$)

Motivation for Mining Evolving Social Data

- Social data are interconnected through associations forming a network or graph $G(V,E)$, where $V$ is the set of nodes and $E$ is the set of edges.
- $V$ may be partitioned into communities:
  - nodes represent entities/objects and edges represent relations
  - different types of nodes and edges
  - weighted/unweighted
  - directed/undirected

In a weighted network each edge $(i,j)$ has an arithmetic value (weight) $w_{ij}$ indicating its significance

Presentation Outline

- Block 1: Introduction
- Block 2: Supervised learning on streams
- Block 3: Unsupervised learning on streams
- Block 4: Mining evolving social data (Athena Vakali)
  - Structure and Models
  - Community Detection in Evolving Social Graphs
  - Applications of Evolving Community Detection
- Block 5: Mining under resource constraints
- Block 6: Conclusions and Outlook

Structures for static social data

- Hypergraph: generalization of a graph where an edge connects more than one nodes
- Folksonomy: lightweight knowledge representation emerging from the use of a shared vocabulary to characterize resources – tripartite hypergraph (technorati, $\text{web2.0}$)
- Projection on bipartite & unipartite graphs for simplicity (Yu Yeung)
  - e.g. tag-tag network where two tags are connected if they have been assigned to the same resource
- Graph’s structure can be encoded in an adjacency matrix $A$ if $G$ is unweighted ($A_{ij} = 1$, $(i,j) \in E$) or a similarity matrix $M$ if $G$ is weighted ($M_{ij} = w_{ij}$, $(i,j) \in E$)
Models for evolving social data

- Coarse-grained
- Fine-grained

Evolution of graph structure over time

Assumptions for modeling graph updates

- Assuming discrete time-step
- Between-time step and \( t+1 \):
  - Only one update operation can take place (single update)
  - Multiple update operations can take place (batch update)

Models for evolving social data

- Evolutionary bipartite graphs as sequences of observed new edges and weights
- Updates are modeled with slice matrices \( S_1, S_2, \ldots, S_k \) and \( k \in \mathbb{N} \), (Figure)

Evolution of the graph model using aggregation techniques

- The graph at time step \( t \) can be created by aggregating the updates observed at previous time-slices
- Creation and update of a “time aggregate adjacency matrix” \( M(t) \) placing different emphasis on links based on their age:
  - Global Aggregation
  - Sliding Window
  - Exponential Weighting

Evolution of graph structure \( G_t \) to \( G_{t+1} \):

- Growth – arrival of new nodes
- Shrinkage – departure of existing nodes
- Densification – creation of new edges
- Sparsification – deletion of existing edges
- Weight updates – weight increase / decrease for existing edges

Preprocessing techniques:

- Identify graph segments consisting of similar snapshots and compare smooth graph approximation for each segment (Figure)
Sliced matrices modeled with Updates are weights. Evolving structural similarity betweeness centrality tf-idf cosine similarity co-occurrence between nodes.

Quantitative, for assessing relations

Community Detection in Social Graphs

<table>
<thead>
<tr>
<th>Measures</th>
<th>Qualitative, for evaluating community structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>co-occurrence</td>
<td>modularity</td>
</tr>
<tr>
<td>cosine similarity</td>
<td>local modularity</td>
</tr>
<tr>
<td>betweenness centrality</td>
<td>node outwardness</td>
</tr>
<tr>
<td>structural similarity</td>
<td></td>
</tr>
</tbody>
</table>

Methods

- Graph partitioning
- Spectral algorithms
- Clustering
- Methods based on statistical inference
- Divisive algorithms
- Dynamic algorithms
- Modularity-based methods

Motivation for community detection in social data

- Interesting to look for communities since they form functional units (e.g., sets of resources relevant to a topic) for the local (within group) and the global (over the whole graph) level.
- Community detection can influence tasks such as:
  - designing crawl strategies on the Web
  - predicting evolution of network-based data collected from the Web
  - developing improved methodologies for content outsourcing, recommendation, etc.
  - revealing (hidden) emerging phenomena on the Web
  - understanding social interactions in Web 2.0 settings
  - automatic event identification from social data
Community Detection in Evolving Social Graphs

Classification of methods for community detection in evolving graphs

Community detection is performed on graphs and evolving structures

Based on clique percolation

Classification of methods for community detection in evolving graphs

Community mapping

Approximations and order of relative overlap

1. Compute relative overlap
2. For each community 
3. Match pairs of communities in descending order of relative overlap

Typical procedure:
1. Static snapshot graphs for each time-step
2. Traditional community detection technique applied on each snapshot
3. For each snapshot: mapping of each community to its predecessor and successor using a similarity measure and/or temporal smoothing technique

Community Detection with Clique Percolation Method (CPM)

- Community detection as a percolation process (haloct)
- Starting from a $k$-clique, nodes are attached as long as they are reachable through clique adjacency
- $k$-clique community: the union of $k$-cliques that can be reached from one to the other through a sequence of adjacent $k$-cliques

CPM applied to static time-dependent snapshots of two social networks (haloct)
- Los Alamos cond-mat article archive (142 months, 30000 authors)
- Mobile phone-call network (52 weeks, 4 million users)

Community mapping

$G'$ joint graph comprising the union of links from networks $G'_t$ and $G'_t$

$G_c, G_t$: community structure of $G'_t$ and $G'_t$

For each community $C'_t \in C'_t$ and $C'_t \in C'_t$

there is one community $C'_t \in C'_t$ containing it.

Mapping procedure:
1. For each $C'_t \in C'_t$ find sets $C'_t$ and $C'_t$ of enclosed communities
2. For each pair $(C'_t, C'_t)$, $C'_t \in C'_t$ and $C'_t \in C'_t$, compute relative overlap
3. Match pairs of communities in descending order of relative overlap

Community evolution results in the co-authorship network
Community detection in snapshot graphs and mapping across successive structures based on Community detection in snapshot graphs and mapping across successive structures based on

Community detection based on mutual awareness expansion

- Identification and evolution of thematic communities in the Blogosphere (Lin07)
- Mutual awareness leads to community formation.
- Given a query, construct a time-dependent directed graph where nodes are bloggers and edges their interactions.
- Model mutual awareness expansion with a random walk process and find community structure for each graph $G(V,E)$
  - Compute expected path length (symmetric social distance - ssd) between every pair of nodes
  - Iteratively isolate a set of nodes $S$ (community) from $G$ so that ssd between members of $S$ and $V\setminus S$ is maximized
- Identify community evolution by mapping communities from two subsequent structures

Community mapping using interaction correlation

- Each community is represented with a vector in an interaction space, modeling the interactions between all actors
- Histogram intersection is used to compute the interaction correlation of two communities
- For each community $C_t$ find $\text{Post}(C_t^+)$: the one that is more similar to it in the future
- For each community $C_t^{+1}$ find $\text{Prior}(C_t^\rightarrow)$: the one that is more similar to it in the past

Evolution of communities in the "Katrina"-query graph

- Political community about "Katrina" emerges
- Split: $C_t$ has more than one successor
- Mergence: $C_t$ owns more than one predecessor
- Birth: $C_t$ has no predecessor
- Death: $C_t$ has no successor

CoreTracker

- Observations: community membership is generally unstable
- Focus on representative core nodes to track community evolution (CoreTracker)
- Community detection in graph snapshots with whichever algorithm
- Identification of core nodes in each community based on centrality
- Community $C_{t+1}$ successor of $C_t$ if they share a common core vertex AND $C_{t+1}$ shares a common core vertex with an ancestor of $C_t$
Evolutionary community identification

- Real world data are usually ambiguous and noisy. An algorithm extracting communities at each time-step independently of the other, often results in community structures with high temporal variation.

- Evolutionary community identification methods lead to smoother community evolution, as they utilize the history of the community structure to maximize temporal smoothness

Approaches:
- Traditional clustering techniques in an evolutionary setting
- Spectral clustering
- Non-negative matrix factorization
- Methods identifying graph stream segments and detecting community structure in them

Spectral clustering approach

Spectral clustering uses the spectrum of the graph similarity matrix to perform dimensionality reduction for clustering in fewer dimensions

**Evolutionary spectral clustering** applied in multi-mode networks [Tang08]

- m-mode network: $X_1 \times X_2 \times \cdots \times X_m$
- Interaction between two modes approximated by interactions between communities: $K^{m-1}C^{(m-1)}(X_{m-1})$ where $C^{(m)}(X) = \sum_{i=1}^{N} \text{ Community membership for actors in mode } X$ and:
  - Group interaction

**Spectral clustering approach**

Spectral clustering uses the spectrum of the graph similarity matrix to perform dimensionality reduction for clustering in fewer dimensions

**Evolutionary spectral clustering** applied in multi-mode networks [Tang08]

- m-mode network: $X_1 \times X_2 \times \cdots \times X_m$
- Interaction between two modes approximated by interactions between communities: $K^{m-1}C^{(m-1)}(X_{m-1})$ where $C^{(m)}(X) = \sum_{i=1}^{N} \text{ Community membership for actors in mode } X$ and:
  - Group interaction

The effect of temporal change is adopted as a regularization term

Spectral clustering approach

Spectral clustering uses the spectrum of the graph similarity matrix to perform dimensionality reduction for clustering in fewer dimensions

**Evolutionary spectral clustering** applied in multi-mode networks [Tang08]

- m-mode network: $X_1 \times X_2 \times \cdots \times X_m$
- Interaction between two modes approximated by interactions between communities: $K^{m-1}C^{(m-1)}(X_{m-1})$ where $C^{(m)}(X) = \sum_{i=1}^{N} \text{ Community membership for actors in mode } X$ and:
  - Group interaction

The effect of temporal change is adopted as a regularization term
Non-negative matrix/tensor factorization

Model of community with:
- Structural aspects: community graph representing intra-community interactions.
- Temporal aspects: community intensity representing its activity level over time.

Representation of blogosphere at time \( t \) as a mixture of community graphs weighted by their intensities – Community Factorization [Chi07]

Let \( \mathbf{A} = \{ A_1, A_2, \ldots, A_T \} \in \mathbb{R}^{m \times n \times T} \) be the tensor created by the adjacency matrices of the successive graph snapshots. From each \( A_t \), a set of basis dense subgraphs \( \{ \mathbf{B}_t \} \) is extracted via graph partitioning. Basis tensor \( \mathbf{B} \) is obtained by stacking all \( \{ \mathbf{B}_t \} \) together.

Community graph: \( \mathbf{C}_i = \sum_v \mathbf{u}_{vi} \mathbf{B}_v \), where \( w \) is the weight of all basis subgraphs and \( \mathbf{u}_{vi} \) weight equivalent to

\[
\min \sum_{i=1}^{T} \| \mathbf{A}_i \|_F^2 + \lambda \sum_{i=1}^{T} \| \mathbf{C}_i \|_F^2 + \lambda \sum_{i=1}^{T} \| \mathbf{v}_i \|_2^2,
\]

where \( \mathbf{v}_i \) is the intensity of community \( C_i \) at time \( t \).

Identification of graph stream segments and community structure

Problem definition: given a graph stream, find good change points in time to segment it, and identify communities within each segment.

Community factorization

\( U, V \) are identified by non-negative matrix factorization.

Regularization terms are also inserted to smooth community temporal trends (evolution of intensity over time).

Community graph and temporal trends for community related to hurricane Katrina.

Identification of graph stream segments and community structure
GraphScope
Method applied on unweighted undirected bipartite graphs [Sun07]
Operates without parameters based on Minimum Description Length (MDL)
Considering the graph as a binary adjacency matrix with "1" denoting the presence of a link between two nodes, the goal is to organize the graph into homogeneous sub-matrices with low entropy and compress them separately.

\[
\text{Graph scope cost} = \sum_{ij} \left( \log \left( \frac{2}{n} \right) + \log \left( \frac{1}{e} \right) \right) + \sum_{ij} \log \left( \frac{2}{n} \right)
\]

\[
\text{Partition encoding cost} = \sum_{ij} \log \left( \frac{2}{n} \right) + \log \left( \frac{1}{e} \right) + \log \left( \frac{2}{n} \right) + \log \left( \frac{2}{n} \right)
\]

Aim: the minimization of total encoding cost

Finally
Compression with GraphScope achieves in fitting the graph in less than 4% of the original space
Data are organized into few homogeneous communities

Stream-Group
Stream-Group: applied on dynamic weighted directed graphs [Duan09]
Uses:
- Random Walk with Restart to compute the graph’s relevance matrix \( R \)
  - \( r_{ij} \) expresses the probability that random walker will stay at \( i \) when starting from \( j \)
  - relevance scores between within community nodes are usually higher than the ones between different communities
- an extension of modularity to evaluate the goodness of a partition

Stream-Group
Stream-Group: applied on dynamic weighted directed graphs [Duan09]
Uses:
- Random Walk with Restart to compute the graph’s relevance matrix \( R \)
  - \( r_{ij} \) expresses the probability that random walker will stay at \( i \) when starting from \( j \)
  - relevance scores between within community nodes are usually higher than the ones between different communities
- an extension of modularity to evaluate the goodness of a partition

Procedure
1. For each new graph arrival, identify its community structure
2. Compute similarity between new structure and current segment’s structure
3. If similarity over threshold, the community structure of the current segment is updated incrementally with the new data
4. else, a new segment is initiated
**Stream-Group**

*Stream-Group:* applied on dynamic weighted directed graphs (Duan09)

**Uses:**
- Random Walk with Restart to compute the graph’s relevance matrix $R$
  - $r_j$ expresses the probability that random walker will stay at $j$ when starting from $j$
  - relevance scores between within community nodes are usually higher than the ones between different communities
- an extension of **modularity** to evaluate the goodness of a partition

**Procedure**
1. For each new graph arrival, identify its community structure
2. Compute similarity between new structure and current segment’s structure
3. If similarity over threshold, the community structure of the current segment is updated incrementally with the new data
4. else, a new segment is initiated

---

**Mixed methods combining evolutionary community detection and community mapping**

**Approaches:**
- Identification of temporally smooth local communities and then mapping across successive community structures (Kim09)
- Community detection on graph segment approximations and then mapping across successive community structures (Yang09)

---

**Tracing the Timeline of Networks**

- Identifies changepoints in the graph stream by measuring the distance between nodes (new-born, deceased, stable) of subsequent snapshots (Yang09)
  - Generates smooth approximations of graph segments to address the problem of noise, by iteratively adding to the approximation graph edges whose nodes have small distance

---

**Community detection on graph segment approximations and then mapping across successive community structures**

- Evaluates embedding and projection community detection and community mapping

---

**Stream-Group**

*Stream-Group:* applied on dynamic weighted directed graphs (Duan09)

**Uses:**
- Random Walk with Restart to compute the graph’s relevance matrix $R$
  - $r_j$ expresses the probability that random walker will stay at $j$ when starting from $j$
  - relevance scores between within community nodes are usually higher than the ones between different communities
- an extension of **modularity** to evaluate the goodness of a partition

**Procedure**
1. For each new graph arrival, identify its community structure
2. Compute similarity between new structure and current segment’s structure
3. If similarity over threshold, the community structure of the current segment is updated incrementally with the new data
4. else, a new segment is initiated
Maximizing clusters density of links between local communities modeled as a $l$-clique-by-clique based on an automatic determination parameter $\epsilon$ and density-based, DENGRAPH.

Community mapping

Community detection method requiring definition of three parameters:
- Identification of all $l$-cliques in each approximate graph
- Multiple community memberships are solved assigning the respective nodes to the community with which they have most interactions (weight $\mu$).
- Each community attract its neighboring nodes whose weight to the community is over a judgment threshold $\epsilon$.

Community correlation and evaluation

This method does not apply community mapping directly, but rather evaluates the smoothness of successive community structures by calculating their correlation considering both edge and node overlap.

A Particle – and Density-based Method

Removes the constraint for identical number of communities between successive structures ($\text{Kmer}$)

Generates smooth approximations of graph segments to address the problem of noise by iteratively adding to the approximation graph edges whose nodes have small distance.

Community detection in a $l$-clique-by-clique ($\text{Kmer}$) with cost embedding $\text{cost}_i = \sigma \epsilon_i t N_i - \sigma \epsilon_i t N_i$.
Effects of increased proximity among nodes upon the community

\[ |\text{dist}(p, q) - \text{dist}(p, q)| \leq \varepsilon \]

New Core Vertex | New Border Vertex | New Neighbor

Local Clustering

Creation | Absorption | Merge

Effects of decreased proximity or node/edge removal

\[ |\text{dist}(p, q) - \text{dist}(p, q)| > \varepsilon \]

Lost Core | Lost Edge to Core | Lost Neighbor | Lost Border

Local Clustering

Removal | Split | Reduction

Clust Neighborhood for other id

Reduction | Move

Presentation Outline

- Block 1: Introduction
- Block 2: Supervised learning on streams
- Block 3: Unsupervised learning on streams
- Block 4: Mining evolving social data
  - Structure and Models
  - Applications of Mining Evolving Social Data
- Block 5: Mining under resource constraints
- Block 6: Conclusions and Outlook

Applications of Mining Evolving Social Data

The results of community detection, or different mining techniques, on evolving social data can be exploited in applications:

- Social network analysis
  - Data mining can offer new insights in the analysis of the structure and evolution of social networks.
  - Important technique: evolving community detection, as communities constitute meaningful units of organization
  - The methods we mentioned so far have presented research results for data networks derived from the following sources…

Social network analysis

<table>
<thead>
<tr>
<th>Source</th>
<th>Community detection method</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic datasets</td>
<td>ChiC7, Ghani07, Gkanti07, Lüdi07, Tang08, Yang08</td>
</tr>
<tr>
<td>Benchmark datasets (e.g., dolphin's network, G2E, VAST dataset)</td>
<td>ChiC7, Yang08</td>
</tr>
<tr>
<td>Article online archives (e.g., DBLP, arXiv)</td>
<td>Gkanti07, Lüdi07, Pabai07, Tang08, Wang08, Yang08</td>
</tr>
<tr>
<td>Mobile phone call networks</td>
<td>Pabai07, Sun07, Wang08, Yang08</td>
</tr>
<tr>
<td>Movie actor collaboration dataset</td>
<td>Wang08</td>
</tr>
<tr>
<td>Email e-mail dataset</td>
<td>Duan08, Faloutsos08, Sun07, Tang08, Wang08, Yang08</td>
</tr>
<tr>
<td>Blog data</td>
<td>ChiC7, Lüdi07, Lüdi07</td>
</tr>
<tr>
<td>Flickr</td>
<td>Chakrabarti08</td>
</tr>
<tr>
<td>Mobile device proximity networks</td>
<td>Sun07</td>
</tr>
<tr>
<td>Company call networks</td>
<td>Yang08</td>
</tr>
<tr>
<td>Football match schedule</td>
<td>Kim09</td>
</tr>
<tr>
<td>Department e-mail dataset</td>
<td>Gkanti07</td>
</tr>
<tr>
<td>Digg</td>
<td>Lüdi07</td>
</tr>
</tbody>
</table>
Trend detection

- Social data fluctuate in their structure and frequency as they evolve. At each time-period there are some topics, images, tags, etc. that are most popular amongst users (trends). Data mining can be used for detecting trends in evolving social data.
- Trends can be identified globally or even locally (within communities) and they usually indicate what interests users the most at a given time.

Trend identification in Twitter

- Twitter: popular microblogging website where users are allowed to post short messages (up to 140 chars) and "follow" the posts of others.
- Rich source of rapidly evolving social data which are also public and suitable for trend detection.
- Recently, there have been many attempts to statistically analyze Twitter data.

- Evolving Twitter data to identify trending keywords for different weekdays.
- Microblogging as a form of electronic word-of-mouth for sharing consumer opinions using brand-related hashtags.
- Sentiment identification performed in Twitter posts to identify trending sentiments about brands.

Clustering of users exploiting the dimension of time

Social data can be analyzed for automatic synthesis of user profiles.

Case study: Social Tagging Systems (STS)

Clustering of users in STS according to topics of interest and the time locality of tagging activity.

- Example: a user tagged a set of photos depicting sports is probably interested in sports. However, if his tagging activity took place during the Olympic games, maybe he is simply interested in the Olympics and is not a regular sports fan.

- Segmentation of time in frames
  - A user is related to a given tag if he has assigned at least one semantically close tag. The topic distance between two users is calculated based on the similarity of their relations to all involved tags.
  - Time-similarity between a user and a tag is calculated with the cosine coefficient.
  - The time-distance between two users is calculated considering their similarity over all timeframes.
  - Topic-based clustering with K-means, then refinement with time criterion.

References (Block 4) (1/3)

- [Quack08] Quack08, location.
- [Zhao07] Zhao07, location.
- [Koutsonikola09] Koutsonikola09, location.
References (Block 4) (2/3)


References (Block 4) (3/3)


End of Block 4

Thank you!

Questions?
Introduction and Motivation

Data streams are mostly generated or sent to resource constrained computing environments:
- Data generated on-board astronomical spacecrafts
- Data generated in sensor networks. The additional constraint that sensor nodes consume their energy rapidly with data transmission
- Data received in resource-constrained environment represents a different category of applications:
  - Personal Digital Assistants PDAs: users might request sheer amounts of data of interest to be streamed to their mobile devices. Storing and retrieving these huge amounts of data are also infeasible in such an environment

Applications

- Monitoring physiological data streams obtained from wearable sensing devices. Such monitoring can be either for:
  - applications for pervasive healthcare management,
  - Applications for seniors,
  - emergency response personnel,
  - soldiers in the battlefield
  - or athletes
- Onboard analysis of data streams: data generation would exceed the bandwidth to transfer these streams of data to ground stations for analysis. They necessitate the need for onboard analysis of data streams.

Applications

- Wearable sensors available in the market
  - SenseWear Armband from BodyMedia
  - wearable West
  - Lifespring Garment from Vivometrics
- SenseWear armband can measure heat flux, accelerometer, galvanic skin response, skin temperature, near body temperature
- Arm band can store up to about 5 days of data.
- Detecting emerging patterns in soldiers, elderly individuals, animals...

Applications

- Vehicle Data Stream Mining [Kargupta]
  - Vehicle Health Monitoring and Maintenance:
    - Detecting unusual behavior for a subsystem
  - Fuel Consumption Analysis:
    - Is the vehicle burning fuel efficiently? Identify influencing factors
    - Detect influence of driver behavior on gas mileage
  - Driver Behavior Monitoring:
    - Route monitoring: Fixed and variable routes
    - Direct Cost Issues: e.g. idling, braking habits, Safety issues
  - Vehicle location related services
  - Vehicular network security and privacy management

Computational resources affected

Stream mining affects settings in:
- memory,
- processing cycles,
- communication bandwidth and battery.
- Stream mining algorithms:
  - typically linear or sub-linear algorithms
  - characterized by being space efficient.
- BUT:
  - Most of these algorithms are not designed with regard to adaptation to resource availability
Issues (Gaber)
• the input,
• output, and
• processing settings of an algorithm
  - could be changed according to measurements of resource availability in a time frame:
  - Find:
    - resource consumption patterns and
    - algorithm settings.
• Challenge:
  • Limited computational resources
  • Limited bandwidth
  • Change of the user’s context

Approaches
1. Based on user specified quality requirements the algorithm derives resource requirements, i.e., the memory needed for managing stream approximations in order to guarantee the requested quality.
2. Input adaptation:
   • load shedding and data synopsis creation using wavelets
   • Some drawbacks:
     • solution is through a specific algorithm,
     • does not handle the situations where more than one resource is constrained, may not apply to all kinds of resources,
     • usually requires some extra processing such as sampling

Presentation Outline
• Block 1: Introduction
• Block 2: Supervised learning on streams
• Block 3: Unsupervised learning on streams
• Block 4: Mining evolving social data
• Block 5: Mining under resource constraints
  • Introduction
  • Approaches
• Block 6: Conclusions and Outlook (Ernestina Menasalvas)

Memory availability (Frakle)
2 ways of putting resource and quality awareness:
1. Claim for specific quality requirements and deduce the needed resources to achieve this quality.
2. Limit the resources provided for processing and deduce the achievable quality.
   - Q -> R ?
   - R -> Q ?

Association mining. [Nan Jiang and Le Gruenwald]
Issues to take into account:
1. Data Processing method
2. Memory management method
3. Data structure to keep itemsets
4. One pass algorithm
5. Maintenance and generation
Data Processing model

Landmark model
- mines all frequent itemsets over the entire history of stream data from a specific time point called landmark to the present.
- not suitable for applications where people are interested only in the most recent information of the data streams, such as in the stock monitoring systems.

Damped model, (Time-Fading model), mines frequent itemsets in stream data in which each transaction has a weight and this weight decreases with age.
- Older transactions contribute less weight toward itemset frequencies. Different weights for new and old transactions.
- suitable for applications in which old data has an effect on the mining results, but the effect decreases as time goes on.

Sliding Windows model finds and maintains frequent itemsets in sliding windows. Only part of the data streams within the sliding window are stored and processed at the time when the data flows in.
- The size of the sliding window may be decided according to applications and system resources.
- The mining result of the sliding window method totally depends on recently generated transactions in the range of the window.
- all the transactions in the window need to be maintained in order to remove their effects on the current mining results when they are out of range of the sliding window.

Memory management

- Classical association rule mining algorithms on static data collect the count information for all itemsets and discard the non-frequent itemsets and their count information after multiple scans of the database. When we mine association rules in stream data:
  1. there is not enough memory space to store all the itemsets and their counts when a huge amount of data comes continuously.
  2. the counts of the itemsets are changing with time when new stream data arrives. Therefore, we need to collect and store the least information possible.
- Some methods uses sizes of itemsets such as 3 or 2 to generate only this itemsets.

One Pass Algorithm to Generate Association Rules

Association rules can be found in two steps:
1. finding large itemsets (support is greater than user specified support) for a given threshold support and
2. generate desired association rules for a given confidence

Frequent itemsets

Should we use an exact or approximate algorithm to perform association rule mining in data streams?
Can its error be guaranteed if it is an approximate algorithm?
How to reduce and guarantee the error?
What is the tradeoff between accuracy and processing speed?
Is data processed within one pass?
Can this algorithm handle a large amount of data?
Up to how many frequent itemsets can this algorithm mine?
Can this algorithm handle concept drifting and how?
Association generation and maintenance [Nan Jiang and Le Gruenwald]

Mining association rules involves a lot of memory and CPU costs. This is especially a problem in data streams since the processing time is limited to one online scan. Approaches of the static world: frequent updating data stream environment, stream data are added continuously, and therefore, if we update association rules too frequently, the cost of computation will increase drastically. Some methods assume little concept drifting, that is to say the change of data distribution is relatively small.

Resources awareness

Some approaches:
- Gaber: ADG, which uses a control parameter to control its output rate according to memory, time constrains and data stream rate (see later further)
- Teng et al.: RAMIDS algorithm to not only reduce the memory required for data storage but also retain good approximation of temporal patterns given limited resources like memory space and computation power

CFI-Stream: [Nan Jiang and Le Gruenwald]

Mining closed frequent itemsets over datastreams:
- computes and maintains closed itemsets online and incrementally,
- can output the current closed frequent itemsets in real time based on users’ specified thresholds.
- time and space efficient, has good scalability as the number of transactions processed increases and adapts very rapidly to the change in datastreams.

Addition procedure of the algorithm [Nan Jiang and Le Gruenwald]

Checks if X is in the current closed itemsets set C.
- If X is in C, it updates X’s support, and for all X’s subsets Y belonging to C, it updates Y’s supports.
- Else, if X is not in C and X has been included by at least one transaction in the original transaction set, it checks whether it is a closed itemset for itself and all its subsets; and it updates the associated supports for all the closed itemsets.
  - If X is a newly arrived closed itemset and does not exist in the DIU tree, the algorithm adds it as a new node to the DIU tree.
  - Else, if X is the added transaction itself, it adds X into the closed itemset (lines 10-15); if X is the subset of added transaction, a closure checking is performed

Deletion phase of the algorithm [Nan Jiang and Le Gruenwald]

When an itemset X leaves the current sliding window CFI-Stream:
- checks if X is in the current closed itemsets set C and its count is greater or equal to two; if so, it updates X’s support and X’s subsets’ support belonging to C.
- Otherwise, it checks the itemset X and all its subsets which are in the current closed itemset C to see whether they are still closed itemsets and updates the support for all its subsets that are in the current closed itemsets
  - If the subset Y exists in transaction, Y should keep
  - Otherwise a closure check for the subset Y is performed

Performance analysis [Nan Jiang and Le Gruenwald]

Time and space efficiency independent of support information, and it can adapt to the concept-drifting in data streams. better performance than other state-of-the-art approaches in terms of both time and space overhead
- especially when the minimum support is low,
- and the dataset is dense
**COFI-tree mining** M. [El-Hajj and O.R. Zaiane]  

FP-trees used in the mining process can all fit in memory.  

COFI algorithm—as an alternative to the FP-growth algorithm—consists of three main phrases:  

- the construction of an FP-tree representing the original database,  
- The construction of a COFI-tree (Co-Occurrence Frequent Item tree) for each frequent item, and  
- the mining of frequent patterns from each COFI-tree.  

In the first phrase, a global FP-tree is constructed in the same way as in the FP-growth algorithm. Thus, 2 database scan are required—one scan for finding the frequency of each item and another scan for building the FP-tree.

**COFI tree**  

the COFI-tree contains:  

1. the item,  
2. its frequency count and  
3. its participation counter. This counter is initialized to 0, and is incremented every time the node is revisited/participated. At the end of the mining process for the COFI-tree of x, the value of this counter is equal to its frequency count. Note that the COFI algorithm requires at most two trees (i.e., the global FP-tree and the COFI-tree for a specific item) to co-exist at any time during the mining process, whereas FP-growth usually keeps more than two trees.

**Carson Kai-Sang Leung, Dale A. Brajczuk, Jialiang Yu**  

to reduce memory consumption required by FP-growth, authors replace the first step of FP-streaming by applying the COFI algorithm (instead of the FP-growth algorithm) to find “frequent” patterns.

Such a replacement of the algorithm in the first step of FP-streaming leads to both an increase and a decrease in memory consumption in different stages of the mining process. On the positive side, the use of COFI algorithm reduces memory consumption by avoiding recursive projections and constructions of FP-trees. Thus, instead of multiple FP-trees, at most two trees (namely, the global FP-tree and a COFI-tree for an item x) need to co-exist during the mining process. However, on the negative side, the construction of a COFI-tree for item x requires more memory space than the construction of an FP-tree for the same item x.

**Frequent Itemsets FP tree (Franke)**  

Han et al., 2000: FP-tree  
Giannella et al., 2003:  
- extension to mine streaming data in a time-sensitive way.  
- tilted time window tables represents window-based counts of the itemsets. Allow to maintain summaries of frequency information of recent transactions in the data at a finer granularity.  
- Update the extended FP tree in batches: accumulate incoming transactions until enough transactions of the stream have arrived. Then, the transactions of the batch are inserted into the tree.  
- Mining the tree: modification of the FP-growth algorithm that uses the tilted window table. The original approach assumes that there is enough memory available to deliver results in any required quality.

**Franke et al approach**  

1. Tries to calculate memory needed for the quality required  
2. Adapts operators to the memory available

**Franke. Adapting dynamically the tree size**  

Adapting \( \epsilon \)  
- \( f < 0.85 \): Decrease \( \epsilon \) by ten percent. Use this \( \epsilon \) when processing the following batches.  
- \( 0.85 < f < 1.0 \): The value of \( \epsilon \) remains fixed.  
- \( f > 1.0 \): Increase \( \epsilon \) by ten percent. Conduct tail pruning at the TTWTs of each node in the pattern tree and drop all nodes with empty TTWTs. Repeat these steps as long as \( f > 1.0 \).

Adapting \( \sigma \)  
- As \( \sigma \) does not influence the size of the tree directly, \( \epsilon / \sigma \) remains the same. That is why the user does not provide a fixed value of \( \sigma \), but rather claims for a certain \( \epsilon / \sigma \) that should be guaranteed. Again, the user may also specify a lower bound for the value of \( \sigma \).
Orthogonal decision trees (ODTs) offer an effective way to construct a redundancy-free, accurate, and meaningful representation of large decision-tree-ensembles:

1. First construct an algebraic representation of trees using multivariate discrete Fourier bases.
2. The new representation is then used for eigen-analysis of the covariance matrix generated by the decision trees in Fourier representation. Converts the corresponding principal components to decision trees.
3. These trees are functionally orthogonal to each other and they span the underlying function space. These orthogonal trees are in turn used for accurate (in many cases with improved accuracy) and redundancy-free (in the sense of orthogonal basis set) compact representation of large ensembles.
ODT. Kargupta: Application wearable

Variables sensed:
- Heat flux: The amount of heat dissipated by the body.
- Accelerometer: Motion of the body
- Galvanic Skin Response: Electrical conductivity between two points on the wearer’s arm
- Skin Temperature: Temperature of the skin and is generally reflective of the body’s core temperature
- Near-Body Temperature: Air temperature immediately around the wearer’s armband.

Experiments with ODT [H. Kargupta]

construction of ODTs using four C4.5 trees reports the structure of an ODT obtained by projecting the trees along the first principle component.

aggregated orthogonal decision trees have accuracy comparable to that of large Bagging ensembles.

an aggregated ODT is a good solution for classification problems on PDAs, pocket PCs or cell-phones

Performance analysis in ODT [H. Kargupta]

In resource constrained environments it is often necessary to keep track of the amount of memory used to store the ensemble.

The current implementation storing a node data structure in a tree requires approximately 1 KB of memory.

Consider an ensemble of 20 trees. If the average number of nodes in the trees in the Bagging ensemble is 7, then we are required to store 140 KB of data.

Orthogonal decision trees on the other hand are smaller in size, with less redundancy. In the experiments they typically have a complexity of 3 nodes. This means that to store only 60 KB of Data are required

ODT - resource constraint [H. Kargupta]

-Tree Complexity Ratio (TCR): total number of nodes in the ODT versus the total number of nodes in the Bagging ensemble

-It may be noted that in resource constrained environments one can opt for meaningful trees of smaller size and comparable accuracy as opposed to larger ensembles with a slightly better accuracy

ODT error in classification [H. Kargupta]
Response time comparison [H. Kargupta]
on a pocket PC using a Bagging ensemble and an equivalent orthogonal decision tree ensemble

Response time (ms) versus number of trees in ensemble

Presentation Outline
- Block 1: Introduction
- Block 2: Supervised learning on streams
- Block 3: Unsupervised learning on streams
- Block 4: Mining evolving social data
- Block 5: Mining under resource constraints
  - Introduction
  - Approaches: Classification, Clustering and Association
- Block 6: Conclusions and Outlook

Gaber-AOG (Algorithm Output Granularity)
Adapting the algorithm output according to:
- resource availability and
- data stream generation/input rate.
The AOG approach is based on the following axioms:
1. The algorithm output rate (AR) is function in a data rate (DR),
   \[ AR = f(DR) \]
2. The time needed to fill the available memory by the algorithm results (TM) is function in (AR)
   \[ TM = f(AR) \]
3. The algorithm accuracy (AC) is function in (TM)
   \[ AC = f(TM) \]

Gaber-AOG (cont.)
- three-stage,
- resource-aware
- distance-based mining data streams approach

The process of mining the data stream (Gaber)
1. Determine the frequency of adaptation and mining.
2. According to the data rate, calculate the algorithm output rate and the algorithm threshold.
3. Mine the incoming stream using the calculated algorithm threshold.
4. Adjust the threshold after a time frame to adapt with the change in the data rate using linear regression
5. Repeat steps 3 and 4 till the algorithm lasts the time interval threshold.
6. Perform knowledge integration of the results.
AOG based algorithms

- **LightWeight Clustering (LWC):** the threshold is used to specify the minimum distance between the cluster center and the data element/record;
- **LightWeight Classification (LWClass):** In addition of using the threshold in specifying the distance, the class label is checked. If the class label of the stored items and the new item that are similar (within the accepted distance) is the same, the weight of the stored item is increased along with the weighted average of the other attributes, otherwise the weight is decreased and the new item is ignored;
- **LightWeight Frequent patterns (LWF):** the threshold is used to determine the number of counters for the heavy hitters.

*Resource aware architecture (RASA)*

Comparison of K-means and LWC (gaber)

- LWC outperforms K-means even with the fine threshold that leads to creating large number of clusters
- The generated centers for both algorithms are very close and have the same trend
- The unpredictability of the number of passes needed by K-means leads to fluctuating running time with similar data set sizes.
- Experiment with an increase in the dataset sizes to show stability in the AOG overhead.

Granularity-based Approach [Gaber]

Combining the three possible granularity-based adaptation, namely:
1. AIG: Algorithm Input Granularity
2. AOG: Algorithm Output Granularity
3. APC: Algorithm Processing Granularity

Granularity Based Algorithms [Gaber]

Clustering:
- Light-Weight Clustering
- RA-Cluster
- DRA-Cluster
- RA-VFKM

Change Detection:
- CHANGE-DETECT

Classifiers:
- Light-Weight Class (LWClass)
- RA-Class
- DRA-Class

Time Series Analysers:
- RA-SAX
- Frequent Items and Associations:
  - LWF (Light-Weight Frequent Items)
  - HiCoRE (Highly Correlated Energy-Efficient Rules)

Gaber. Resource aware-Clustering

Enables resource-awareness in streaming computation using algorithm granularity settings changes the resource consumption patterns periodically. Framework is applied to a novel threshold-based microclustering algorithm to test its validity and feasibility.

RA-Cluster:
- The first stream clustering algorithm that can adapt to the changing availability of different resources.
- The experimental results showed the applicability of the framework and the algorithm in terms of resource-awareness and accuracy.

Algorithm settings:

- Input Granularity (AIG):
  - Sampling, load shedding, and creating data synopsis techniques.
- Algorithm Output Granularity (AOG):
  - Number of knowledge structures created or level of output granularity.
- Algorithm Processing Granularity (APG): changing the processing settings of the algorithm to consume smaller amount of resources according to consumption measures over the last frame:
  - Error rate of approximation algorithms.
Algorithm settings (II)

Algorithm Output Granularity (AOG):
- number of knowledge structures:
  - number of clusters or rules.
- The output size could be changed also using level of output granularity which means the less detailed output, the higher the granularity and vice versa.

Algorithm Processing Granularity (APG):
- Randomization and approximation techniques

The process of enabling resource awareness should be very lightweight in order to be feasible in a streaming environment characterized by its scarcity of resources.

System architecture [Gaber]

RA-cluster algorithm

RA-Cluster: combines resource-awareness, adaptation and real-time all in a holistic approach.

Process:
1. starts with using an initial threshold to run the algorithm and after a fixed time frame the resource consumption patterns of the CPU, memory, and battery given that we run in a resource constrained environment is assessed.
2. According to the above assessment, the algorithm settings are changed to cope with the data rate.

RA-Cluster is an incremental online micro-clustering algorithm that has all the required parameters to enable resource-awareness

Some results [Gaber]

AG main lacks [Gaber]

AG model has proven its applicability to change the resource consumption, but:
- The bounds over the AG settings have no guarantee over the quality of the output, quality relies on other interleaving factors such as data distribution and the running mining technique.
- changes in the AG settings are not quality-aware, the algorithm changes according only to the availability of computational resources.
- loss in accuracy: in some cases, we can gain the same accuracy using less resources.
- The AG settings do not take into consideration the interaction among the different settings. Addressing this issue can optimize the use of resources.
Quality assurance (Gaber, Franke, Karnstedt)

Generic 3 layer model for quality guaranteed resource-aware (QGRA) data mining on data streams.

applicable to a wide variety of stream mining techniques.

The model bridges the gap between quality aware mining and general resource-adaptivity by monitoring the resource consumption.

Quality model

To control the adaptation the algorithm uses:
1. Function to determine the algorithmic parameters from the assessed resources.
2. Function to determine the output quality based on the chosen algorithmic parameters.
3. Functions to compute lower bounds for the algorithmic parameters in order to maintain the quality of the output.

Different quality measures (Gaber, Franke, Karnstedt)

Different quality measures

1. Methodological quality \( Q_M \)
2. Temporal quality \( Q_T \)
3. Quality of appreciation \( Q_A \)
4. Quality of interestingness \( Q_I \)
5. Time range \( Q_{t_r} \)
6. Reactivity \( Q_{r} \)

The three layers [Gaber, Franke, Karnstedt]

1. Resource monitoring: works over dynamic time intervals. Dynamic time window that changes according to the criticality of the available computational resources.
2. Real-time quality assessment: Able to provide information about the quality of the output given the availability of resources. Also provides the system with information about preserving computational resources while maintaining the same quality.
3. (AGS) component: feeds the mining algorithms with input, output and processing settings.

QRSA algorithm

QRSA algorithm:
1. \( B_t = \text{resource monitoring} \)
2. \( F_t = \text{dynamically provided resource limits,} \)
3. \( P = \text{adaptation factors} \)
4. \( Q = \text{algorithmic parameters} \)
5. \( P_t = \text{parameters} \)
6. \( Q_t = \text{quality parameters} \)
7. \( Q_{t,r} = \text{time range parameters} \)
8. \( R = \text{residual resources} \)
9. \( S = \text{set of resources} \)
10. \( \text{adaptation takes place} \)
11. \( \text{query each of these independent sections separately.} \)

Requirements of the algorithms

- Parameters must exist in the algorithm, a strong correlation between the adaptation factors and the algorithm's resource requirements helps estimating the quality of the output.
- The algorithm must show homogeneous behavior
- Stream with homogeneous properties maintained
- Able to establish a partitioning into independent sections in the mining result. Thus different values of the adaption factors only have “local” effects in the mining results.
- Possible to query each of these independent sections separately.
End of Block 5

Thank you!

Questions?

References (Block 5) (1/5)
Marcel Karnstedt, Conny Franke, Mohamed Medhat Gaber: A Model for Quality Guaranteed Resource-Aware Stream Mining, Fifth International Workshop on Knowledge Discovery from Ubiquitous Data Streams icw ECML,PKDD’07, Warsaw, Poland, Sep 2007, pp. 71–82

Conny Franke, Marcel Karnstedt, Kai-Uwe Sattler: Mining Data Streams under Dynamically Changing Resource Constraints, KDML 2006: Knowledge Discovery, Data Mining, and Machine Learning, October 2006, Hildesheim, Germany, pp. 262–269

Conny Franke, Michael Hartung, Marcel Karnstedt, Kai-Uwe Sattler: Quality-Aware Mining of Data Streams, 4th International Conference on Information Quality (ICIQ) 2005, Boston, USA

References (Block 5) (2/5)


References (Block 5) (3/5)

Haghighi P. D., Gaber M. M., Krishnaswamy S., Zaslavsky A, and Loke S. W., An Architecture for Context-Aware Adaptive Data Stream Mining, Proceedings of the International Workshop on Knowledge Discovery from Ubiquitous Data Streams (MKDUDS07), in conjunction with ECML and PKDD 2007, September 17, Warsaw, Poland, 2007

Phung N. D., Gaber M. M., and Rohm U., Resource-aware Distributed Online Data Mining for Wireless Sensor Networks, Proceedings of the International Workshop on Knowledge Discovery from Ubiquitous Data Streams (MKDUDS07), in conjunction with ECML and PKDD 2007, September 17, Warsaw, Poland, 2007

References (Block 5) (4/5)
Gaber, M. M., Krishnaswamy S., and Zaslavsky, A., Resource-Aware Mining of Data Streams, Journal of Universal Computer Science, Vol. 11, No. 8 (2005), pp. 1440–1453, ISSN 0948-6958, Special Issue on Knowledge Discovery in Data Streams, Jesus S. Aguilar-Ruiz and Joao Gama Eds.) , Verlag der Technischen Universitat Graz, Know-Center Graz, Austria, August 2005.

Hamed Chok, Le Gruenwald: Spatio-temporal association rule mining framework for real-time sensor network applications. CIKM 2009: 1761–1764


Wei-Guang Teng, Ming-Syan Chen, and Philip S. Yu: Resource-Aware Mining with Variable Granularities in Data Streams. SIAM Int’l Conf. on Data Mining. 2004

References (Block 5) (5/5)


Presentation Outline

- Block 1: Introduction
- Block 2: Supervised learning on streams
- Block 3: Unsupervised learning on streams
- Block 4: Mining evolving social data
- Block 5: Mining under resource constraints
- Block 6: Conclusions and Outlook

Outlook I

- Learning on complex stream data
  - Multiple, interrelated and interdependent streams
  - Data that involve static objects & streams feeding them
  - Probabilistic models and tensor-based learning
  - Object profiling: A model for all data or a model for each object (or both)?

Challenges of new applications
- Data from mobile devices
- Data from devices that are available irregularly
- Learning using limited computational resources
- Learning under time constraints
- Capturing context

Outlook II – Old challenges, not yet solved

- Dealing with time: Multi-horizon and multi-granularity analysis
- Scalability for large data volumes
  - Decrease complexity
  - Increase parallelism
- Robustness, esp. if the learners are complex
- Learning for online or real-time applications
- Visualization
  - Visualization of WHAT?
The streams, the objects, the models
  - Visualization for WHO?
The expert, the data owner, the decision maker, the casual observer
- Evaluation: Measures & Benchmarks

Outlook II – Old challenges, not yet solved

- Supervised learning for evolving data
  - Dealing with emerging and evolving concepts
  - Delayed labeling
  - Label availability
  - Cost-benefit trade-off of the model update
  - Change description
  - Predicting re-occurring contexts
  - Multi-label prediction
  - Reliability and uncertainty

Outlook II – Old challenges, not yet solved

- Unsupervised learning for evolving data
  - Dealing with emerging and evolving concepts
  - Waiving the assumptions about the data generating process (or verifying them)
  - Change description
  - Setting up the learner:
    - What is the influence of parameter x, given that there is no ground truth in the data?
    - If Gibbs sampling, then on what sample?
  - Evaluation

Outlook – Challenges, amplified by evolving social/data

- Community identification in evolving social data
  - Exploitation of evolving communities
  - Event prediction
  - Opinion mining

- Visualization of community evolution
  - Human-friendly
  - Demonstrating the role of communities and the impact of change in a comprehensible way
  - Application-specific
- Supporting online applications
  - Recommendation engines
Thank you very much!

QUESTIONS