Workshop on Ubiquitous Data Mining

in conjunction with the
23rd. International Joint Conference on Artificial Intelligence (IJCAI 2013)
Beijing, China
August 3 - 9, 2013

Edited by

João Gama, Michael May, Nuno Marques and Paulo Cortez
Preface

Ubiquitous Data Mining (UDM) uses Data Mining techniques to extract useful knowledge from data, namely when its characteristics reflect a World in Movement. The goal of this workshop is to convene researchers (from both academia and industry) who deal with techniques such as: decision rules, decision trees, association rules, clustering, filtering, learning classifier systems, neural networks, support vector machines, preprocessing, postprocessing, feature selection and visualization techniques for UDM of distributed and heterogeneous sources in the form of a continuous stream with mobile and/or embedded devices and related themes.

This is the third workshop in the topic. We received 12 submissions that were evaluated by 3 members of the Program Committee. The PC recommended accepting 8 full papers and 2 Position Papers. We have a diverse set of papers focusing from activity recognition, predicting taxis demand, trend mining to more theoretical aspects of learning model rules from data streams. All papers deal with different aspects of evolving data and/or distributed data.

We would like to thank all people that make this event possible. First of all, we thank authors that submit their work and the Program Committee for the work in reviewing the papers, and proposing suggestions to improve the works. A final Thanks to the IJCAI Workshop Chairs for all the support.

João Gama, Michael May, Nuno Marques and Paulo Cortez

Program Chairs
Organization

Program Chairs:
João Gama, Michael May, Nuno Marques and Paulo Cortez

Organizing Chairs:
Manuel F. Santos, Pedro P. Rodrigues and Albert Bifet

Publicity Chair:
Carlos A. Ferreira

Organized in the context of the project Knowledge Discovery from Ubiquitous Data Streams (PTDC/EIA-EIA/098355/2008). This workshop is funded by the ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by the Portuguese Government Funds through the FCT (Portuguese Foundation for Science and Technology).

Program Committee:
- Albert Bifet, Technical University of Catalonia, Spain
- Alfredo Cuzzocrea, University of Calabria, Italy
- André Carvalho, University of São Paulo (USP), Brazil
- Antoine Cornuéjols, LRI, France
- Carlos A. Ferreira, Institute of Engineering of Porto, Portugal
- Eduardo Spinosa, University of São Paulo (USP), Brazil
- Elaine Sousa, University of São Paulo, Brazil
- Elena Ikonomovska, University St. Cyril & Methodius, Macedonia
- Ernestina Menasalvas, Technical University of Madrid, Spain
- Florent Masseglia, INRIA, France
- Geoffrey Holmes, University of Waikato, New Zealand
- Hadi Tork, LIAAD-INESC TEC, Portugal
- Jesús Aguilar, University of Pablo Olavide, Spain
- Jiong Yang, Case Western Reserve University, USA
- João Gama, University of Porto, Portugal
• João Gomes, I2R A*Star, Singapore
• João Mendes Moreira, University of Porto, Portugal
• José Ávila, University of Malaga, Spain
• Josep Roure, CMU, USA
• Manuel F. Santos, University of Minho, Portugal
• Mark Last, Ben Gurion University, Israel
• Matjaž Gams, Jožef Stefan Institute, Slovenia
• Michael May, Fraunhofer Bonn, Germany
• Min Wang, IBM, USA
• Miroslav Kubat, University of Miami, USA
• Mohamed Gaber, University of Plymouth, UK
• Myra Spiliopoulou, University of Magdeburg, Germany
• Nuno Marques, University of Nova Lisboa, Portugal
• Paulo Cortez, University of Minho, Portugal
• Pedro P. Rodrigues, University of Porto, Portugal
• Philip Yu, IBM Watson, USA
• Raquel Sebastião, University of Porto, Portugal
• Rasmus Pederson, Copenhagen Business School, Denmark
• Ricard Gavaldà, University of Barcelona, Spain
• Shonali Krishnaswamy, Monash University, Australia
• Xiaoyang S. Wang, University of Vermont, USA
• Ying Yang, Monash University, Australia
# Table of Contents

## Invited Talks

Exploiting Label Relationship in Multi-Label Learning .......................... 1  
*Zhi-Hua Zhou*

NIM: Scalable Distributed Stream Processing System on Mobile Network Data ........................................... 3  
*Wei Fan*

## Papers

Predicting Globally and Locally: A Comparison of Methods for Vehicle Trajectory Prediction .......................... 5  
*William Groves, Ernesto Nunes and Maria Gini*

Learning Model Rules from High-Speed Data Streams ...................... 10  
*Ezilda Almeida, Carlos A. Ferreira and João Gama*

On Recommending Urban Hotspots to Find Our Next Passenger .......... 17  
*Luis Moreira-Matias, Ricardo Fernandes, João Gama, Michel Ferreira, João Mendes-Moreira and Luis Damas*

*Wentao Fan and Nizar Bouguila*

Simultaneous segmentation and recognition of gestures for human-machine interaction ........................................ 29  
*Harold Vasquez, Luis E. Sucar and Hugo J. Escalante*

Cicerone: Design of a Real-Time Area Knowledge-Enhanced Venue Recommender ............................................... 34  
*Daniel Villatoro, Jordi Aranda, Marc Planagumà, Rafael Gimenez and Marc Torrent-Moreno*

Road-quality classification and bump detection with bicycle-mounted smartphones ............................................... 39  
*Marius Hoffmann, Michael Mock and Michael May*

Simulating Price Interactions by Mining Multivariate Financial Time Series .................................................. 44  
*Bruno Silva, Luis Cavique and Nuno Marques*

Trend template: mining trends with a semi-formal trend model ........ 49  
*Olga Streibel, Lars Wißler, Robert Tolksdorf and Danilo Montesi*

Ubiquitous Self-Organizing Maps ................................................. 54  
*Bruno Silva and Nuno Marques*
Exploiting Label Relationship in Multi-Label Learning

Zhi-Hua Zhou

National Key Laboratory for Novel Software Technology,
Nanjing University, China
zhouzh@nju.edu.cn

Abstract

In many real data mining tasks, one data object is often associated with multiple class labels simultaneously; for example, a document may belong to multiple topics, an image can be tagged with multiple terms, etc. Multi-label learning focuses on such problems, and it is well accepted that the exploitation of relationship among labels is crucial; actually this is the essential difference between multi-label learning and conventional (single-label) supervised learning.

Most multi-label learning approaches try to capture label relationship and then apply it to help construct prediction models. Some approaches rely on external knowledge resources such as label hierarchies, and some approaches try to exploit label relationship by counting the label co-occurrences in training data. These approaches are effective in many cases; however, in real practice, the external label relationship is often unavailable, and generating label relationship from training data and then applying to the same training data for model construction will greatly increase the overfitting risk. Moreover, the label relationship is usually assumed symmetric, and almost all existing approaches exploit it globally by assuming the label correlation be shared among all instances.

Short Bio

Zhi-Hua Zhou is a professor at Nanjing University. His research interests are mainly in machine learning, data mining, pattern recognition and multimedia information retrieval. In these areas he has published more than 100 papers in leading international journals or conferences, and holds 12 patents. He is the recipient of the IEEE CIS Outstanding Early Career Award, the Fok Ying Tung Young Professorship Award, the Microsoft Young Professorship Award,
the National Science & Technology Award for Young Scholars of China, and many other awards including nine international journal/conference paper or competition awards. He is an Associate Editor-in-Chief of "Chinese Science Bulletin", Associate Editor or Editorial Boards member of "ACM Trans. Intelligent Systems and Technology" and twelve other journals. He is the Founder and Steering Committee Chair of ACML, and Steering Committee member of PAKDD and PRICAI. He is the Chair of the AI&PR Technical Committee of the China Computer Federation, Chair of the Machine Learning Technical Committee of the China Association of AI, the Vice Chair of the Data Mining Technical Committee of the IEEE Computational Intelligence Society, and the Chair of the IEEE Computer Society Nanjing Chapter. He is a Fellow of the IAPR, Fellow of the IEEE, and Fellow of the IET/IEE.
NIM: Scalable Distributed Stream Processing System on Mobile Network Data

Wei Fan

IBM T.J. Watson Research, Hawthorne, NY, USA
weifan@us.ibm.com

Abstract

As a typical example of New Moore's law, the amount of 3G mobile broadband (MBB) data has grown from 15 to 20 times in the past two years (30TB to 40TB per day on average for a major city in China), real-time processing and mining of these data are becoming increasingly necessary. The overhead of storage and file transfer to HDFS, delay in processing, etc are making offline analysis on these datasets obsolete. Analysis of these datasets are non-trivial, examples include mobile personal recommendation, anomaly traffic detection, and network fault diagnosis. In this talk, we describe NIM - Network Intelligence Miner. NIM is a scalable and elastic streaming solution that analyzes MBB statistics and traffic patterns in real-time and provides information for real-time decision making. The accuracy of statistical analysis and pattern recognition of NIM is identical to that of off line analysis, while NIM can process data at line rate. The design and the unique features (e.g., balanced data grouping, aging strategy) of NIM will be helpful not only for the network data analysis but also for other applications.

Short Bio

Dr. Wei Fan is the associate director of Huawei Noah's Ark Lab. Prior to joining Huawei, he received his PhD in Computer Science from Columbia University in 2001 and had been working in IBM T.J. Watson Research since 2000. His main research interests and experiences are in various areas of data mining and database systems, such as, stream computing, high performance computing, extremely skewed distribution, cost-sensitive learning, risk analysis, ensemble methods, easy-to-use nonparametric methods, graph mining, predictive feature discovery, feature selection, sample selection bias, transfer learning, time series
analysis, bioinformatics, social network analysis, novel applications and commercial data mining systems. His co-authored paper received ICDM'2006 Best Application Paper Award; he lead the team that used Random Decision Tree to win 2008 ICDM Data Mining Cup Championship. He received 2010 IBM Outstanding Technical Achievement Award for his contribution to IBM InfoSphere Streams. He is the associate editor of ACM Transaction on Knowledge Discovery and Data Mining (TKDD).
Predicting Globally and Locally: A Comparison of Methods for Vehicle Trajectory Prediction

William Groves, Ernesto Nunes, and Maria Gini
Department of Computer Science and Engineering, University of Minnesota
{groves, enunes, gini}@cs.umn.edu

Abstract

We propose eigen-based and Markov-based methods to explore the global and local structure of patterns in real-world GPS taxi trajectories. Our primary goal is to predict the subsequent path of an in-progress taxi trajectory. The exploration of global and local structure in the data differentiates this work from the state-of-the-art literature in trajectory prediction methods, which mostly focuses on local structures and feature selection. We propose four algorithms: a frequency-based algorithm FreqCount, which we use as a benchmark, two eigen-based (EigenStrat, LapStrat), and a Markov-based algorithm (MCStrat). Pairwise performance analysis on a large real-world data set reveals that LapStrat is the best performer, followed by MCStrat.

1 Introduction

In order to discover characteristic patterns in large spatio-temporal data sets, mining algorithms have to take into account spatial relations, such as topology and direction, as well as temporal relations. The increased use of devices that are capable of storing driving-related spatio-temporal information helps researchers and practitioners gather the necessary data to understand driving patterns in cities, and to design location-based services for drivers. To the urban planner, the work can help to aggregate driver habits and uncovery alternative routes that could help alleviate traffic. Additionally, it also helps prioritize the maintenance of roads.

Our work combines data mining techniques that discover global structure in the data, and local probabilistic methods that predict short-term routes for drivers, based on past driving trajectories through the road network of a city.

The literature on prediction has offered Markov-based and other probabilistic methods that predict paths accurately. However, most methods rely on local structure of data, and use many extra features to improve prediction accuracy. In this paper we use only the basic spatio-temporal data stream. We advance the state-of-the-art by proposing the LapStrat algorithm. This algorithm reduces dimensionality and clusters data using spectral clustering to then predict a subsequent path using a Bayesian network. Our algorithm supports global analysis of the data, via clustering, as well as local inference using the Bayesian framework. In addition, since our algorithm only uses location and time data, it can be easily generalized to other domains with spatio-temporal information. Our contributions are summarized as follows:

1. We offer a systematic way of extracting common behavioral characteristics from a large set of observations using an algorithm inspired by principal component analysis (EigenStrat) and our LapStrat algorithm.
2. We compare the effectiveness of methods that explore global structure only (FreqCount and EigenStrat), local structure only (MCStrat), and mixed global and local structure (LapStrat). We show experimentally that LapStrat offers competitive prediction power compared to the more local structure-reliant MCStrat algorithm.

2 Related Work

Eigendecomposition has been used extensively to analyze and summarize the characteristic structure of data sets. The structure of network flows is analyzed in [Lakhina et al., 2004], principal component analysis (PCA) is used to summarize the characteristics of the flows that pass through an internet service provider. [Zhang et al., 2005] identify two weaknesses that make PCA less effective on real-world data, i.e. sensitivity to outliers in the data, and concerns about its interpretation, and present an alternative, Laplacian eigenanalysis. The difference between these methods is due to the set of relationships each method considers: the Laplacian matrix only considers similarity between close neighbors, while PCA considers relationships between all pairs of points. These studies focus on the clustering power of the eigen-based methods to find structures in the data. Our work goes beyond summarizing the structure of the taxi routes, and uses the eigenanalysis clusters to predict the subsequent path of an in-progress taxi trajectory.

Research in travel prediction based on driver behavior has enjoyed some recent popularity. [Krumm, 2010] predicts the next turn a driver will take by choosing with higher likelihood a turn that links more destinations or is more time efficient. [Ziebart et al., 2008] offer algorithms for turn prediction, route prediction, and destination prediction. The study uses a Markov model representation and inverse reinforcement learning coupled with maximum entropy to provide ac-
Our work addresses this shortcoming by providing both an analysis of commuting patterns, using eigenanalysis, and route prediction based on partial trajectories.

### 3 Data Preparation

The GPS trajectories we use for our experiments are taken from the publicly available Beijing Taxi data set which includes 1 to 5-minute resolution location data for over ten-thousand taxis for one week in 2009 [Yuan et al., 2010]. Beijing, China is reported to have seventy-thousand registered taxis, so this data set represents a large cross-section of all taxi traffic for the one-week period [Zhu et al., 2012].

Because the data set contains only location and time information of each taxi, preprocessing the data into segments based on individual taxi fares is useful. The data has sufficient detail to facilitate inference on when a taxi ride is completed: for example, a taxi waiting for a fare will be stopped at a taxi stand for many minutes [Zhu et al., 2012]. Using these inferences, the data is separated into taxi rides.

To facilitate analysis, the taxi trajectories are discretized into transitions on a region grid with cells of size $1.5 \times 1.5$ km square. $V = \{v_1, v_2, \ldots, v_m\}$ is a collection of trajectories. We divide it into $V_{TR}$, $V_{TE}$, $V_{VA}$ which are the training, test, and validation sets, respectively. A trajectory $v_i$ is a sequence of $N$ time-ordered GPS coordinates: $v_i = <c_{i1}, \ldots, c_{iN}>$. Each coordinate contains a GPS latitude and longitude value, $c_{ij} = (x_j, y_j)$. Given a complete trajectory $(v_i)$, a partial trajectory (50% of a full trajectory) can be generated as $v_{i\text{partial}} = <c_{i1}, c_{i2}, \ldots, c_{iN/2}>$. The last location of a partial trajectory $v_{i\text{last}} = <c_{iN/2}>$ is used to begin the prediction task.

The relevant portion of the city’s area containing the majority of the city’s taxi trips, called a *city grid*, is enclosed in a matrix of dimension $17 \times 20$. Each $s_i$ corresponds to the center of a grid square in the euclidean $xy$-space. The city graph is encoded as a rectilinear grid with directed edges $(e_{s_i,s_j})$ between adjacent grid squares. $\pi(s_i, s_j)$ is an indicator function that returns 1 if GPS coordinate $c_j$ is closer to grid center $s_i$ than to any other grid center and otherwise returns 0. Equation 1 shows an indicator function to determine if two GPS coordinates indicate traversal in the graph.

\[
\Phi(c_j, c_k, s_i, s_m) = \begin{cases} 
1, & \text{if } (\pi(c_j, s_i) \cap \pi(c_k, s_m)) = 1 \\
0, & \text{Otherwise}
\end{cases}
\]

From trajectory $v_i$, a policy vector $\pi_i$ is created having one value for each edge in the city grid. Each $\delta_{s_i,s_m}$ is a directed edge coefficient indicating that a transition occurred between $s_i$ and $s_m$ in the trajectory. The policy vectors for this data set graph have length $(|\pi|)$ of 1286, based on the number of edges in the graph. A small sample city grid is in Figure 1. A collection of policies $\Pi = \{\pi_1, \pi_2, \ldots, \pi_w\}$ is computed from a collection of trajectories $V$:

\[
\pi_{vi} = <\delta_{v_{i1},v_{i2}}, \ldots, \delta_{v_{iN},v_{iN+1}>,}
\]

\[
\delta_{v_{iN},v_{iN+1}} = \begin{cases} 
1, & \text{if } \sum_{j=1}^{N-1} \Phi(c_{vi,j}, c_{vi,j+1}) \geq 1 \\
0, & \text{Otherwise}
\end{cases}
\]

A graphical example showing a trajectory converted into a policy is shown in Figure 2. All visited locations for trajectory $v_i$ are shown in the city grid discretization.

![Figure 1: City grid transitions are all rectilinear.](image1.png)

![Figure 2: A trajectory converted to a policy in the city grid.](image2.png)

### 4 Methods

This work proposes four methods that explore either the local or the global structure or a mix of both to predict short-term trajectories for drivers, based on past trajectories.
The probability of a transition \( (s_i \rightarrow s_j) \) is computed as the count of the transition \( s_i \rightarrow s_j \) in \( V_{TR} \) divided by the count of all transitions exiting \( s_i \) in \( V_{TR} \).

Policy iteration (Algorithm 1) is applied to the last location of a partial trajectory using the frequency count policy set \( \Pi_{FreqCount} = \langle \pi^{FreqCount} \rangle \) to determine a basic prediction of future actions. This method only considers frequency of occurrence for each transition in the training set, so it is expected to perform poorly in areas where trajectories intersect.

**Algorithm 1: Policy Iteration**

**Input:** Location vector with last location of taxi \( \theta^{last} \), a policy list \( \Pi \), prediction horizon \( niter \)

**Output:** A location vector containing visit probabilities for future locations \( \hat{\theta} \)

1. \( \theta_{accum} \leftarrow \theta^{last} \)
2. for \( \pi \in \Pi \) do
3. \( t \leftarrow 1 \)
4. \( \theta^t \leftarrow \theta^{last} \)
5. while \( t \leq niter \) do
6. \( \theta^t = \langle \omega^t, \omega^t, \ldots, \omega^t \rangle \)
7. \( t \leftarrow t + 1 \)
8. for \( S_i \in S \) do
9. \( \omega^t_{accum} \leftarrow \max(\omega^t_{accum}, \omega^t) \)
10. end for
11. \( \hat{\theta} = \theta_{accum} \)

**EigenStrat: Eigen Analysis of Covariance.** This method exploits linear relationships between transitions in the grid which 1) can be matched to partial trajectories for purposes of prediction and 2) can be used to study behaviors in the system. The first part of the algorithm focuses on model generation. For each pair of edges, the covariance is computed using the training set observations. The \( n \) largest eigenvectors are computed from the covariance matrix. These form a collection of characteristic eigen-strategies from training data.

When predicting for an in-progress trajectory, the algorithm takes the policy generated from a partial taxi trajectory \( \pi^{\text{predict}} \), a maximum angle to use as the relevancy threshold \( \alpha \), and the eigen-strategies as \( \Pi \). Eigen-strategies having an angular distance less than \( \alpha \) to \( \pi^{\text{predict}} \) are added to \( \Pi_{rel} \). This collection is then used for policy iteration. Optimal values for \( \alpha \) and \( dims \) are learned experimentally.

Eigenpolicies also facilitate exploration of strategic decisions. Figure 7 shows an eigenpolicy plot with a distinct pattern in the training data. Taxis were strongly confined to trajectories either the inside circle or the perimeter of the circle.
Algorithm 2: EigenStrat

**Input:** $\Pi_{TR}$, number of principal components ($\text{dims}$), minimum angle between policies ($\alpha$), prediction horizon ($\text{horizon}$), partial policy ($\pi^\text{partial}_i$)

**Output:** Inferred location vector $\hat{\theta}$

1. Generate covariance matrix $C_{\Pi_{TR} \times |\pi|}$ (where $\pi_i \in \Pi_{TR}$) between transitions on the grid.
2. Get the $\text{dims}$ eigenvectors of $C$ with largest eigenvalues.
3. Compute cosine similarity between $\pi^\text{partial}_i$ and the principal components ($\pi_j$, $j = 1 \ldots \text{dims}$):
   \[
   \Pi_{\text{rel}} = \{ \pi_j | \cos(\pi_j, \pi^\text{partial}_i) > \alpha \}
   \]
4. If $\cos(\pi_j, \pi^\text{partial}_i) < 0$, then flip the sign of the coefficients for this eigensimilarity. Use Algorithm 1 with $\Pi_{\text{rel}}$ on $\pi^\text{partial}$ for $\text{horizon}$ iterations to compute $\hat{\theta}$.

but rarely between these regions. The two series (positive and negative) indicate the sign and magnitude of the grid coefficients for this eigenvector. We believe analysis of this type has great promise for large spatio-temporal data sets.

![Figure 7: An eigenpolicy showing a strategic pattern.](image)

**Algorithm 3: LapStrat**

**Input:** $\Pi_{TR}$, dimension ($\text{dims}$), number of clusters ($k$), similarity threshold ($\epsilon$), prediction (horizon), partial policy ($\pi^\text{partial}_i$)

**Output:** Inferred location vector $\hat{\theta}$

1. Generate similarity matrix $W_{|\Pi_{TR}| \times |\pi|}$ where
   \[
   w_{ij} = \begin{cases} 
   J(\pi_i, \pi_j), & \text{if } J(\pi_i, \pi_j) \geq \epsilon \\
   0 & \text{Otherwise}
   \end{cases}
   \]
2. Generate Laplacian ($L$): $L = D - W$ and $\forall d_{ij} \in D$
   \[
   d_{ij} = \begin{cases} 
   \sum_{z=1}^{\Pi_{TR}} w_{i,z}, & \text{if } i = z \\
   0 & \text{Otherwise}
   \end{cases}
   \]
3. Get the $\text{dims}$ eigenvectors with smallest eigenvalues.
4. Use $k$-means to find the mean centroids ($\pi_j$, $j = 1 \ldots k$) of $k$ policy clusters.
5. Find all centroids similar to $\pi^\text{partial}_i$:
   \[
   \Pi_{\text{rel}} = \{ \pi_j | J(\pi_j, \pi^\text{partial}_i) > \epsilon \}
   \]
6. Use Algorithm 1 with $\Pi_{\text{rel}}$ on $\pi^\text{partial}_i$ for $\text{horizon}$ iterations to compute $\hat{\theta}$.

**LapStrat: Spectral Clustering-Inspired Algorithm.** LapStrat (Algorithm 3) combines spectral clustering and Bayesian-based policy iteration to cluster policies and infer driver next turns. Spectral clustering operates upon a similarity graph and its respective Laplacian operator. This work follows the approach of [Shi and Malik, 2000] using an unnormalized graph Laplacian. We use Jaccard index to compute the similarity graph between policies. We chose the Jaccard index, because it finds similarities between policies that are almost parallel. This is important in cases such as two highways that only have one meeting point; in this case, if the highways are alternative routes to the same intersection, they should be similar with respect to the intersection point. The input to the Jaccard index are two vectors representing policies generated in Section 3. $J(\pi_i, \pi_j)$ is the Jaccard similarity for pair $\pi_i$ and $\pi_j$. The unnormalized Laplacian is computed by subtracting the degree matrix from the similarity matrix in the same fashion as [Shi and Malik, 2000]. We choose the $\text{dims}$ eigenvectors with smallest eigenvalues, and perform $k$-means to find clusters in the reduced dimension. The optimal value for $\text{dims}$ is learned experimentally.

**MCStrat: Markov Chain-Based Algorithm.** The Markov chain approach uses local, recent information from $\pi^\text{partial}$, the partial trajectory to predict from. Given the last $k$ edges traversed by the vehicle, the algorithm finds all complete trajectories in the training set containing the same $k$ edges to build a set of relevant policies $V_{rel}$ using the match function. match($k, a, b$) returns 1 only if at least the last $k$ transitions in the policy generated by trajectory $a$ are also found in $b$. Using Equation 9, $V_{rel}$ is used to build a composite single relevant policy $\pi_{rel}$, that obeys the Markov assumption, so the resulting policy preserves the probability mass.

\[
V_{rel} = \{ \pi_i | \text{match}(k, \pi^\text{partial}_i, \pi^\text{partial}_i) = 1, \pi_i \in \Pi_{TR} \} \tag{7}
\]

\[
\pi_{rel} = \langle \delta_{k_1, s_2}^\text{rel}, \ldots, \delta_{k_l, s_j}^\text{rel} \rangle \tag{8}
\]

\[
\delta_{k_i, s_j}^\text{rel} = \frac{\sum_{\pi \in V_{rel}} \delta_{k_i, s_j}^\pi}{\sum_{\pi \in V_{rel}} \sum_{\delta_{k_i, s_j}^\pi}} \tag{9}
\]

Using the composite $\pi_{rel}$, policy iteration is then performed on the last location vector computed from $\pi^\text{predict}$.

**Method Complexity Comparison.** A comparison of the storage complexity of the methods appears in Table 1.

<table>
<thead>
<tr>
<th>Model</th>
<th>Model Construction</th>
<th>Model Storage</th>
</tr>
</thead>
<tbody>
<tr>
<td>FreqCount</td>
<td>$O(</td>
<td>\pi</td>
</tr>
<tr>
<td>EigenStrat</td>
<td>$O(</td>
<td>\Pi_{TR}</td>
</tr>
<tr>
<td>LapStrat</td>
<td>$O(</td>
<td>\Pi_{TR}</td>
</tr>
<tr>
<td>MCStrat</td>
<td>$O(1)$</td>
<td>$O(</td>
</tr>
</tbody>
</table>

Table 1: Space complexity of methods.
5 Results

Given an in-progress taxi trajectory, the methods presented facilitate predictions about the future movement of the vehicle. To simulate this task, a collection of partial trajectories (e.g. Figure 4) is generated from complete trajectories in the test set. A set of relevant policy vectors is generated using one of the four methods described, and policy iteration is performed to generate the future location predictions. The inferred future location matrix (e.g. Figure 5) is compared against the actual complete taxi trajectory (e.g. Figure 6). Prediction results are scored by comparing the inferred visited location vector \( \theta \) against the full location vector \( \theta^* \). The scores are computed using Pearson’s correlation:

\[
\text{score} = \text{Cor}(\theta, \theta^*)
\]

The scores reported are the aggregate mean of scores from examples in the validation set. The data set contains 100,000 subtrajectories (of approximately 1 hour in length) from 10,000 taxis. The data set is split randomly into 3 disjoint collections to facilitate experimentation: 90% in the training set, and 5% in both the test and validation sets. For each model type, the training set is used to generate the model. Model parameters are optimized using the test set. Scores are computed using predictions made on partial trajectories from the validation set.

Results of each method for 4 prediction horizons are shown in Table 2. The methods leveraging more local information near the last location of the vehicle (LapStrat, MCStrat) perform better than the methods relying only on global patterns (FreqCount, EigenStrat). This is true for all prediction horizons, but the more local methods have an even greater performance advantage for larger prediction horizons.

<table>
<thead>
<tr>
<th>Method</th>
<th>1</th>
<th>2</th>
<th>4</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>FreqCount</td>
<td>0.579 (.144)</td>
<td>0.593 (.127)</td>
<td>0.583 (.123)</td>
<td>0.573 (.122)</td>
</tr>
<tr>
<td>EigenStrat</td>
<td>0.563 (.143)</td>
<td>0.576 (.134)</td>
<td>0.574 (.140)</td>
<td>0.574 (.140)</td>
</tr>
<tr>
<td>LapStrat</td>
<td>0.590 (.144)</td>
<td>0.618 (.139)</td>
<td>0.626 (.137)</td>
<td>0.626 (.137)</td>
</tr>
<tr>
<td>MCStrat</td>
<td>0.600 (.146)</td>
<td>0.616 (.149)</td>
<td>0.621 (.149)</td>
<td>0.621 (.149)</td>
</tr>
</tbody>
</table>

Table 2: Correlation (std. dev.) by method and prediction horizon. The best score is in **bold**.

Statistical significance testing was performed on the validation set results, as shown in Table 3. The best performing methods (LapStrat and MCStrat) achieve a statistically significant performance improvement over the other methods. However, the relative performance difference between the local methods is not significantly different.

6 Conclusions

The methods presented can be applied to many other spatio-temporal domains where only basic location and time information is collected from portable devices, such as sensor networks as well as mobile phone networks. These predictions assume the action space is large but fixed and observations implicitly are clustered into distinct but repeated goals. In this domain, each observation is a set of actions a driver takes in fulfillment of a specific goal: for example, to take a passenger from the airport to his/her home. In future work, we propose to extend this work using a hierarchical approach which simultaneously incorporates global and local predictions to provide more robust results.

References


Learning Model Rules from High-Speed Data Streams

Ezilda Almeida and Carlos Ferreira and João Gama
LIAAD - INESC Porto L.A., Portugal

Abstract

Decision rules are one of the most expressive languages for machine learning. In this paper we present Adaptive Model Rules (AMRules), the first streaming rule learning algorithm for regression problems. In AMRules the antecedent of a rule is a conjunction of conditions on the attribute values, and the consequent is a linear combination of attribute values. Each rule in AMRules uses a Page-Hinkley test to detect changes in the process generating data and react to changes by pruning the rule set. In the experimental section we report the results of AMRules on benchmark regression problems, and compare the performance of our algorithm with other streaming regression algorithms.

Keywords: Data Streams, Regression, Rule Learning, Change Detection

1 Introduction

Regression analysis is a technique for estimating a functional relationship between a dependent variable and a set of independent variables. It has been widely studied in statistics, machine learning and data mining. Predicting numeric values usually involves complicated regression formulae. Model trees [14] and regression rules [15] are the most powerful data mining models. Trees and rules do automatic feature selection, being robust to outliers and irrelevant features; exhibit high degree of interpretability; and structural invariance to monotonic transformation of the independent variables. One important aspect of rules is modularity: each rule can be interpreted per si [6].

In the data stream computational model [7] examples are generated sequentially from time evolving distributions. Learning from data streams require incremental learning, using limited computational resources, and the ability to adapt to changes in the process generating data. In this paper we present Adaptive Model Rules, the first one-pass algorithm for learning regression rule sets from time-evolving streams. AMRules can learn ordered and unordered rules. The antecedent of a rule is a set of literals (conditions based on the attribute values). The consequent of a rule is a function that minimizes the mean square error of the target attribute computed from the set of examples covered by rule. This function might be either a constant, the mean of the target attribute, or a linear combination of the attributes. Each rule is equipped with an online change detector. It monitors the mean square error using the Page-Hinkley test, providing information about the dynamics of the process generating data. The paper is organized has follows. The next Section presents the related work in learning regression trees and rules from data focusing on streaming algorithms. Section 3 describe in detail the AMRules algorithm. Section 4 presents the experimental evaluation using stationary and time-evolving streams. AMRules is compared against other regression systems. Last Section presents the lessons learned.

2 Related Work

In this section we analyze the related work in two dimensions. One dimension is related to regression algorithms, the other dimension is related to incremental learning of regression algorithms.

In regression domains, [14] presented the system M5. It builds multivariate trees using linear models at the leaves. In the pruning phase for each leaf a linear model is built. Later, [5] have presented M5′ a rational reconstruction of Quinlan’s M5 algorithm. M5′ first constructs a regression tree by recursively splitting the instance space using tests on single attributes that maximally reduce variance in the target variable. After the tree has been grown, a linear multiple regression model is built for every inner node, using the data associated with that node and all the attributes that participate in tests in the subtree rooted at that node. Then the linear regression models are simplified by dropping attributes if this results in a lower expected error on future data (more specifically, if the decrease in the number of parameters outweighs the increase in the observed training error). After this has been done, every subtree is considered for pruning. Pruning occurs if the estimated error for the linear model at the root of a subtree is smaller or equal to the expected error for the subtree. After pruning terminates, M5′ applies a smoothing process that combines the model at a leaf with the models on the path to the root to form the final model that is placed at the leaf.

Cubist [15] is a rule based model that is an extension of Quinlan’s M5 model tree. A tree is grown where the terminal leaves contain linear regression models. These models are
based on the predictors used in previous splits. Also, there are intermediate linear models at each step of the tree. A prediction is made using the linear regression model at the terminal node of the tree, but is smoothed by taking into account the prediction from the linear model in the previous node of the tree (which also occurs recursively up the tree). The tree is reduced to a set of rules, which initially are paths from the top of the tree to the bottom. Rules are eliminated via pruning and/or combined for simplification.

2.1 Streaming Regression Algorithms

Many methods can be found in the literature for solving classification tasks on streams, but only a few exist for regression tasks. To the best of our knowledge, we note only two papers for online learning of regression and model trees. In the algorithm of [13] for incremental learning of linear model trees the splitting decision is formulated as hypothesis testing. The split least likely to occur under the null hypothesis of non-splitting is considered the best one. The linear models are computed using the RLS (Recursive Least Square) algorithm that has a complexity, which is quadratic in the dimensionality of the problem. This complexity is then multiplied with a user-defined number of possible splits per numerical attribute for which a separate pair of linear models is updated with each training example and evaluated. The Fast Incremental Model Tree (FIMT) proposed in [10], is an incremental algorithm for any-time model trees learning from evolving data streams with drift detection. It is based on the Hoeffding tree algorithm, but implements a different splitting criterion, using a standard deviation reduction (SDR) based measure more appropriate to regression problems. The FIMT algorithm is able to incrementally induce model trees by processing each example only once, in the order of their arrival. Splitting decisions are made using only a small sample of the data stream observed at each node, following the idea of Hoeffding trees. Another data streaming issue addressed in [10] is the problem of concept drift. Data streaming models capable of dealing with concept drift face two main challenges: how to detect when concept drift has occurred and how to adapt to the change. Change detection in the FIMT is carried out using the Page-Hinkley change detection test [11]. Adaptation in FIMT involves growing an alternate subtree from the node in which change was detected.

IBLStreams (Instance Based Learner on Streams) is an extension of MOA that consists in an instance-based learning algorithm for classification and regression problems on data streams by [1]; IBLStreams optimizes the composition and size of the case base autonomously. On arrival of a new example \((x_0, y_0)\), this example is first added to the case base. Moreover, it is checked whether other examples might be removed, either since they have become redundant or since they are outliers. To this end, a set \(C\) of examples within a neighborhood of \(x_0\) is considered as candidates. This neighborhood if given by the \(k_c\) nearest neighbors of \(x_0\), determined according a distance measure \(\Delta\). and the candidate set \(C\) consists of the examples within that neighborhood. The most recent examples are excluded from removal due to the difficulty to distinguish potentially noisy data from the beginning of a concept change. Even though unexpected observations should be removed only in the former but not in the latter case.

```
Algorithm 1: AMRules Algorithm

Input:
S: Stream of examples
ordered-set: boolean flag
N_{min}: Minimum number of examples
\lambda: Constant to solve ties
\alpha: the magnitude of changes that are allowed
j: rule index

Result: RS Set of Decision Rules

begin
   Let RS \{\}\rightarrow (L \leftarrow NULL)
   defaultRule \{\} \rightarrow \{L \leftarrow NULL\}
   foreach example \((x, y)\) do
      foreach Rule \(r \in RS\) do
         if \(r\) covers the example then
            Let \(y_i\) be the prediction of the rule \(r\),
            computed using \(L_r\).
            Compute error \(=(y_i - y)^2\)
            Call PHTTest(error, \alpha, \lambda)
            if Change is detected then
               Remove the rule
            else
               Update sufficient statistics of \(r\)
               Update Perceptron of \(r\)
               if Number of examples in \(L_r\) \(\geq N_{min}\) then
                  \(r \leftarrow \text{ExpandRule}(r)\)
               if ordered-set then
                  BREAK
         if none of the rules in RS triggers then
            Update sufficient statistics of the default rule
            Update Perceptron of the default rule
            if Number of examples in \(L\) \(\geq N_{min}\) then
               RS \leftarrow RS \cup \text{ExpandRule}(defaultRule)
   end
end
```

3 The AMRules Algorithm

The problem of learning model rules from data streams raises several issues. First, the dataset is no longer finite and available prior to learning, it is impossible to store all data in memory and learn from them as a whole. Second, multiple sequential scans over the training data are not allowed. An algorithm must therefore collect the relevant information at the speed it arrives and incrementally decide about splitting decisions. Third the training dataset may consist of data from different distributions. In this section we present an incremental algorithm for learning model rules to address these issues, named Adaptive Model Rules from High-Speed Data Streams (AMRules). The pseudo code of the algorithm is given in Algorithm 1.

The algorithm begins with an empty rule set (RS), and a default rule \{\} \rightarrow \(L\), where \(L\) is initialized to NULL. \(L\) is a data structure used to store the sufficient statistics required to expand a rule and for prediction. Every time a new training example is available the algorithm proceeds with checking
Algorithm 2: Expandrule: Expanding one Rule

Input:
\( r \): One Rule
\( \tau \): Constant to solve ties
\( \delta \): Confidence
Result: \( r' \): Expanded Rule

begin
  1. Let \( X_a \) be the attribute with greater SDR
  2. Let \( X_b \) be the attribute with second greater SDR
  3. Compute \( \epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2N}} \) (Hoeffding bound)
  4. Compute \( r = \frac{SDR(X_a)}{SDR(X_b)} \) (Ratio of the SDR values for the best two splits)
  5. Compute \( Upper \text{ Bound} = r + \epsilon \)
  6. if \( Upper \text{ Bound} < 1 \) then
     1. Extend \( r \) with a new condition based on the best attribute \( X_a \leq v_j \) or \( X_a > v_j \)
     2. Release sufficient statistics of \( L' \)
     3. \( r \leftarrow r \cup \{X_a \leq v_j \text{or} X_a > v_j\} \)
  return \( r \)

whether for each rule from rule set (RS) the example is covered by any rule, that is if all the literals are true for the example. The target values of the examples covered by a rule are used to update the sufficient statistic of the rule \( (\bar{S}) \). To detect changes we propose to use the Page-Hinkley (PH) change detection test. If a change is detected the rule is removed from the rule set. Otherwise, the rule might be expanded. The expansion of the rule is considered only after certain minimum number of examples \( (N_{min}) \). The expansion of a rule is explained in Algorithm 2.

The set of rules is learned in parallel, as described in Algorithm 1. We consider two cases: learning ordered or unordered set of rules. In the former case, every example updates statistics of the first rule that covers it. In the latter every example updates statistics of all the rules that covers it. If an example is not covered by any rule, the default rule is updated.

3.1 Expansion of a Rule

Before discussing how rules are expanded, we will first discuss the evaluation measure used in the attribute selection process. [10] describe a standard deviation reduction measure (SDR) for determining the merit of a given split. It can be efficiently computed in an incremental way. Given a leaf where a sample of the dataset \( S \) of size \( N \) has been observed, a hypothetical binary split \( h_A \) over attribute \( A \) would divide the examples in \( S \) in two disjoint subsets \( S_L \) and \( S_R \), with sizes \( N_L \) and \( N_R \) respectively. The formula for SDR measure of the split \( h_A \) is given below:

\[
SDR(h_A) = sd(S) - \frac{NL}{N}sd(S_L) - \frac{NR}{N}sd(S_R)
\]

\[
sd(S) = \sqrt{\frac{1}{N} \sum_{i=1}^{N} (y_i - \bar{y})^2}
\]

To make the actual decision regarding a split, the SDR measured for the best two potential splits are compared, by dividing the second-best value by the best one to generate a ratio \( r \) in the range 0 to 1. Having a predefined range for the values of the random variables, the Hoeffding probability bound \( (\epsilon) \) [17] can be used to obtain high confidence intervals for the true average of the sequence of random variables. The value of \( \epsilon \) is calculated using the formula:

\[
\epsilon = \sqrt{\frac{R^2 \ln(1/\delta)}{2N}}
\]

The process to expand a rule by adding a new condition works as follows. For each attribute \( X_a \), the value of the SDR is computed for each attribute value \( v_j \). If the upper bound \( (r^+ = r + \epsilon) \) of the sample average is below 1 then the true mean is also below 1. Therefore with confidence \( 1 - \epsilon \) the best attribute over a portion of the data is really the best attribute. In this case, the rule is expanded with condition \( X_a \leq v_j \) or \( X_a > v_j \). However, often two splits are extremely similar or even identical, in terms of their SDR values, and despite the \( \epsilon \) intervals shrinking considerably as more examples are seen, it is still impossible to choose one split over the other. In these cases, a threshold \( (\tau) \) on the error is used. If \( \epsilon \) falls below this threshold and the splitting criterion is still not met, the split is made on the best split with a higher SDR value and the rule is expanded.

3.2 Prediction Strategies

The set of rules learned by AMRules can be ordered or unordered. They employ different prediction strategies to achieve optimal prediction. In the former, only the first rule that cover an example is used to predict the target example. In the latter, all rules covering the example are used for prediction and the final prediction is decided by using weighted vote.

Each rule in AMRules implements 3 prediction strategies: i) the mean of the target attribute computed from the examples covered by the rule; ii) a linear combination of the independent attributes; iii) an adaptive strategy, that chooses between the first two strategies, the one with lower MSE in the previous examples.

Each rule in AMRules contains a linear model, trained using an incremental gradient descent method, from the examples covered by the rule. Initially, the weights are set to small random numbers in the range -1 to 1. When a new example arrives, the output is computed using the current weights. Each weight is then updated using the Delta rule:

\[
w_i \leftarrow w_i + \eta (y - \hat{y}) x_i,
\]

where \( \hat{y} \) is the output, \( y \) the real value and \( \eta \) is the learning rate.

3.3 Change Detection

The AMRules uses the Page-Hinkley (PH) test [12] to monitor the error and signals a drift when a significant increase of this variable is observed. The PH test is a sequential analysis technique typically used for monitoring change detection in
signal processing. The PH test is designed to detect a change in the average of a Gaussian signal [11]. This test considers a cumulative variable $m_T$, defined as the accumulated difference between the observed error and the mean of the error till the current moment:

$$m_T = \sum_{t=1}^{T} (e_t - \bar{e}_T - \alpha)$$

where $\bar{e}_T = 1/T \sum_{t=1}^{T} e_t$ and $\alpha$ corresponds to the magnitude of changes that are allowed.

The minimum value of this variable is also computed: $M_T = \min(m_t, t = 1 \ldots T)$. As a final step, the test monitors the difference between $M_T$ and $m_T$: $PH_T = m_T - M_T$. When this difference is greater than a given threshold ($\lambda$) we signal a change in the distribution. The threshold $\lambda$ depends on the admissible false alarm rate. Increasing $\lambda$ will entail fewer false alarms, but might miss or delay change detection.

4 Experimental Evaluation

The main goal of this experimental evaluation is to study the behavior of the proposed algorithm in terms of mean absolute error (MAE) and root mean squared error (RMSE). We are interested in studying the following scenarios:

- How to grow the rule set?
  - Update only the first rule that covers training examples. In this case the rule set is ordered, and the corresponding prediction strategy uses only the first rule that covers test examples.
  - Update all the rules that covers training examples. In this case the rule set is unordered, and the corresponding prediction strategy uses a weighted sum of all rules that covers test examples.

- How does AMRules compares against other streaming algorithms?
- How does AMRules compares against other state-of-the-art regression algorithms?
- How does AMRules learned models evolve in time-changing streams?

4.1 Experimental Setup

All our algorithms were implemented in java using Massive Online Analysis (MOA) data stream software suite [2]. For all the experiments, we set the input parameters of AMRules to: $N_{\text{min}} = 200$, $\tau = 0.05$ and $\delta = 0.01$. The parameters for the Page-Hinkley test are $\lambda = 50$ and $\alpha = 0.005$. Table 1 summarizes information about the datasets used and reports the learning rate used in the perceptron learning.

All of the results in the tables 2, 3 and 4 are averaged of ten-fold cross-validation [16]. The accuracy is measured using the following metrics: Mean absolute error (MAE) and root mean squared error (RMSE) [19]. We used two evaluation methods. When no concept drift is assumed, the evaluation method we employ uses the traditional train and test scenario. All algorithms learn from the same training set and the error is estimated from the same test sets. In scenarios with concept drift, we use the prequential (predictive sequential) error estimate [8]. This evaluation method evaluates a model sequentially. When an example is available, the current regression model makes a prediction and the loss is computed. After the prediction the regression model is updated with that example.

Datasets

The experimental datasets include both artificial and real data, as well sets with continuous attributes. We use ten regression datasets from the UCI Machine Learning Repository [3] and other sources. The datasets used in our experimental work are:

- **2dplanes** this is an artificial data set described in [4].
- **Airlerons** this data set addresses a control problem, namely flying a F16 aircraft.
- **Puma8NH** and **Puma32H** is a family of datasets synthetically generated from a realistic simulation of the dynamics of a Unimation Puma 560 robot arm. **Pol** this is a commercial application described in [18].The data describes a tele communication problem. **Elevators** this data set is also obtained from the task of controlling a F16 aircraft.
- **Fried** is an artificial data set used in Friedman (1991) and also described in Breiman (1996,p.139).
- **Bank8FM** a family of datasets synthetically generated from a simulation of how bank-customers choose their banks. **Kin8nm** this dataset is concerned with the forward kinematics of an 8 link robot arm. **Airline** this dataset using the data from the Data Expo competition (2009). The dataset consists of a large amount of records, containing flight arrival and departure details for all the commercial flights within the USA, from October 1987 to April 2008. This is a large dataset with nearly 120 million records (11.5 GB memory size) [10]. Table 1 summarizes the number of instances and the number of attributes of each dataset.

<table>
<thead>
<tr>
<th>Datasets</th>
<th># Instances</th>
<th># Attributes</th>
<th>Learning rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>2dplanes</td>
<td>40768</td>
<td>11</td>
<td>0.01</td>
</tr>
<tr>
<td>Airlerons</td>
<td>13750</td>
<td>41</td>
<td>0.01</td>
</tr>
<tr>
<td>Puma8NH</td>
<td>8192</td>
<td>9</td>
<td>0.01</td>
</tr>
<tr>
<td>Puma32H</td>
<td>8192</td>
<td>32</td>
<td>0.01</td>
</tr>
<tr>
<td>Pol</td>
<td>13000</td>
<td>49</td>
<td>0.001</td>
</tr>
<tr>
<td>Elevators</td>
<td>8752</td>
<td>19</td>
<td>0.001</td>
</tr>
<tr>
<td>Fried</td>
<td>40769</td>
<td>11</td>
<td>0.01</td>
</tr>
<tr>
<td>Bank8FM</td>
<td>8192</td>
<td>9</td>
<td>0.01</td>
</tr>
<tr>
<td>Kin8nm</td>
<td>8192</td>
<td>9</td>
<td>0.01</td>
</tr>
<tr>
<td>Airline</td>
<td>113Million</td>
<td>11</td>
<td>0.01</td>
</tr>
</tbody>
</table>

4.2 Experimental Results

In this section, we empirically evaluate the AMRules. The results are described in four parts. In the first part we compare the AMRules variants, the second part we compare AMRules against other streaming algorithms and the third part compare AMRules against other state-of-the-art regression algorithms. The last part presents the analysis of AMRules behavior in the context of time-evolving data streams.
Comparison between AMRules Variants

In this section we focus on two strategies that we found potentially interesting. It is a combination of expanding only one rule, the rule that first triggered, with predicting strategy uses only the first rule that covers test examples. Obviously, for this approach it is necessary to use ordered rules (AMRules*). The second setting employs unordered rule set, where all the covering rules expand and the corresponding prediction strategy uses a weighted sum of all rules that cover test examples (AMRules+).

Ordered rule sets specializes one rule at a time and, as a result it often produces less rules than the unordered strategy. Ordered rules need to consider the previous rules and remaining combinations, which might not be easy to interpret in more complex sets. Unordered rule sets are more modular, because they can be interpreted alone.

Table 2 summarize the mean absolute error and the root mean squared error of these variants. Overall, the experimental results points out the unordered rule sets are more competitive than ordered rule sets in terms of MAE and RMSE.

### Table 2. Results of ten-fold cross-validation for AMRules algorithms

<table>
<thead>
<tr>
<th>Datasets</th>
<th>AMRules</th>
<th>AMRules+</th>
<th>AMRules*</th>
<th>AMRules*</th>
</tr>
</thead>
<tbody>
<tr>
<td>Airline</td>
<td>1.10E-04 (0.00)</td>
<td>1.00E-04 (0.00)</td>
<td>1.90E-04 (0.00)</td>
<td>1.70E-04 (0.00)</td>
</tr>
<tr>
<td>Pol</td>
<td>15.6E+00 (3.70)</td>
<td>15.6E+00 (3.70)</td>
<td>23.0E00 (44.50)</td>
<td>23.3E00 (4.08)</td>
</tr>
<tr>
<td>Elevators</td>
<td>5.0E-03 (0.00)</td>
<td>1.90E-03 (0.00)</td>
<td>4.80E-03 (0.00)</td>
<td>2.20E-03 (0.00)</td>
</tr>
<tr>
<td>Fried</td>
<td>2.0E-00 (0.00)</td>
<td>1.13E+00 (0.03)</td>
<td>2.70E+00 (0.03)</td>
<td>1.67E+00 (0.25)</td>
</tr>
<tr>
<td>BankSFM</td>
<td>4.31E-02 (0.00)</td>
<td>4.30E-02 (0.00)</td>
<td>4.80E-02 (0.00)</td>
<td>4.30E-02 (0.00)</td>
</tr>
<tr>
<td>KinSm</td>
<td>1.60E-01 (0.00)</td>
<td>1.50E-01 (0.00)</td>
<td>2.10E-01 (0.00)</td>
<td>2.00E-01 (0.00)</td>
</tr>
</tbody>
</table>

Comparison with other Streaming Algorithms

We compare the performance of our algorithm with three other streaming algorithms, FIMT and IBLStreams. FIMT is an incremental algorithm for learning model trees, addressed in [10]. IBLStreams is an extension of MOA that consists in an instance-based learning algorithm for classification and regression problems on data streams by [1].

The performance measures for these algorithms are given in Table 3. The comparison of these streaming algorithms shows that AMRules get better results.

Comparison with State-of-the-art Regression Algorithms

Another experiment which involves adaptive model rules is shown in Table 4. We compare AMRules with other non-incremental regression algorithms available in WEKA [9]. All these experiments using algorithms are performed using WEKA. We use the standard method of ten-fold cross-validation, using the same folds for all the algorithms included.

The comparison of these algorithms show that AMRules is very competitive in terms of (MAE, RMSE) than all the other methods, except M5Rules. AMRules is faster than all the other algorithms considered in this study. These results were somewhat expected, since these datasets are relatively small for the incremental algorithm.

Table 5. Average results from the evaluation of change detection over ten experiments.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Delay</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>AMRules</td>
<td>1485</td>
<td>56 (nr. Rules)</td>
</tr>
<tr>
<td>FIMT</td>
<td>2096</td>
<td>590 (nr. Leaves)</td>
</tr>
</tbody>
</table>

Evaluation in Time-Evolving Data streams

In this subsection we first study the evaluation of the error measurements (MAE and RMSE) and evaluate the change detection method. After, we evaluate the streaming algorithms on non-stationary streaming real-world problem, we use the Airline dataset from the DataExpo09 competition.

To simulate drift we use Fried dataset. The simulations allow us to control the relevant parameters and to evaluate the drift detection. Figures 1 and Figure 2 depict the MAE and RMSE curves of the streaming algorithms using the dataset Fried. These figures also illustrate the point of drift and the points where the change was detected. Only two of the algorithms – FIMT and AMRules – were able to detect a change. Table 5 report the average results over ten experiments varying the seed of the Fried dataset. We measure the number of nodes for FIMT, the number of rules AMRules and the the delay (in terms of number of examples) in detection the drift. The delay gives indication of how fast the algorithm will be able to start the adaptation strategy. These two algorithms obtained similar results. The general conclusions are that FIMT and AMRules algorithms are robust and have better results than IBLStreams. Figures 3 and 4 show the evaluation of the MAE and the RMSE of the streaming algorithms on non-stationary real-world problem. FIMT and AMRules obtain approximately similar behavior in terms of MAD and MSE. Both exhibit somewhat better performance than IBLStreams, but not significantly different.
Table 3. Results of ten-fold cross-validation for Streaming Algorithms

<table>
<thead>
<tr>
<th>Datasets</th>
<th>AMRules(^a) (MAE)</th>
<th>FIMT (MAE)</th>
<th>IBLStreams (MAE)</th>
<th>AMRules(^a) (RMSE)</th>
<th>FIMT (RMSE)</th>
<th>IBLStreams (RMSE)</th>
</tr>
</thead>
<tbody>
<tr>
<td>2dplanes</td>
<td>1.16E+00 (0.01)</td>
<td>8.00E-01 (0.00)</td>
<td>1.03E+00 (0.00)</td>
<td>1.52E+00 (0.01)</td>
<td>1.00E+00 (0.00)</td>
<td>1.30E+00 (0.00)</td>
</tr>
<tr>
<td>Airlerons</td>
<td>1.00E-04 (0.00)</td>
<td>1.90E-04 (0.00)</td>
<td>3.20E-04 (0.00)</td>
<td>1.70E-04 (0.00)</td>
<td>1.00E-09 (0.00)</td>
<td>3.00E-04 (0.00)</td>
</tr>
<tr>
<td>Puma32H</td>
<td>2.66E+00 (0.01)</td>
<td>3.26E+00 (0.03)</td>
<td>3.27E+00 (0.01)</td>
<td>4.28E+00 (0.01)</td>
<td>1.20E+03 (0.00)</td>
<td>3.84E+00 (0.02)</td>
</tr>
<tr>
<td>Airlerons</td>
<td>2.00E-02 (0.00)</td>
<td>7.90E-03 (0.00)</td>
<td>2.20E-02 (0.00)</td>
<td>1.00E+00 (0.00)</td>
<td>1.20E-02 (0.00)</td>
<td>2.70E-02 (0.00)</td>
</tr>
<tr>
<td>Pol</td>
<td>1.56E+00 (3.70)</td>
<td>38.2E+00 (0.17)</td>
<td>29.7E+00 (0.55)</td>
<td>4.28E+00 (0.03)</td>
<td>1.00E+09 (0.00)</td>
<td>3.84E+00 (0.02)</td>
</tr>
<tr>
<td>Airlerons</td>
<td>1.00E-04 (0.00)</td>
<td>1.40E-04 (0.00)</td>
<td>1.10E-04 (0.00)</td>
<td>1.70E-04 (0.00)</td>
<td>2.00E-04 (0.00)</td>
<td>4.45E+00 (0.01)</td>
</tr>
<tr>
<td>Puma32H</td>
<td>3.26E+00 (0.03)</td>
<td>3.34E+00 (0.17)</td>
<td>3.64E+00 (0.01)</td>
<td>2.00E+03 (0.00)</td>
<td>1.75E+03 (1.38)</td>
<td>50.7E+00 (0.71)</td>
</tr>
<tr>
<td>Elevators</td>
<td>1.90E-03 (0.00)</td>
<td>5.00E-03 (0.00)</td>
<td>2.20E+03 (0.00)</td>
<td>3.00E+03 (0.00)</td>
<td>6.20E+03 (0.00)</td>
<td>9.60E+00 (0.00)</td>
</tr>
<tr>
<td>Fried</td>
<td>1.13E+00 (0.00)</td>
<td>1.97E+00 (0.00)</td>
<td>1.67E+00 (0.00)</td>
<td>1.67E+00 (0.00)</td>
<td>2.11E+00 (0.00)</td>
<td>2.71E+00 (0.00)</td>
</tr>
<tr>
<td>Bank8FM</td>
<td>4.30E-02 (0.00)</td>
<td>3.30E-02 (0.00)</td>
<td>7.0E-02 (0.00)</td>
<td>2.20E-03 (0.00)</td>
<td>9.60E-02 (0.00)</td>
<td>9.60E-02 (0.00)</td>
</tr>
<tr>
<td>Pol</td>
<td>1.56E+00 (3.70)</td>
<td>38.2E+00 (0.17)</td>
<td>29.7E+00 (0.55)</td>
<td>4.28E+00 (0.03)</td>
<td>1.00E+09 (0.00)</td>
<td>3.84E+00 (0.02)</td>
</tr>
<tr>
<td>Elevators</td>
<td>1.20E-02 (0.00)</td>
<td>6.80E-03 (0.00)</td>
<td>2.30E-03 (0.00)</td>
<td>2.00E-03 (0.00)</td>
<td>8.60E-03 (0.00)</td>
<td>3.10E-02 (0.00)</td>
</tr>
<tr>
<td>Fried</td>
<td>1.13E+00 (0.00)</td>
<td>2.20E-02 (0.00)</td>
<td>2.29E-03 (0.00)</td>
<td>1.67E+00 (0.00)</td>
<td>1.69E+00 (0.00)</td>
<td>2.62E+00 (0.00)</td>
</tr>
<tr>
<td>Kin8nm</td>
<td>1.60E-01 (0.00)</td>
<td>1.60E-01 (0.00)</td>
<td>9.50E-01 (0.00)</td>
<td>2.00E-01 (0.00)</td>
<td>2.10E-01 (0.00)</td>
<td>1.20E-01 (0.00)</td>
</tr>
</tbody>
</table>

Fig. 2. Root mean squared error of streaming algorithms using the dataset Fried.

Fig. 3. Mean absolute error of streaming algorithms using the dataset Airlines.

Fig. 4. Root mean squared error of streaming algorithms using the dataset Airlines.

the best of our knowledge, in the literature there is no other method that addresses this issue.

AMRules learns ordered and unordered rule sets. The experimental results point out that unordered rule sets, in comparison to ordered rule sets, are more competitive in terms of error metrics (MAE and RMSE). AMRules achieves better results than the others algorithms even for medium sized datasets. The AMRule algorithm is equipped with explicit change detection mechanisms that signals change points during the learning process. This information is relevant to understand the dynamics of evolving streams.

Acknowledgments:
The authors acknowledge the financial support given by the projects FCT-KDUS (PTDC/EIA/098355/2008); FCOMP - 01-0124-FEDER-010053, the ERDF through the COMPETE
Programme and by Portuguese National Funds through FCT within the project FCOMP - 01-0124-FEDER-022701.

References

On Recommending Urban Hotspots to Find Our Next Passenger

Luis Moreira-Matias\textsuperscript{1,2,3}, Ricardo Fernandes\textsuperscript{1}, João Gama\textsuperscript{2,5}, Michel Ferreira\textsuperscript{1,4}, João Mendes-Moreira\textsuperscript{2,3}, Luis Damas\textsuperscript{6}

\textsuperscript{1} Instituto de Telecomunicações, University of Porto, Portugal
\textsuperscript{2} LIACC/INESC TEC, Porto, Portugal
\textsuperscript{3} Department of Informatics Engineering, Faculty of Engineering, University of Porto, Portugal
\textsuperscript{4} Department of Computer Sciences, Faculty of Sciences, University of Porto, Portugal
\textsuperscript{5} Faculty of Economics, University of Porto, Portugal
\textsuperscript{6} GEOLINK, Porto, Portugal

\{luis.m.matias, jjgama, joao.mendes.moreira\}@atlineosporto.pt, \{jff, michel\}@atdcc.fc.up.pt, luis@geolink.pt

Abstract

The rising fuel costs is disallowing random cruising strategies for passenger finding. Hereby, a recommendation model to suggest the most passenger-profitable urban area/stand is presented. This framework is able to combine the 1) underlying historical patterns on passenger demand and the 2) current network status to decide which is the best zone to head to in each moment. The major contribution of this work is on how to combine well-known methods for learning from data streams (such as the historical GPS traces) as an approach to solve this particular problem. The results were promising: 395.361/506.873 of the services dispatched were correctly predicted. The experiments also highlighted that a fleet equipped with such framework surpassed a fleet that is not: they experienced an average waiting time to pick-up a passenger 5% lower than its competitor.

1 Introduction

The taxis became crucial for human mobility in medium/large-sized urban areas. They provide a direct, comfortable and speedy way to move in and out of big town centers - as complement to other transportation means or as a main solution. In the past years, the city councils tried to guarantee that the running vacant taxis will always meet the demand in their urban areas by emitting more taxi licenses than the necessary. As result, the cities’ cores are commonly crowded by a huge number of vacant taxis - which take desperate measures to find new passengers such as random cruise’ strategies. These strategies have undesirable side effects like large wastes of fuel, an inefficient traffic handling, an increase of the air pollution.

The taxi driver mobility intelligence is one of the keys to mitigate this problems. The knowledge about where the services (i.e. the transport of a passenger from a pick-up to a drop-off location) will actually emerge can truly be useful to the driver – especially where there are more than one competitor operating. Recently, the major taxi fleets are equipped with GPS sensors and wireless communication devices. Typically, these vehicles will transmit information to a data center about their location and the events undergoing like the passenger pick-up and drop-off. These historical traces can reveal the underlying running mobility patterns. Multiple works in the literature have already explored this kind of data successfully with distinct applications like smart driving [Yuan et al., 2010], modeling the spatiotemporal structure of taxi services [Deng and Ji, 2011; Liu et al., 2009; Yue et al., 2009], building passenger-finding strategies [Li et al., 2011; Lee et al., 2008] or even predicting the taxi location in a passenger-perspective [Phithakkitnukoon et al., 2010]. Despite their useful insights, the majority of the techniques reported are offline, discarding the main advantages of this signal (i.e. a streaming one).

In our work, we focus on the online choice problem about which is the best taxi stand to go to after a passenger drop-off (i.e. the stand where we will pick-up another passenger quicker). Our goal is to use the vehicular network communication framework to improve their reliability by combining all drivers’ experience. In other words, the idea is to forecast how many services will arise in each taxi stand based on the network past behavior to feed a recommendation model to calculate the best stand to head to. An illustration about our problem is presented in Fig. 1 (the five blue dots represent possible stands to head to after a passenger drop-off; our recommendation system outputs one of them as the best choice at the moment).

Such recommendation model can present a true advantage for a fleet when facing other competitors, which will work with less information than you do. This tool can improve the informed driving experience by transmitting to the driver

Figure 1: Taxi Stand choice problem.
which is the stand where 1) he will wait less time to get a pas-
enger in; or where 2) he will get the service with the greatest
revenue.

The smart stand-choice problem is based on four key de-
cision variables: the expected price for a service over time,
the distance/cost relation with each stand, how many taxis are
already waiting at each stand and the passenger demand for
each stand over time. The taxi vehicular network can be a
ubiquitous sensor of taxi-passerger demand from which we
can continuously mine the reported variables. However, the
work described here will just address the decision process
based on the last three variables.

In our previous work [Moreira-Matias et al., 2012], we al-
ready proposed a model to predict the spatiotemporal distri-
bution of the taxi passenger demand (i.e. the number of ser-
dices that will emerge along the taxi stand network). This
study departed from this initial work to extend it along three
different dimensions:

1. The Recommendation System: we use these predic-
tions as input to a Recommendation System that also
accounts the number of taxis already in a stand and the
distance to it. Such framework will improve the taxi
driver mobility intelligence in real time, helping him to
decide which is the most profitable stand in each mo-
ment. It will be based not only in his own past decisions
and outcomes, but on a combination of everyone experi-
tence, taking full advantage of the ubiquitous charac-
teristics of the vehicular communicational networks.

2. Test-bed: Our experiments took advantage of the ve-
hicular network online information to feed the predic-
tive framework. Moreover, the recommendation perfor-
cance was evaluated in real-time, demonstrating its
robustness and its ability to learn, decide and evolve
without a high computational effort;

3. Dataset: 506.873 services were dispatched to our 441
vehicle fleet during our experiments. This large scale
test was carried out along 9 months.

There are some works in the literature related with this prob-
lem, namely: 1) mining the best passenger-finding strate-
gies [Li et al., 2011; Lee et al., 2008], 2) dividing the ur-
ban area into attractive clusters based on the historical pas-
senger demand (i.e.: city zones with distinct demand pat-
terns) [Deng and Li, 2011; Liu et al., 2009; Yue et al., 2009]
and even 3) predicting the passenger demand at certain ur-
ban hotspots [Li et al., 2012; Kaltenbrunner et al., 2010;
Chang et al., 2010]. The major contribution of this work
facing this state-of-the-art is to build smart recommenda-
tions about the taxi stand to head to in an online streaming
environment (i.e. real-time; while the taxis are operating)
based not only on their historical trace but also on the cur-
rent network status. In fact, the reported works present offline
frameworks and/or test-beds or just account a low number of
decision variables.

The results were obtained using two distinct test-beds:
firstly, (1) we let the stream run continuously between Au-
gust 2011 and April 2012. The predictive model was trained
during the first five months and it was stream-tested in the
last four. Secondly, (2) we used a traffic simulator to test
if our Recommendation System could beat the drivers’ ex-
pected behavior. We simulated a competitive scenario –
with two fleets - using the services historical log and on the exist-
ing road network system. The obtained results validated that
our method can effectively help the drivers to decide where
they can achieve more profit.

The remainder of the paper is structured as follows. Sec-
tion 2 formally presents our predictive model while Section
3 details our recommendation one. The fourth section de-
scribes our case study, how we acquired and preprocessed the
data used as well as some statistics about it. The fifth section
describes how we tested the methodology in a concrete sce-
nario: firstly, we introduce the two experimental setups and
the metrics used to evaluate both models. Then, the obtained
results are detailed, followed by some important remarks. Fi-
nally, conclusions are drawn.

2 The Predictive Model

In this section we present some relevant definitions and a brief
description of the predictive model on taxi passenger demand.
The reader should consult the section II in [Moreira-Matias et
al., 2012] for further details. Let 
\[ S = \{s_1, s_2, ..., s_N\} \]
be the set of \( N \) taxi stands of interest and 
\[ D = \{d_1, d_2, ..., d_j\} \]
a set of \( j \) possible passenger destinations. Our problem is to
choose the best taxi stand at the instant \( t \) according with our
forecast about passenger demand distribution over the time
stands for the period \([t, t + P]\).

Consider \( X_k = \{X_{k,0}, X_{k,1}, ..., X_{k,t}\} \) to be a discrete
time series (aggregation period of \( P \)-minutes) for the number
of demanded services at a taxi stand \( k \). The goal is to build a
model which determines the set of service counts \( X_{k,t+1}\)
for instant \( t + 1 \) and per taxi stand \( k \in \{1, ..., N\} \). To do
so, three distinct short-term prediction models are proposed,
as well as a well-known data stream ensemble framework to
use all models. We briefly describe these models along this
section.

2.1 Time Varying Poisson Model

Consider the probability for \( n \) taxi assignments to emerge in
a certain time period - \( P(n) \) - following a Poisson Distri-
bution. It is possible to define it using the following equation

\[
P(n; \lambda) = \frac{e^{-\lambda} \lambda^n}{n!}
\]

(1)

where \( \lambda \) represents the rate (average demand for taxi services)
in a fixed time interval. However, in this specific problem,
the rate \( \lambda \) is not constant but time-variant. Therefore, it was
adapted as a function of time, i.e. \( \lambda(t) \), transforming the Pois-
son distribution into a non homogeneous one. Let \( \lambda_0 \)
be the average (i.e. expected) rate of the Poisson process over a
full week. Consider \( \lambda(t) \) to be defined as follows

\[
\lambda(t) = \lambda_0 \delta_{d(t)} \eta_{h(t), b(t)}
\]

(2)

where \( \delta_{d(t)} \) is the relative change for the weekday \( d(t) \) (e.g.: Saturdays have lower day rates than Tuesdays); \( \eta_{h(t), b(t)} \) is
the relative change for the period \( h(t) \) in the day \( d(t) \) (e.g. the
peak hours); \( d(t) \) represents the weekday \( 1=Sunday, 2=Mon-
day, ..., \); and \( h(t) \) represents the period when time \( t \) falls (e.g.
the time 00:31 is contained in period 2 if we consider 30-minutes periods).

2.2 Weighted Time Varying Poisson Model
The model previously presented can be faced as a time-dependent average which produces predictions based on the long-term historical data. However, it is not guaranteed that every taxi stand will have a highly regular passenger demand: actually, the demand in many stands can often be seasonal.

The sunny beaches are a good example on the demand seasonality: the taxi demand around them will be higher on summer weekends rather than other seasons along the year.

To face this specific issue, a weighted average model is proposed based on the one presented before: the goal is to increase the relevance of the demand pattern observed in the recent week (e.g. what happened on the previous Tuesday is more relevant than what happened two or three Tuesdays ago). The weight set \( \omega \) is calculated using a well-known time series approach to these type of problems: the Exponential Smoothing [Holt, 2004]. This model will enhance the importance of the mid-term historical data rather than the long-term one already proposed in the above section.

2.3 Autoregressive Integrated Moving Average Model
The two previous models assume the existence of a regular (seasonal or not) periodicity in taxi service passenger demand (i.e. the demand at one taxi stand on a regular Tuesday during a certain period will be highly similar to the demand verified during the same period on other Tuesdays). However, the demand can present distinct periodicities for different stands. The ubiquitous features of this network force us to rapidly decide if and how the model is evolving so that it is possible to adapt to these changes instantly.

The AutoRegressive Integrated Moving Average Model (ARIMA) [Box et al., 1976] is a well-known methodology to both model and forecast univariate time series data such as traffic flow data [Min and Wynter, 2011], electricity price [Contreras et al., 2003] and other short-term prediction problems such as the one presented here. There are two main advantages of using ARIMA when compared to other algorithms. Firstly, 1) it is versatile to represent very different types of time series: the autoregressive (AR) ones, the moving average ones (MA) and a combination of those two (ARMA); Secondly, 2) it combines the most recent samples from the series to produce a forecast and to update itself to changes in the model. A brief presentation of one of the simplest ARIMA models (for non-seasonal stationary time series) is presented below following the existing description in [Zhang, 2003] (however, our framework can also detect both seasonal and non-stationary series). For a more detailed discussion, the reader should consult a comprehensive discussion in this specific stand during the sliding window \( H \). In fact, \( \rho_H \) works as a certainty about our prediction (i.e. if two stands have the same \( SD \) but our model is experiencing a bigger error in one of them, the other stand should be picked instead).

Let \( C_{k,t+1} \) be the number of taxis already parked in the stand \( k \) in the drop-off moment and \( L_{k,w} \) be the number of services departed from the same stand between this moment and the moment of the last forecast (i.e.: \( t \)). We can define the service deficit - \( SD_{k,t+w} \) on the taxi stand \( k \) i.e.; a prediction on the number of services that will be demanded in the stand discounting the vehicles already waiting in the line) as

\[
SD_{k,t+w} = (X_{k,t+1} - C_{k,t+1} - L_{k,w}) \times \rho_H
\]

3 Recommendation Model
Let \( X_{k,t+1} \) be the number of services to be demanded in the taxi stand \( k \) during the 30-minutes period next to the time instant \( t \). Then, a passenger is dropped-off somewhere by a vehicle of interest \( w \) minutes after the last forecast on the instant \( t \). The problem is to choice one of the possible taxi stands to head to. This choice is related with four key variables: the expected price for a service over time, the distance to each stand, how many taxis are already waiting at each stand and the predicted passenger demand. However, here we solve this issue like a minimization problem: we want to rank the stands according the minimum waiting time (target variable) to pick-up a passenger, whenever it is directly picked-up or dispatched by the central.

Let \( U_k \) be the distance (in kilometres) between the drop-off location and the taxi stand \( k \). We can define the normalized distance to the stand - \( U_k \) - as follows

\[
U_k = 1 - \frac{U_k}{\xi}
\]

where \( \xi \) is the distance to the farthest stand. We can calculate the Recommendation Score of the taxi stand \( k \) as

\[
RS_k = U_k \times SD_{k,t+w}
\]

Then, we calculate the Recommendation Score of every stands and we recommend to the driver the stand with the highest one.

4 Data Acquisition and Preprocessing
The stream events data of a taxi company operating in the city of Porto, Portugal, was used as case study. This city is the center of a medium-sized urban area (consisting of 1.3 million inhabitants) where the passenger demand is lower than
the number of running vacant taxis, resulting in a huge competition between both companies and drivers. The data was continuously acquired using the telematics installed in each one of the 441 running vehicles of the company fleet throughout a non-stop period of nine months. This study just uses as input/output the services obtained directly at the stands or those automatically dispatched to the parked vehicles (more details in the section below). This was done because the passenger demand at each taxi stand is the main feature to aid the taxi drivers’ decision.

Statistics about the period studied are presented. Table 1 details the number of taxi services demanded per daily shift and day type. Table 2 contains information about all services per taxi/driver and cruise time. The service column in Table 2 represents the number of services taken by the taxi drivers, while the second represents the total cruise time of every service. Additionally, it is possible to state that the central service assignment is 24% of the total service (versus the 76% of the service requested directly on the street) while 77% of the service is demanded directly to taxis parked in a taxi stand (and 23% is assigned while they are cruising). The average waiting time (to pick-up passengers) of a taxi parked at a taxi stand is 42 minutes while the average time for a service is only 11 minutes and 12 seconds. Such low ratio of busy/vacant time reflects the current economic crisis in Portugal and the regulators’ inability to reduce the number of taxis in the city. It also highlights the importance of the predictive system presented here, where the shortness of services could be mitigated by obtaining services from the competitors.

5 Experimental Results

In this section, we firstly describe the experimental setup developed to test our predictive model on the available data. Secondly, we introduce our simulation model and the experiments associated with. Thirdly, we present our Recommendation System and the metrics used to evaluate our methods. Finally, we present the results.

5.1 Experimental Setup for the Predictive Model

Our model produces an online forecast for the taxi-passenger demand at all taxi stands at each P-minutes period. Our test-bed was based on prequential evaluation: data about the network events was continuously acquired.

Each data chunk was transmitted and received through a socket. The model was programmed using the R language. The prediction effort was divided into three distinct processes running on a multicore CPU (the time series for each stand is independent from the remaining ones) which reduced the computational time of each forecast. The pre-defined functions used and the values set for the models parameters are detailed along this section.

An aggregation period of 30 minutes was set (i.e. a new forecast is produced each 30 minutes; P=30) and a radius of 100 m (W = 100 ¿ 50 defined by the existing regulations). It was set based on the average waiting time at a taxi stand, i.e. a forecast horizon lower than 42 minutes.

The ARIMA model (p,d,q values and seasonality) was firstly set (and updated each 24h) by learning/detecting the underlying model (i.e. autocorrelation and partial autocorrelation analysis) running on the historical time series curve for each considered taxi stand. To do so, we used an automatic time series function in the [forecast] R package [Yeasmin and Rob, 1999] – auto-arima – with the default parameters. The weights/parameters for each model are specifically fit for each period/prediction using the function arima from the built-in R package [stats].

The time-varying Poisson averaged models (both weighted and non-weighted) were also updated every 24 hours. A sliding window of 4 hours (H=8) was considered in the ensemble.

5.2 Traffic Simulator: An Online Test-Bed

The DIVERT [Conceicao et al., 2008] is a high-performance traffic simulator framework which uses a realistic microscopic mobility model. The main advantage of this framework when facing others is the easiness to create new simulation modules efficiently. Hence, we have created a new model that simulates the real behavior of a taxi fleet. Upon a request, a central entity elects one taxi to do the requested service. Once the service is finished, the same entity recommends a new taxi-stand for the taxi to go to and wait for a new service.

This framework was employed as an online test-bed for our Recommendation System. Firstly, the realistic map of the city of Porto - containing the real road network topology and the exact location of the 63 taxi stands in the city – was loaded. Secondly, we fed the framework with a service log (i.e. a time-dependent origin-destination matrix) correspondent to the studied period. However, we just accessed the log of one out of the two running fleets in Porto (the largest one, with 441 vehicles). To simulate a scenario similar to our own, we divided this fleet into two using a ratio close to real one (60% for the fleet A1 and 40% to the fleet B1). The services dispatched from the central were also divided in the same proportion while the services demanded in each taxi stand will be the same. The fleet B1 will use the most common and traditional way to choose the best taxi-stand: it will go to the nearest taxi stand of each drop-off location (i.e. after a drop-off, each driver has to head to a specific taxi stand of its own choice). However, the fleet A1 will use our Recommendation System to do an informed driving, which considers multiple

<table>
<thead>
<tr>
<th>Daytype Group</th>
<th>Total Services</th>
<th>Averaged Service Demand per Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Emerged</td>
<td>0am to 6am</td>
</tr>
<tr>
<td>Workdays</td>
<td>957265</td>
<td>935</td>
</tr>
<tr>
<td>Weekends</td>
<td>226504</td>
<td>947</td>
</tr>
<tr>
<td>All Daytypes</td>
<td>1380153</td>
<td>1029</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Services per Driver</th>
<th>Total Cruise Time (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum</td>
<td>6751</td>
</tr>
<tr>
<td>Minimum</td>
<td>100</td>
</tr>
<tr>
<td>Mean</td>
<td>2679</td>
</tr>
<tr>
<td>Std. Dev.</td>
<td>1162</td>
</tr>
</tbody>
</table>
variables – like the number of taxis in each stand or the demand prediction on them - to support this important decision. Finally, we ran the simulation and we extract the metrics for each fleet. The framework is used to calculate the optimal paths between the taxi stand and the passenger location and the dependent behavior of the fleets (the location of each vehicle will affect the way they get the services). Our main goal is to simulate a real scenario behavior and its competitive characteristics while we are testing the Recommendation System. It is important to notice that both fleets would get similar results if they did not use any Recommendation System. We also highlight that the vehicles will remain parked in the stand waiting for a service whenever the time it takes to appear. In this case, we consider the maximum threshold of 120 minutes that is deeply detailed in the following section, along with the remaining evaluation metrics.

### 5.3 Evaluation Methods

We used the data obtained from the last four months to evaluate our both experimental setups (where 506873 services emerged). Firstly, we present two error measurements which were employed to evaluate our output: one from the literature and another from our own specifically adapted to our current problem. Secondly, we detail the two performance metrics used to evaluate our recommendation models.

Consider $R_k = \{R_{k,0}, R_{k,1}, \ldots, R_{k,t}\}$ to be a discrete time series (aggregation period of 15 minutes) with the number of services predicted for a taxi stand of interest $k$ in the period $[1, t]$ and $X_k = \{X_{k,0}, X_{k,1}, \ldots, X_{k,t}\}$ the number of services actually emerged in the same conditions. The (1) Symmetric Mean Percentage Error ($sMAPE$) is a well-known metric to evaluate the success of time series forecast models. However, this metric can be too intolerant with small magnitude errors (e.g. if two services are predicted on a given period for a taxi stand of interest but no one actually emerges, the error within that period would be 1). Then, we propose to also use an adapted version of Normalized Mean Absolute Error (NMAE).

The (2) Average Weighted Error ($AVE$) is a metric of our own based on the NMAE. We defined it as

\[
AVE' = \frac{\sum_{i=1}^{t} \theta_{k,i} \cdot X_{k,i}}{\sum_{i=1}^{t} \sigma_{k,i} \cdot \psi_k}
\]

where $\psi_k$ is the total of services emerged at the taxi stand $k$ during the time period $[1, t]$. The main feature about this metric is to weight the error in each period by the number of real events actually emerged (i.e. the errors on periods where more services were actually demanded are more relevant than the remaining ones).

Both metrics are focused just on one time series for a given taxi stand. However, the results presented below use an averaged error measured based on all stands series – $GA$. Consider $\beta$ to be an error metric of interest. $AG_{\beta}$ is an aggregated metric given by a weighted average of the error in all stands. It is formally presented in the following equation.

\[
AG_{\beta} = \frac{\sum_{k=1}^{N} G_{k,\beta,k} \cdot \psi_k}{\mu}, \quad \mu = \sum_{k=1}^{N} \psi_k
\]

We considered three performance metrics in the evaluation of our recommendation models: (1) the Waiting Time (WT) and (2) the Vacant Running Distance (VRD) and the number of No Services (NS). The Waiting Time is the total time that a driver takes between a drop-off and a pick-up (i.e. to leave a stand with a passenger or to get one in his/her current location). The Vacant Running Distance is the distance that a driver does to get into a stand after a drop-off (i.e.: without any passenger inside). Independently on the time measured on the simulation, we always consider a maximum threshold of 120 minutes to the Waiting Time. The No Service metric is a ratio between the number of times that a taxi parked on a stand had a waiting time greater than the 120 minutes threshold and the number of services effectively dispatched by the respective fleet.

### 5.4 Results

Firstly, we present the results obtained by the online experiments done with the predictive models. The error measured for each model is highlighted in Table 3 and Table 4. The results are firstly presented per shift and then globally. These error values were aggregated using the $AG_{\beta}$ previously defined.

Secondly, the values calculated for our performance metrics using the traffic simulator previously described are detailed in the Table 5. The fleet $A1$ used the Recommendation Model 1 (RS1) while the $B1$ uses the common expected behavior (previously defined). Distinct metrics values are presented for the two using different aggregations like the arithmetic mean (i.e. average). The No Services ratio is also displayed.

#### 6 Final Remarks

In this paper, we present a novel application of time series forecasting techniques to improve the taxi driver mobility intelligence. We did it in three distinct steps: firstly (1) we mined both GPS and event signals emitted by a company operating in Porto, Portugal (where the passenger demand is

<table>
<thead>
<tr>
<th>Model</th>
<th>00h–08h</th>
<th>08h–16h</th>
<th>16h–00h</th>
<th>24h</th>
</tr>
</thead>
<tbody>
<tr>
<td>Poisson Mean</td>
<td>21.28%</td>
<td>24.88%</td>
<td>22.88%</td>
<td>23.43%</td>
</tr>
<tr>
<td>W. Poisson Mean</td>
<td>23.32%</td>
<td>28.37%</td>
<td>26.77%</td>
<td>26.74%</td>
</tr>
<tr>
<td>ARIMA</td>
<td>20.85%</td>
<td>26.12%</td>
<td>22.92%</td>
<td>20.91%</td>
</tr>
<tr>
<td>Ensemble</td>
<td>14.37%</td>
<td>18.18%</td>
<td>17.19%</td>
<td>15.89%</td>
</tr>
</tbody>
</table>

Table 3: Error Measured on the Models using $AVE$
Table 4: Error Measured on the Models using \(sMAPE\)

<table>
<thead>
<tr>
<th>Periods</th>
<th>Poisson Mean</th>
<th>W. Poisson Mean</th>
<th>ARIMA</th>
<th>Ensemble</th>
</tr>
</thead>
<tbody>
<tr>
<td>00h−08h</td>
<td>15.09%</td>
<td>17.32%</td>
<td>16.81%</td>
<td>14.37%</td>
</tr>
<tr>
<td>08h−16h</td>
<td>19.20%</td>
<td>20.66%</td>
<td>18.59%</td>
<td>18.18%</td>
</tr>
<tr>
<td>16h−00h</td>
<td>17.51%</td>
<td>19.88%</td>
<td>17.85%</td>
<td>17.19%</td>
</tr>
<tr>
<td>24h</td>
<td>16.84%</td>
<td>18.47%</td>
<td>18.51%</td>
<td>15.89%</td>
</tr>
</tbody>
</table>

lower than the vacant taxis). Secondly, we predicted - in a real-time experiment - the distribution of the taxi-passerger demand for the 63 taxi stands at 30-minute period intervals. Finally, we recreated the scenario running in Porto, where two fleets (the fleet A and B, which contain 441 and 250 vehicles, respectively) compete to get as many services as possible. We did it using a traffic simulation framework fed by the real services historical log of the largest operating fleet.

One of the fleets used our Recommendation System for the Taxi Stand choice problem while the other one just picked the stand using a baseline model corresponding to the driver common behavior in similar situations.

Our predictive model demonstrated a more than satisfactory performance, anticipating in real time the spatial distribution of the passenger demand with an error of just 20%. We believe that this model is a true novelty and a major contribution to the area through its online adapting characteristics:

- It takes advantage of the ubiquitous characteristics of a taxi communicational network, assembling the experience and the knowledge of all vehicles/drivers while they usually use just their own;
- It simultaneously uses long-term, mid-term and short term historical data as a learning base;
- It rapidly produces real-time short-term predictions of the demand, which can truly improve drivers’ mobility intelligence and consequently, their profit.

This approach meets no parallel in the literature also by its test-bed: the models were tested in a streaming environment, while the state-of-art presents mainly offline experimental setups. Our simulation results demonstrated that such informed driving can truly improve the drivers’ mobility intelligence: the fleet A1 had an Average Waiting Time 5% lower than its competitor – even if it has a larger fleet. We also highlight the reduction of the No Service ratio in 50% while the Vacant Running Time faced an increase. It is important to state that this Recommendation System is focused on a scenario like our own – two or more competitors operating in a medium/large city where the demand is lower than the number of running vehicles. Its main goal is to recommend a stand where a service will rapidly emerge – even if this stand is far away. The idea is to be in a position able to pick-up the emerging service demand before the remaining competition. This factor can provoke a slight increase on the Vacant Running Time but it will also reduce the usually large Waiting Times to pick-up passengers. Other scenarios may require a distinct calibration of the model to account different needs/goals.

Acknowledgments

The authors would like to thank to Geolink and to its team for the data supplied to this work. This work was supported by the projects DRIVE-IN: "Distributed Routing and Infotainment through Vehicular Internet-working", VTL: “Virtual Traffic Lights” and KDUS: "Knowledge Discovery from Ubiquitous Data Streams” under the Grants CMU-PT/NGN/0052/2008, PTDC/EIA-CCO/118114/2010, PTDC/EIA-EIA/093855/2008, respectively, and also by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness), by the Portuguese Funds through the FCT(Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-022701.

References


[Kaltenbrunner et al., 2010] Andreas Kaltenbrunner, Rodrigo Meza, Jens Grivolla, Joan Codina, and Rafael


Visual Scenes Clustering Using Variational Incremental Learning of Infinite Generalized Dirichlet Mixture Models

Wentao Fan
Electrical and Computer Engineering
Concordia University, Canada
wenta.fa@encs.concordia.ca

Nizar Bouguila
Institute for Information Systems Engineering
Concordia University, Canada
nizar.bouguila@concordia.ca

Abstract
In this paper, we develop a clustering approach based on variational incremental learning of a Dirichlet process of generalized Dirichlet (GD) distributions. Our approach is built on nonparametric Bayesian analysis where the determination of the complexity of the mixture model (i.e. the number of components) is sidestepped by assuming an infinite number of mixture components. By leveraging an incremental variational inference algorithm, the model complexity and all the involved model’s parameters are estimated simultaneously and effectively in a single optimization framework. Moreover, thanks to its incremental nature and Bayesian roots, the proposed framework allows to avoid over- and under-fitting problems, and to offer good generalization capabilities. The effectiveness of the proposed approach is tested on a challenging application involving visual scenes clustering.

1 Introduction
Incremental clustering plays a crucial role in many data mining and computer vision applications [Opelt et al., 2006; Sheikh et al., 2007; Li et al., 2007]. Incremental clustering is particularly efficient in the following scenarios: when data points are obtained sequentially, when the available memory is limited, or when we have large-scale data sets to deal with. Bayesian approaches have been widely used to develop powerful clustering techniques. Bayesian approaches applied for incremental clustering fall basically into two categories: parametric and non-parametric, and allow to mimic the human learning process which is based on iterative accumulation of knowledge. As opposed to parametric approaches in which a fixed number of parameters is considered, Bayesian non-parametric approaches use an infinite-dimensional parameter space and allow the complexity of models to grow with data size. The consideration of an infinite-dimensional parameter space allows to determine appropriate model complexity, which is normally referred to as the problem of model selection or model adaptation. This is a crucial issue in clustering since it permits to capture the underlying data structure more precisely, and also to avoid over- and under-fitting problems. This paper focuses on the latter one since it is more adapted to modern data mining applications (i.e. modern applications involve generally dynamic data sets).

Nowadays, the most popular Bayesian nonparametric formalism is the Dirichlet process (DP) [Neal, 2000; Teh et al., 2004] generally translated to a mixture model with a countably infinite number of components in which the difficulty of selecting the appropriate number of clusters, that usually occurs in the finite case, is avoided. A common way to learn Dirichlet process model is through Markov chain Monte Carlo (MCMC) techniques. Nevertheless, MCMC approaches have several drawbacks such as the high computational cost and the difficulty of monitoring convergence. These shortcomings of MCMC approaches can be solved by adopting an alternative namely variational inference (or variational Bayes) [Attias, 1999], which is a deterministic approximation technique that requires a modest amount of computational power. Variational inference has provided promising performance in many applications involving mixture models [Corduneanu and Bishop, 2001; Constantinopoulos et al., 2006; Fan et al., 2012; 2013]. In our work, we employ an incremental version of variational inference proposed by [Gomes et al., 2008] to learn infinite generalized Dirichlet (GD) mixtures in the context where data points are supposed to arrive sequentially. The consideration of the GD distribution is motivated by its promising performance when handling non-Gaussian data, and in particular proportional data (which are subject to two restrictions: nonnegativity and unit-sum) which are naturally generated in several data mining, machine learning, computer vision, and bioinformatics applications [Bouguila and Ziou, 2006; 2007; Boutemedjet et al., 2009]. Examples of applications include textual documents (or images) clustering where a given document (or image) is described as a normalized histogram of words (or visual words) frequencies.

The main contributions of this paper are listed as the following: 1) we develop an incremental variational learning algorithm for the infinite GD mixture model, which is much more efficient when dealing with massive and sequential data as opposed to the corresponding batch approach; 2) we apply the proposed approach to tackle a challenging real-world problem namely visual scenes clustering. The effectiveness and merits of our approach are illustrated through extensive simulations. The rest of this paper is organized as follows. Section 2 presents the infinite GD mixture model. The incre-
mental variational inference framework for model learning is described in Section 3. Section 4 is devoted to the experimental results. Finally, conclusion follows in Section 5.

2 The Infinite GD Mixture Model

Let \( \bar{Y} = (Y_1, \ldots, Y_D) \) be a D-dimensional random vector drawn from an infinite mixture of GD distributions:

\[
p(\bar{Y}|\bar{\alpha}, \bar{\beta}) = \sum_{j=1}^{\infty} \pi_j \text{GD}(\bar{Y}|\bar{\alpha}_j, \bar{\beta}_j)
\]

where \( \bar{\alpha} \) represents the mixing weights that are positive and sum to one. \( \bar{\alpha}_j = (\alpha_{j1}, \ldots, \alpha_{jD}) \) and \( \bar{\beta}_j = (\beta_{j1}, \ldots, \beta_{jD}) \) are the positive parameters of the GD distribution associated with component \( j \), while GD(\( \bar{Y}|\bar{\alpha}_j, \bar{\beta}_j \)) is defined as

\[
\text{GD}(\bar{Y}|\bar{\alpha}_j, \bar{\beta}_j) = \prod_{l=1}^{D} \frac{\Gamma(\alpha_{jl} + \beta_{jl})}{\Gamma(\alpha_{jl})\Gamma(\beta_{jl})} (1 - \sum_{k=1}^{l} Y_k)^{\alpha_{jl} - 1} (1 - \sum_{k=1}^{D} Y_k)^{\beta_{jl} - 1}
\]

where \( \sum_{l=1}^{D} Y_l < 1 \) and \( 0 < y_l < 1 \) for \( l = 1, \ldots, D \), \( \gamma_{jl} = \beta_{jl} - \alpha_{jl+1} - \beta_{jl+1} \) for \( l = 1, \ldots, D - 1 \), and \( \gamma_{jD} = \beta_{jD} - 1 \). \( \Gamma(\cdot) \) is the gamma function defined by \( \Gamma(x) = \int_{0}^{\infty} u^{x-1}e^{-u}du \). Furthermore, we exploit an interesting and convenient mathematical property of the GD distribution which is thoroughly discussed in [Boutemedjet et al., 2009], to transform the original data points into another D-dimensional space where the features are conditionally independent and rewrite the infinite GD mixture model in the following form

\[
p(\bar{X}|\bar{\alpha}, \bar{\beta}) = \sum_{j=1}^{\infty} \pi_j \prod_{i=1}^{D} \text{Beta}(X_i|\alpha_{jl}, \beta_{jl})
\]

where \( X_i = Y_i \) and \( X_i = Y_j / (1 - \sum_{k=1}^{l-1} Y_k) \) for \( l > 1 \). Beta(\( X_i|\alpha_{jl}, \beta_{jl} \)) is a Beta distribution parameterized with \( (\alpha_{jl}, \beta_{jl}) \).

In this work, we construct the Dirichlet process through a stick-breaking representation [Sethuraman, 1994]. Therefore, the mixing weights \( \pi_j \) are constructed by recursively breaking a unit length stick into an infinite number of pieces as \( \pi_j = \lambda_j \prod_{k=1}^{j-1} (1 - \lambda_k) \), \( \lambda_j \) is known as the stick breaking variable and is distributed independently according to \( \lambda_j \sim \text{Beta}(1, \xi) \), where \( \xi > 0 \) is the concentration parameter of the Dirichlet process.

For an observed data set \( \{\bar{X}_1, \ldots, \bar{X}_N\} \), we introduce a set of mixture component assignment variables \( \bar{Z} = (Z_1, \ldots, Z_N) \), one for each data point. Each element \( Z_i \) of \( \bar{Z} \) has an integer value \( j \) specifying the component from which \( \bar{X}_i \) is drawn. The marginal distribution over \( \bar{Z} \) is given by

\[
p(\bar{Z}|\bar{\alpha}, \bar{\beta}) = \prod_{i=1}^{N} \prod_{j=1}^{M} \lambda_j^{1[Z_i = j]} (1 - \lambda_j)^{\sum_{k=1}^{j-1} (1 - \lambda_k)}
\]

where \( 1[\cdot] \) is an indicator function which equals to 1 when \( Z_i = j \), and equals to 0 otherwise. Since our model framework is Bayesian, we need to place prior distributions over random variables \( \bar{\alpha} \) and \( \bar{\beta} \). Since the formal conjugate prior for Beta distribution is intractable, we adopt Gamma priors \( \mathcal{G}(\cdot) \) to approximate the conjugate priors of \( \bar{\alpha} \) and \( \bar{\beta} \) as:

\[
p(\bar{\alpha}) = \mathcal{G}(\bar{\alpha}|\bar{u}, \bar{v}) \quad \text{and} \quad p(\bar{\beta}) = \mathcal{G}(\bar{\beta}|\bar{\xi}, \bar{\tau})
\]

with the assumption that these parameters are statistically independent.

3 Model Learning

In our work, we adopt an incremental learning framework proposed in [Gomes et al., 2008] to learn the proposed infinite GD mixture model through variational Bayes. In this algorithm, data points can be sequentially processed in small batches where each one may contain one or a group of data points. The model learning framework involves the following two phases: 1) model building phase: to infer the optimal mixture model with the currently observed data points; 2) compression phase: to estimate which mixture component that groups of data points should be assigned to.

3.1 Model Building Phase

For an observed data set \( X = (\bar{X}_1, \ldots, \bar{X}_N) \), we define \( \Theta = \{\bar{Z}, \bar{\alpha}, \bar{\beta}, \bar{\lambda}\} \) as the set of unknown random variables. The main target of variational Bayes is to estimate a proper approximation \( q(\Theta) \) for the true posterior distribution \( p(\Theta|X) \). This problem can be solved by maximizing the free energy \( F(X, q) \), where \( F(X, q) = \int q(\Theta) \ln[p(X|\Theta)/q(\Theta)]d\Theta \). In our algorithm, inspired by [Blei and Jordan, 2005], we truncate the variational distribution \( q(\Theta) \) at a value \( M \), such that \( \lambda_M = 1, \pi_j = 0 \) when \( j > M \), and \( \sum_{j=1}^{M} \pi_j = 1 \), where the truncation level \( M \) is a variational parameter which can be freely initialized and will be optimized automatically during the learning process [Blei and Jordan, 2005]. In order to achieve tractability, we also assume that the approximated posterior distribution \( q(\Theta) \) can be factorized into disjoint tractable factors as:

\[
q(\Theta) = \prod_{i=1}^{N} q(Z_i) \prod_{j=1}^{M} q(\alpha_{jl}|\bar{u}_j, v_j) \prod_{j=1}^{M} q(\beta_{jl}|\bar{\xi}_j, \bar{\tau}_j)
\]

By maximizing the free energy \( F(X, q) \) with respect to each variational factor, we can obtain the following update equations for these factors:

\[
q(\bar{\lambda}) = \prod_{i=1}^{N} \prod_{j=1}^{M} \frac{\Gamma(\alpha_{jl} + \beta_{jl})}{\Gamma(\alpha_{jl})\Gamma(\beta_{jl})} (1 - \sum_{k=1}^{j-1} Y_k)^{\alpha_{jl} - 1} (1 - \sum_{k=1}^{D} Y_k)^{\beta_{jl} - 1}
\]

\[
q(\bar{\beta}) = \prod_{j=1}^{M} \prod_{l=1}^{D} \frac{\Gamma(\alpha_{jl} + \beta_{jl})}{\Gamma(\alpha_{jl})\Gamma(\beta_{jl})} (1 - \sum_{k=1}^{j-1} Y_k)^{\alpha_{jl} - 1} (1 - \sum_{k=1}^{D} Y_k)^{\beta_{jl} - 1}
\]

where we have defined

\[
r_{ij} = \prod_{j=1}^{M} \frac{\exp(\rho_{ij})}{\sum_{j=1}^{M} \exp(\rho_{ij})}
\]

\[
\rho_{ij} = \sum_{i=1}^{D} [\bar{\lambda}_j + (\alpha_{jl} - 1) \ln X_{ii} + (\beta_{jl} - 1) \ln(1 - X_{ii})]
\]

\[
+ \langle \ln \bar{\lambda}_j \rangle + \sum_{k=1}^{M} \langle \ln(1 - \lambda_k) \rangle
\]

\[
u_{ij}^* = v_{ij} + \sum_{i=1}^{N} \langle Z_i = j \rangle |\Psi(\bar{\alpha}_{jl} + \bar{\beta}_{jl}) - \Psi(\bar{\alpha}_{jl}) + \bar{\beta}_{jl}|
\]
where $\Psi(\cdot)$ is the digamma function, and $\langle \cdot \rangle$ is the expectation evaluation. Note that, $\tilde{R}$ is the lower bound of $R = \langle \ln(1 + \frac{1}{N}) \rangle$. Since this expectation is intractable, the second-order Taylor series expansion is applied to find its lower bound. The expected values in the above formulas are given by $\langle Z_l = j \rangle = r_{lj}$, $a_{lj} = \langle \alpha_{lj} \rangle = a_{lj}^*/v_{lj}$, $\hat{\beta}_{lj} = (\hat{\beta}_{lj}^*) = s_{lj}^*/v_{lj}$, $\langle \ln \lambda_l \rangle = \Psi(a_{lj}) - \Psi(a_{lj} + b_{lj})$, $\langle \ln(1 - \lambda_l) \rangle = \Psi(b_{lj}) - \Psi(a_{lj} + b_{lj})$, $\langle \ln \alpha_{lj} \rangle = \Psi(a_{lj}^*) - \ln v_{lj}$, and $\langle \ln \beta_{lj} \rangle = \Psi(\hat{\beta}_{lj}^*) - \ln v_{lj}^*$. After convergence, the currently observed data points are clustered into $M$ groups according to corresponding responsibilities $r_{lj}$ through Eq. (7). According to [Gomes et al., 2008], these newly formed groups of data points are also denoted as “clumps”. Following [Gomes et al., 2008], these clumps are subject to the constraint that all data points $\tilde{X}_c$ in the clump $c$ share the same $\theta(Z_c) \equiv \theta^*(Z_c)$ which is a key factor in the following compression phase.

Algorithm 1

1: Choose the initial truncation level $M$.
2: Initialize the values for hyper-parameters $a_{lj}$, $v_{lj}$, $s_{lj}$, $t_{lj}$ and $\xi_j$.
3: Initialize the values of $r_{lj}$ by K-Means algorithm.
4: while More data to be observed do
5: Perform the model building phase through Eqs. (5) and (6).
6: Initialize the compression phase using Eq. (10).
7: while $MC \geq C$ do
8: for $j = 1$ to $M$ do
9: if evaluated($j$) = false then
10: Split component $j$ and refine this split using Eqs (9).
11: $\Delta F(j)$ = change in Eq. (8).
12: evaluated($j$) = true.
13: end if
14: end for
15: Split component $j$ with the largest value of $\Delta F$. $j$.
16: $M = M + 1$.
17: end while
18: Discard the current observed data points.
19: Save resulting components into next learning round.
20: end while

3.2 Compression Phase

Within the compression phase, we need to estimate clumps that are possibly belong to the same mixture component while taking into consideration future arriving data. Now assume that we have already observed $N$ data points, our aim is to make an inference at some target time $T$ where $T \geq N$. We can tackle this problem by scaling the observed data to the target size $T$, which is equivalent to using the variational posterior distribution of the observed data $N$ as a predictive model of the future data [Gomes et al., 2008]. We then have a modified free energy for the compression phase in the following form

$$
\Phi = \sum_{j=1}^{M} \sum_{l=1}^{D} \left[ \ln \frac{p(\alpha_{lj} | s_{lj}, v_{lj})}{q(\alpha_{lj})} + \ln \frac{p(\beta_{lj} | s_{lj}, t_{lj})}{q(\beta_{lj})} \right] + \sum_{j=1}^{M} \sum_{l=1}^{D} \left[ \ln \frac{p(\lambda_l | s_{lj}, v_{lj})}{q(\lambda_l)} + \ln \frac{\theta(\lambda_l | s_{lj}, v_{lj})}{\theta(\lambda_l)} \right]
$$

where $|c|$ represents the number of data points in clump $c$ and $T$ is the data magnification factor. The corresponding update equations for maximizing this free energy function can be obtained as

$$
r_{cj} = \exp(\rho_{cj}) / \sum_{j=1}^{C} \exp(\rho_{cj})
$$

$$
\rho_{cj} = \sum_{i=1}^{D} \left[ \ln \lambda_i + \ln(1 - \lambda_i) \right] + \sum_{l=1}^{N} \left[ \ln(1 - \lambda_l) \right] - \theta(\lambda_l | s_{lj}, v_{lj})
$$

$$
s_{lj} = s_{lj} + \frac{T}{N} \sum_{c} \left[ n_c | c | \theta(\alpha_{lj} + \beta_{lj}) - \theta(\alpha_{lj} + \beta_{lj}) \right]
$$

$$
t_{lj} = t_{lj} - \frac{T}{N} \sum_{c} \left[ n_c | c | \ln(1 - \theta(\lambda_l | s_{lj}, v_{lj})) \right]
$$

$$
\xi_j = \exp(\rho_{cj}) / \sum_{j=1}^{C} \exp(\rho_{cj})
$$

where $|c|$ denotes average over all data points contained in clump $c$. The first step of the compression phase is to assign each clump or data point to the component with the highest responsibility $r_{cj}$ calculated from the model building phase as

$$
I_c = \arg \max r_{cj}
$$

where $\{I_c\}$ denote which component the clump (or data point) $c$ belongs to in the compression phase. Next, we cycle through each component and split it along its principal component into two subcomponents. This split is refined by updating Eqs. (9). The clumps are then hard assigned to one
of the two candidate components after convergence for refining the split. Among all the potential splits, we select the one that results in the largest change in the free energy (Eq. (8)). The splitting process repeats itself until a stopping criterion is met. According to [Gomes et al., 2008], the stopping criterion for the splitting process can be expressed as a limit on the amount of memory required to store the components. In our case, the component memory cost for the mixture model is $MC = 2DN_c$, where $2D$ is the number of parameters contained in a $D$-variate GD component, and $N_c$ is the number of components. Accordingly, we can define an upper limit on the component memory cost $C$, and the compression phase stops when $MC \geq C$. As a result, the computational time and the space requirement is bounded in each learning round. After the compression phase, the currently observed data points are discarded while the resulting components can be treated in the same way as data points in the next round of leaning. Our incremental variational inference algorithm for infinite GD mixture model is summarized in Algorithm 1.

![Sample images from the OT data set.](image)

Figure 1: Sample images from the OT data set.

## 4 Visual Scenes Clustering

In this section, the effectiveness of the proposed incremental infinite GD mixture model ($InGDM$) is tested on a challenging real-world application namely visual scenes clustering. The problem is important since images are being produced at exponential increasing rates and very challenging due to the difficulty of capturing the variability of appearance and shape of diverse objects belonging to the same scene, while avoiding confusing objects from different scenes. In our experiments, we initialize the truncation level $M$ as 15. The initial values of the hyperparameters are set as: $(u_{ij}, v_{ij}, s_{ij}, t_{ij}, \xi_j) = (1, 0.01, 1, 0.01, 0.1)$, which have been found to be reasonable choices according to our experimental results.

### 4.1 Database and Experimental Design

In this paper, we test our approach on a challenging and publicly available database known as the OT database, which was introduced by Oliva and Torralba [Oliva and Torralba, 2001]. This database contains 2,688 images with the size of $256 \times 256$ pixels, and is composed of eight urban and natural scene categories: coast (360 images), forest (328 images), highway (260 images), inside-city (308 images), mountain (374 images), open country (410 images), street (292 images), and tall building (356 images). Figure 1 shows some sample images from the different categories in the OT database.

<table>
<thead>
<tr>
<th>Category</th>
<th>C</th>
<th>F</th>
<th>H</th>
<th>I</th>
<th>M</th>
<th>O</th>
<th>S</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>Coast (C)</td>
<td>127</td>
<td>10</td>
<td>4</td>
<td>2</td>
<td>3</td>
<td>31</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Forest (F)</td>
<td>2</td>
<td>155</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Highway (H)</td>
<td>0</td>
<td>0</td>
<td>122</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>Inside-city (I)</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>119</td>
<td>3</td>
<td>2</td>
<td>15</td>
<td>7</td>
</tr>
<tr>
<td>Mountain (M)</td>
<td>6</td>
<td>21</td>
<td>4</td>
<td>5</td>
<td>139</td>
<td>9</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>Open country (O)</td>
<td>2</td>
<td>22</td>
<td>19</td>
<td>15</td>
<td>9</td>
<td>131</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Street (S)</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>8</td>
<td>5</td>
<td>5</td>
<td>122</td>
<td>1</td>
</tr>
<tr>
<td>Tall building (T)</td>
<td>4</td>
<td>9</td>
<td>7</td>
<td>23</td>
<td>3</td>
<td>19</td>
<td>3</td>
<td>110</td>
</tr>
</tbody>
</table>

Table 1: Average rounded confusion matrix for the OT database calculated by $InGDM$.

### 4.2 Experimental Results

In our experiments, we randomly divided the OT database into two halves: one for constructing the visual vocabulary, another for testing. Since our approach is unsupervised, the class labels are not involved in our experiments, except for evaluation of the clustering results. The entire methodology was repeated 30 times to evaluate the performance. For comparison, we have also applied three other mixture-modeling approaches: the finite GD mixture model ($FiGDM$), the infinite Gaussian mixture model ($InGMM$) and the finite Gaussian mixture model ($FiGMM$). To make a fair comparison, all of the aforementioned approaches are learned through incremental variational inference. Table 1 shows the average confusion matrix of the OT database calculated by the proposed $InGDM$. Table 2 illustrates the average categorization performance using different approaches for the OT database. As we can see from this table, it is obvious that our approach ($InGDM$) provides the best performance in
terms of the highest categorization rate (77.47%) among all the tested approaches. In addition, we can observe that better performances are obtained for approaches that adopt the infinite mixtures (InGDMM and InGMM) than the corresponding finite mixtures (FiGDMM and FiGMM), which demonstrate the advantage of using infinite mixture models over finite ones. Moreover, according to Table 2, GD mixture has higher performance than Gaussian mixture which verifies that the GD mixture model has better modeling capability than the Gaussian for proportional data clustering.

### 5 Conclusion

In this work, we have presented an incremental nonparametric Bayesian approach for clustering. The proposed approach is based on infinite GD mixture models with a Dirichlet process framework, and is learned using an incremental variational inference framework. Within this framework, the model parameters and the number of mixture components are determined simultaneously. The effectiveness of the proposed approach has been evaluated on a challenging application namely visual scenes clustering. Future works could be devoted to the application of the proposed algorithm for other data mining tasks involving continually changing or growing volumes of proportional data.

### References

Simultaneous segmentation and recognition of gestures for human-machine interaction

Harold Vasquez, L. EnriqueSucar, Hugo Jair Escalante
Department of Computational Sciences
Instituto Nacional de Astrofísica, Óptica y Electrónica,
Tonantzintla, 72840, Puebla, Mexico.
{hvasquez, esucar, hugojair}@inaoep.mx

Abstract

Human-activity and gesture recognition are two problems lying at the core of human-centric and ubiquitous systems: knowing what activities/gestures users are performing allows systems to execute actions accordingly. State-of-the-art technology from computer vision and machine intelligence allow us to recognize gestures at acceptable rates when gestures are segmented (i.e., each video contains a single gesture). In ubiquitous environments, however, continuous video is available and thus systems must be capable of detecting when a gesture is being performed and recognizing it. This paper describes a new method for the simultaneous segmentation and recognition of gestures from continuous videos. A multi-window approach is proposed in which predictions of several recognition models are combined; where each model is evaluated using a different segment of the continuous video. The proposed method is evaluated in the problem of recognition of gestures to command a robot. Preliminary results show the proposed method is very effective for recognizing the considered gestures when they are correctly segmented; although there is still room for improvement in terms of its segmentation capabilities. The proposed method is highly efficient and does not require learning a model for no-gesture, as opposed to related methods.

1 Introduction

Human-computer interaction technology plays a key role in ubiquitous data mining (i.e., the extraction of interesting patterns from data generated in human-centric environments), see [Eunju, 2010]. From all of the alternative forms of interaction, gestures are among the most natural and intuitive for users. In fact, gestures are widely used to complement verbal communication between humans. Research advances in computer vision and machine learning have lead to the development of gesture recognition technology that is able to recognize gestures at very acceptable rates [Aggarwal and Ryoo, 2011; Mitra, 2007]. However, most of the available methods for gesture recognition require gestures to be segmented before the recognition process begins [Aviles et al., 2011]. Clearly, this type of methods is not well suited for ubiquitous systems (and real applications in general), where the recognition of gestures must be done from a continuous video in real time [Eunju, 2010; Huynh et al., 2008].

This paper introduces a new approach for the simultaneous segmentation and recognition of gestures in continuous video. The proposed method implements a voting strategy using the predictions obtained from multiple gesture models evaluated at different time-windows, see Figure 1. Windows are dynamically created by incrementally scanning the continuous video. When the votes from the multiple models favor a particular gesture, we segment the video and make a prediction: we predict the gesture corresponding to the model that obtained the majority of votes across windows.

We use as features the body-part positions obtained by a Kinect\textsuperscript{TM} sensor. As predictive model we used Hidden Markov Models (HMMs), one of the most used for gesture recognition [Aviles et al., 2011; Aggarwal and Ryoo, 2011; Mitra, 2007]. The proposed method is evaluated in the problem of recognition of gestures to command a robot. Preliminary results show the proposed method is very effective for recognizing the considered gestures when they are correctly segmented. However, there is still room for improvement in terms of its segmentation capabilities. The proposed method is highly efficient and does not require learning a model for no-gesture, as opposed in related works.

The rest of this paper is organized as follows. The next section briefly reviews related works on gesture spotting. Section 3 describes the proposed approach. Section 4 reports experimental results that show evidence of the performance of proposed technique. Section 5 outlines preliminary conclusions and discusses future work direction.

2 Related work

Several methods for the simultaneous segmentation and recognition of gestures (a task also known as gesture spotting) have been proposed so far [Derpanis et al., 2010; Yuan et al., 2009; Malgireddy et al., 2012; Kim et al., 2007; Yang et al., 2007]. Some methods work directly with spatio-temporal patterns extracted from video [Derpanis et al., 2010; Yuan et al., 2009]. Although being effective, these methods
are very sensitive to changes in illumination, scale, appearance and viewpoint.

On the other hand, there are model-based techniques that use the position of body-parts to train probabilistic models (e.g., HMMs) [Aggarwal and Ryoo, 2011; Mitra, 2007]. In the past, these type of methods were limited because of the need of specialized sensors to obtain body-part positions. Nowadays, the availability of Kinect\textsuperscript{TM} (which can extract skeleton information in real time) has partially circumvented such limitation [Webb and Ashley, 2012].

Besides the data acquisition process, some of these methods require the construction of a no-gesture model (e.g., [Kim et al., 2007]) or transition-gesture model (e.g., [Yang et al., 2007]). The goal of such models is to determine within a video when the user (if any) is not performing any gesture or the transition between different gestures. Building a model for no-gesture is a complicated and subjective task that depends on the particular application where the gesture recognition system is to be implemented [Kim et al., 2007]. In ubiquitous systems, however, we want gesture recognition methods to work in very general conditions and under highly dynamic environments. Hence, a model for no-gesture is much more complicated to generate in these conditions.

Finally, it is worth to mention that many of the available techniques for gesture spotting can be very complex to implement. This is a particularly important aspect to consider for some domains, for example in mobile devices and/or for human-robot interaction; where there are limited resources and restricted programming tools for the implementation of algorithms. Thus, sometimes simplicity is preferred at the expense of loosing a little bit in precision in these domains.

The method we propose in this paper performs segmentation and recognition of gestures simultaneously and attempts to address the limitations of most of the available techniques. Specifically, our proposal is efficient and very simple to implement; it is robust, to some extend, to problems present in appearance-based methods; and, more importantly, does not require the specification of a no-gesture model.

3 Multiple-windows approach

We face the problem of simultaneously segmenting and recognizing gestures in continuous video\footnote{Although we use (processed) body-part positions as features, we refer to the sequence of these features as video. This is in order to simplify explanations.}. That is, given a sequence of images (video) we want to determine where a gesture is being performed (independently of the type of gesture) and next to recognize what is the actual gesture being performed. We propose a solution based on multiple windows that are incrementally and dynamically created. Each window is passed through predictive models each trained to recognize a particular gesture. The predictions of models for different windows are accumulated, when the model for a particular gesture is extended by \( \Delta \) time units. That is, at the beginning \( W_1 \) is created, then at \( t_1 \), \( W_2 \) is created and \( W_1 \) is extended by \( \Delta \), and so on. At \( t_3 \) there are 3 windows of different size, for each window we estimate the probability of all gestures using HMMs.

As classification model we consider an HMM\footnote{We used the HMM implementation from Matlab\textsuperscript{TM}’s statistics toolbox.}, one of the most popular models for gesture recognition [Aviles et al., 2011; Aggarwal and Ryoo, 2011; Mitra, 2007]. For each gesture \( i \) to be recognized we trained an HMM, let \( M_i \) denote the transitions of no gesture (NG) and gesture, which are identified by the class of gesture \( (G_1, G_2, G_3) \). Below we illustrate a series of windows that are dynamically created and extended each \( \Delta \) time units. That is, at the beginning \( W_1 \) is created, then at \( t_1 \), \( W_2 \) is created and \( W_1 \) is extended by \( \Delta \), and so on. At \( t_3 \) there are 3 windows of different size, for each window we estimate the probability of all gestures using HMMs.

Those of other models will be low. Accumulating predictions allow us to be more confident in that the gesture is being performed within a neighborhood of temporal windows.

The rest of this section describes in detail the proposed technique. First we describe the considered features, next the predictive models and finally the approach to simultaneous segmentation and recognition of gestures.

3.1 Features

We use the information obtained through a Kinect\textsuperscript{TM} as inputs for our gesture spotting method. The Kinect\textsuperscript{TM} is capable of capturing RGB and depth video, as well as the positions of certain body-parts at rates up to 30 frames-per-second (fps). In this work we considered gestures to command a robot that are performed with the hands. Therefore, we used the position of hands as given by Kinect\textsuperscript{TM} as features. For each hand, we obtain per each frame a sextuple indicating the position of both hands in the \( x, y, \) and \( z \) coordinates. Since we consider standard hidden Markov models (HMMs) for classification, we had to preprocess the continuous data provided by the considered sensor. Our preprocessing consisted in estimating tendencies: we obtain the difference in the positions obtained in consecutive frames and codify them into two values: +1 when the difference is positive and a 0 when the difference is zero or negative. Thus, the observations are sextuples of zeros and ones (the number of different observations is \( 2^6 \)). These are the inputs for the HMMs.

3.2 Gesture recognition models

As classification model we consider an HMM, one of the most popular models for gesture recognition [Aviles et al., 2011; Aggarwal and Ryoo, 2011; Mitra, 2007]. For each gesture \( i \) to be recognized we trained an HMM, let \( M_i \) denote the
HMM for the \( i^{th} \) gesture, where \( i = \{1, \ldots, K\} \) when considering \( K \) different gestures. The models are trained with the Baum-Welch algorithm using complete sequences depicting (only) the gestures of interest. Each HMM was trained for a maximum of 200 iterations and a tolerance of 0.00001 (the training process stops when changes between probabilities of successive transition/emission matrices do not exceed this value); the number of states in the HMM was fixed to 3, after some preliminary experimentation.

For making predictions we evaluate the different HMMs over the test sequence using the Forward algorithm, see [Rabiner, 1990] for details. We use the probabilities returned by each HMM as its confidence on the gesture class for a particular window.

3.3 Simultaneous segmentation and recognition

The multi-windows approach to gesture segmentation and recognition is as follows, see Figure 1. For processing a continuous video we trigger windows incrementally: at time \( t_0 \) a temporal window \( W_0 \) of length \( \Delta \) is triggered and all of the (trained) HMMs are evaluated in this window. At time \( t_1 \) we trigger another window \( W_1 \) of length \( \Delta \) and increase window \( W_0 \) by \( \Delta \) frames, the HMMs are evaluated in these two windows too. This process is repeated until certain condition is met (see below) or until window \( W_j \) surpass a maximum length, which corresponds to the maximum number of allowed simultaneous window, \( q \).

In this way, at a time \( t_g \), we have \( q \) windows of varying lengths, and the outputs of the \( K \)–HMMs for each window (i.e., a total of \( g \times K \) probabilities, where \( K \) is the number of gestures or activities that the system can recognize). The outputs of the HMMs are given in the form of probabilities. To obtain a prediction for each window \( i \) we simply keep the label/gesture corresponding to the model that obtains the highest probability in window \( i \), that is, \( \arg \max_k P(M_k, W_i) \).

In order to detect the presence of a gesture in the continuous video we estimate at each time \( t_j \) the percentage of votes that each of the \( K \)–gestures obtains, by considering the predictions for the \( j \)–windows. If the number of votes exceeds a threshold, \( \tau \), we trigger a flag indicating that a gesture has been recognized. When the flag is on, we keep increasing and generating windows and storing predictions until there is a decrement in the percentages of votes for the dominant gesture. That is, end of the gesture happens in the frame where there is a decrement in the number of votes. Alternatively, we also experimented with varying the window in which we segment the gesture: we segmented the gesture 10 frames before and 10 frames after we detect the decrement in the percentage of votes, we report experimental results under the three settings in Section 4. At this instant the votes for each type of gesture are counted, and the gesture with the maximum number of votes is selected as the recognized gesture. Once a gesture is recognized, the system is reset; that is, all ongoing windows are discarded and the process starts again with a single window.

One should note that the less windows we consider for taking a decision the higher the chances that we make a mistake. Therefore, we ban the proposed technique for making predictions before having analyzed at least \( p \)–windows. Under these settings, our proposal will try to segment and recognize gestures only when the number of windows/predictions is between \( (p, q) \).

Figure 2 illustrates the process for simultaneous segmentation and recognition for a particular test sequence containing one gesture. The first three plots show the probabilities returned by the HMMs for three gestures; we show the probabilities for windows starting at different frames of the continuous sequence. The fourth plot shows the percentage of votes for a particular gesture at different segments of the video. For this particular example, the proposed approach is able to segment correctly the gesture (the boundaries for the gesture present in the sequence are shown in gray). In the next section we report experimental results obtained with our method for simultaneous segmentation and recognition of gestures.

4 Experimental results

We performed experiments with the multi-windows approach by trying to recognize gestures to command a robot. Specifically, we consider three gestures: move-right (MR), attention (ATTN), move-left (ML), these are illustrated in Figure 3. For evaluation we generated sequences of gestures of varying lengths and applied our method. The number of training and testing gestures are shown in Table 1. Training gestures were manually segmented. Test sequences are not segmented; they contain a single gesture, but the gesture is surrounded by large portions of continuous video without a gesture, see Figure 2.

Three different subjects recorded the training videos. The test sequences were recorded by six subjects (three of which were different from those that recorded the training ones). The skeleton information was recorded with the NUI Capture software\(^3\) at a rate of 30fps. The average duration of training gestures was of 35.33 frames, whereas the average duration of test sequences was of 94 frames (maximum and minimum durations were of 189 and 55 frames respectively).

All of the parameters of our model were fixed after preliminary experimentation. The better values we found for them are as follows: \( \Delta = 10 \); \( p = 30 \); \( q = 60 \); \( \tau = 100 \). After training the HMMs individually, we applied the multi-windows approach to each of the test sequences.

We evaluate the segmentation and recognition performance as follows. We say the proposed method correctly segments

\(^3\)http://nuicapture.com/
Figure 2: Multi-windows technique in action. The first three plots show probabilities obtained per each HMM for windows starting at different times. In the bottom-right plot we show the number of votes obtained by the dominant HMM, note that the number of votes start to diminish, this is taken as an indication of the end of the gesture (best viewed in color).

Table 1: Characteristics of the data set considered for experimentation. We show the number of training videos per gesture, and, in row two, the number of gestures present in the test sequences.

<table>
<thead>
<tr>
<th>Feature</th>
<th>MR</th>
<th>ATTN</th>
<th>ML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training vids.</td>
<td>30</td>
<td>30</td>
<td>30</td>
</tr>
<tr>
<td>Testing vids.</td>
<td>18</td>
<td>18</td>
<td>21</td>
</tr>
</tbody>
</table>

Table 2: Segmentation (Seg.) and recognition (Rec.) performance of the multi-windows technique.

<table>
<thead>
<tr>
<th></th>
<th>Before</th>
<th>In</th>
<th>After</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>29.82%</td>
<td>82.35%</td>
<td>26.32%</td>
</tr>
<tr>
<td>10</td>
<td>54.39%</td>
<td>67.74%</td>
<td>63.16%</td>
</tr>
<tr>
<td>15</td>
<td>59.65%</td>
<td>64.71%</td>
<td>70.18%</td>
</tr>
<tr>
<td>20</td>
<td>78.95%</td>
<td>62.22%</td>
<td>80.70%</td>
</tr>
</tbody>
</table>

Despite the fact that segmentation performance may seem low, one should note that for the considered application it is not too bad for an user to repeat a gesture 3 times in order that a robot correctly identifies the command we want to transmit. Instead, accurate recognition systems are required so that the robot clearly understand the ordered command, even when the user has to repeat the gesture a couple of times.
frames, we were able to segment up to 80% of the gestures. Recognition rates decreased accordingly. When we compare the segmentation performance obtained when segmenting the gesture before, in or after the decrement of votes, we found that the performance was very similar. Although, segmenting the gesture 10 frames before we detected the decrement seems to be a better option. This makes sense, as we would expect to see a decrement of votes when the gesture already has finished.

Regarding efficiency, in preliminary experiments we have found the proposed method can run in near real-time. In a state-of-the-art workstation, it can process data at a rate of 30fps, which is enough for many human-computer interaction tasks. Nevertheless, we still have to perform a comprehensive evaluation of our proposal in terms of efficiency and taking into account that in some scenarios a high-performance computers are not available.

From the experimental study presented in this section we can conclude that the proposed method is a promising solution to the problem of simultaneous gesture segmentation and recognition. The simplicity of implementation and the efficiency of our approach are beneficial for the development of ubiquitous and human-centric systems.

### 5 Conclusions and future work directions

We proposed a new method for simultaneous segmentation and recognition of gestures in continuous video. The proposed approach combines the outputs of classification models evaluated in multiple temporal windows. These windows are dynamically and incrementally created as the video is scanned. We report preliminary results obtained with the proposed technique for segmenting and recognizing gestures to command a robot. Experimental results reveal that the recognition performance of our method is very close to that obtained when using manually segmented gestures. Segmentation performance of our proposal is still low, yet current performance is acceptable for the considered application. The following conclusions can be drawn so far:

- The proposed method is capable of segmenting gestures (with an error of 5 frames) at low-mild recognition rates. Nevertheless, these rates are accurate-enough for some applications. Recall we are analyzing a continuous sequence of video and that we do not require of a model for no-gesture, as required in related models.
- Recognition rates achieved by the method are acceptable for a number of applications and domains. In fact, recognition results were very close to what we would obtain when classifying manually-segmented gestures.
- The proposed method is very easy to implement and can work in near real-time, hence its applicability in ubiquitous data mining and human-centric applications are quite possible.

The proposed method can be improved in several ways, but it remains to be compared to alternative techniques. In this aspect we have already implemented the method from [Kim et al., 2007], but results are too bad in comparison with our proposal. We are looking for alternative methods to compare our proposal.

Current and future work includes extending the number of gestures considered in this study and implementing the method in the robot of our laboratory. Additionally, we are working in different ways to improve the segmentation performance of our method, including using different voting schemes to combine the outputs of the different windows.

### References


Cicerone: Design of a Real-Time Area Knowledge-Enhanced Venue Recommender

Daniel Villatoro, Jordi Aranda, Marc Planagumà, Rafael Gimenez and Marc Torrent-Moreno
Barcelona Digital Technology Centre, Barcelona, Spain
{dvillatoro,jaranda,mplanagumà,rgimenez,mtorrent}@bdigital.org

Abstract

Smart-devices with information sharing capabilities anytime and anywhere have opened a wide range of ubiquitous applications. Within urban environments citizens have a plethora of locations to choose from, and in the advent of the smart-cities paradigm, this is the scope of location-based recommender systems to provide citizens with the adequate suggestions. In this work we present the design of an in-situ location-based recommender system, where the venue recommendations are built upon the users’ location at request-time, but also incorporating the social dimension and the expertise of the neighboring users knowledge used to build the recommendations. Moreover, we propose a specific easy-to-deploy architecture, that bases its functioning in the participatory social media platforms such as Twitter or Foursquare. Our system constructs its knowledge base from the accessible data in Foursquare, and similarly obtains ratings from geopositioned tweets.

1 Introduction

Urban environments host a plethora of interesting locations such as restaurants, shops, museums, theaters and a wide range of other venues that neither can be known by all users nor they might be interested in visiting all. However, each citizen can potentially become an expert of the neighborhood he visits more often or lives, as he will know, and maybe have visited, more venues in such area. Therefore, it is straightforward to see how for an specific citizen might not be a problem to find an adequate venue for his taste on his neighborhood of expertise, but it potentially becomes cumbersome to do the same task when in a different less-known neighborhood. It becomes then a problem for citizens to find locations they might enjoy when away of their area of expertise. The problem of finding adequate items for specific users is that classically solved by recommender systems.

In our case, we focus on location-based recommender systems [Zheng et al., 2009; Park et al., 2007], where users are recommended locations to visit expecting to maximize users’ satisfaction. These type of recommenders complement the previously analyzed ones as they also have to take into account the context, distance from the user to the recommended venue, and maybe several other factors.

With the penetration of smart-devices, users have the possibility to access information anytime anywhere, and the system we present in this work profits from those ubiquitous computing capabilities; our on-site location-based recommender system allows users to obtain the most adequate venue with respect to their current position. Our approach profits from a different dimension of the users’ parameters space, namely their social relationships and their relative geographical knowledge with respect to the location of the items. This model provides an alternative solution to the problem of providing personalized recommendations in a geospatial domain: user expertise in this type of domain conveys an implicit continuum knowledge of the surrounding geospatial area and the locations within that area. Our solution intelligently combines this user geospatial knowledge to the classical social distances amongst users used in state-of-the-art recommenders.

Our system personalizes recommendations of locations not only considering the past history of a specific user, but also (1) the current location of the user, (2) the social distance with other similar users and (3) their expertise in the area where the recommendation is going to be provided. This aggregation function basically expresses a tendency of a user to visit a certain location given its distance to the location, and the past history of the user and its friends and their knowledge of the area.

To the best of our knowledge, there are no existing recommender systems that profit from the inherent characteristics of the geographical location, such as continuity in space, user’s area expertise or word-of-mouth location suggestions, to generate recommendations to users.

2 State of the Art

As we have discussed previously, the problem of finding adequate venues for citizens to visit is a problem already tackled by the recommender community, under the location-based recommender systems [Zheng et al., 2009; Park et al., 2007]. Despite the impressive amount of literature in such area, this is still an open problem, even for those with access to complete datasets and user profiles [Sklar et al., 2012], as new methods and algorithms are being proposed to boost their ef-
ficiency.
In this work however, we propose the integration of social information into the calculation of the recommendations. Some authors have investigated the potential of the explicit inclusion of information of user’s relationships from social networks to generate the neighborhood used in classical collaborative filtering (CF) algorithms (social filtering), improving the results obtained by the classic CF in the analyzed scenarios [Groh and Ehmig, 2007].

Others [Bonhard and Sasse, 2006] have analyzed how the relationship between advice-seeker and recommender is extremely important in the user-centered recommendations, concluding that familiarity and similarity amongst the different roles in the recommendation process aid judgement and decision making. As well as in our approach, some researchers have considered the important role of experts [Amatriain et al., 2009; Bao et al., 2012], however in our case, these experts are calculated automatically for each specific area of the city and weighted with respect to the social distance amongst the advice-seeker and recommender.

A similar recommendation approach is presented [Ye et al., 2010], where authors also propose the usage of Foursquare information to provide venue recommendations to users; more importantly the social perspective is integrated into their recommendations, developing a Friend-Based Collaborative Filtering (where the neighbours for CF are selected from the social network of users), and an extension of this method Geo-measured Friend-Based Collaborative Filtering (where only closely located friends are selected as neighbours for CF).

Our method then proposes a combination of the Geo-measured Friend-Based Collaborative Filtering [Ye et al., 2010] and experts [Amatriain et al., 2009; Bao et al., 2012], in our specific case, neighborhood or area experts.

3 Cicerone Recommender System

In this section we provide the theoretical framework of the Cicerone location-based recommender system. Firstly we describe the basic terminology used later in the recommendation algorithm. As we have sketched previously, our system bases its functioning in three information elements: the users’ social network, the users’ area knowledge and the current location of the requesting user.

3.1 Basic Terminology

As used herein, the term “location data item” stands for any location item or representation of a location. A “location item” is intended to encompass any type of location which can be represented in a map using a latitude, a longitude, and possibly a category.

The location recommender may be capable of selecting relevant locations for a given target user. To do so, users should be comparable entities and locations as well. It should be understood that the implementations described herein are not item-specific and may operate with any other type of item visited/shared by a community of users. For the specific case of bars or restaurants items, users may interact with the items by visiting them. The Recommendations Set is the locations set formed by the items the user is being recommended. A User A’s Recommendations set will be denoted herein as $R^t_A$.

An essential concept is the one of “Check-in” ($CI^t_{U,L}$) which represents the attendance\(^1\) of a user $U$ to a certain location $L$ in the last $t$ days, and therefore. Our system will generate Location recommendations to the users by considering not only its geographical position, but also its social relationship with other users and their degree of knowledge of the visited locations.

In order to obtain more adequate recommendations in this type of environments, we envision the necessity of certain estimators. Firstly, we need to quantify how well a certain user knows a specific area (by considering the attendance frequency to locations in such area with respect to the rest of the city). Moreover, it becomes necessary to understand the social distance between the target user and the other users, whose opinions are being used to create the recommendations for the target user.

These measures are clearly described and specified next:

The Area Knowledge ($AK^t_{U,L}$) of a user $U$ with respect to a location $L$ is calculated:

$$AK^t_{U,L} = \frac{\sum_{l \in PostalCodes} CI^t_{U,l}}{\sum_{a \in Locations} CI^t_{U,a}} \tag{1}$$

and represents how familiar a user is within an specific area of the city (represented by its postal code).

The Location Frequency ($LF^t_{U,L}$) of a user $U$ in a certain location $L$ is calculated:

$$LF^t_{U,L} = \frac{CI^t_{U,L}}{\sum_{a \in Locations} CI^t_{U,a}} \tag{2}$$

and normalizes the number of visits of the user $U$ to the location $L$.

The Social Importance ($SI^t_{U,U'}$) of a user $U'$ for a user $U$ is calculated:

$$SI^t_{U,U'} = \frac{(Degree_{U'})^{\frac{1}{2}}}{(nodes - 1)} \tag{3}$$

where $Degree_U$ represents the number of connections that $U$ has in its social network, $nodes$ represents the total number of nodes in the social network (and used to normalize the $SI$), and $d(U,U')$ represents the geodesic distance, i.e. minimum number of hops necessary to reach $U'$ from $U$ using the shortest path in their social network\(^2\).

The Location Value ($LV^t_{L,U}$) of a location $L$ for a user $U$ at time $t$ is calculated:

$$LV^t_{L,U} = \frac{\sum_{users \in L} (LF^t_{U,L} \times AK^t_{U,L} \times SI^t_{U,U'})}{|users|} \tag{4}$$

\(^1\)The attendance of a user to a certain location can be captured in several ways, for example, a Foursquare Check-in, a geopositioned tweet, or a CDR trace of a phone call.

\(^2\) $d(U,U') < 0$ means that there is no possible path that connects $U$ and $U'$. 

35
where \(|users|\) represents the number of users that have "checked-in" to that Location.

The resulting value basically aggregates the information of surrounding detected locations considering the social distance of our specific user to the users that visited that location (social information), and their familiarity in the area of the location (geographical information).

### 3.2 Cicerone Recommendation Algorithm

The general algorithm for the functioning is the following:

1. Once the position of a user A is detected, the system automatically captures its Latitude and Longitude and launches the process that builds the personalized recommendation set for that position and user at that certain moment.

2. The system retrieves all the locations in 100m radius of the current position.

3. The recommendation set for user A, \(R_\alpha\), is a set constructed with all the locations in 100m radius of the user current position.

4. The system calculates the location value of each of the locations in that set.

5. The system orders the retrieved locations according to the calculated Location Values and constructs the Recommendation Set with the 3 with a highest value.

### 4 Functional implementation of Cicerone

As explained previously, the theoretical framework to build the recommendation needs from a number of data sources, namely, users locations, venues and the social relationships amongst users. As this recommendation process is envisioned to be executed when users are in-situ, the main functional requirement for our system is to work from a mobile device. Working prototypes have decided to opt for the development of a dedicated app (such as Yelp, TimeOut or TripAdvisor), that users have to download within their devices. The app provides several advantages as the explicit user profiling as well as the definition of the necessary information to obtain the recommendation. However, for us it implies the big problem of reaching a critical mass of users that would make the knowledge base and the recommendation more accurate. To avoid this limitation we have opted to develop our system as a service embedded within already massive social networks. Twitter seems to be the ideal candidate for us for the following reasons:

- Twitter shows a widespread uniform penetration almost worldwide, with an continuously increasing number of users (288 million monthly active users in July 2012, showing an increase of 40% since July 2009 [GlobalWebIndex, 2013]).
- It allows users to associate their location when posting a message, and associate the specific coordinates as meta-information.
- It provides developers with an accessible API to obtain in near real-time the publicly published tweets.
- Twitter captures a social network of followers and followings, publicly available for each user.

As we have initially decided to deploy our application in the city of Barcelona, Twitter confirms to be an ideal candidate. The number of captured geopositioned tweets daily within Barcelona is 6200 (from data coming from 2012), and the social network inferred from the users posting them can be seen in Figure 1 (with an average degree of 2.93).

Similarly, and to populate our items database, we opt to use the crowdsourced database of Foursquare. Foursquare is a location-based social media platform to communicate the venues a user is in. This platform allows users to input into their databases new locations, by introducing not only the venue’s name and specific location (with the GPS position and postal address) but also a semantic category. Foursquare describes the places according to a rather complete taxonomy, where about 400 kinds of places are identified and grouped in 9 wide categories. Foursquare provides an accessible API allowing us to take snapshots of the existing locations in a certain city. Within the city of Barcelona, from a snapshot taken April 2013, we have detected over 66,000 foursquare locations, uniformly distributed amongst the different districts.

Moreover, within the OpenData movement, the city of Barcelona provides a machine-readable administrative division necessary for our theoretical calculations (namely the District divisions).
5.1 Social Networks Monitoring: Sensing the City

The usage of social media platforms in our system are twofold: (1) information acquisition to feed the knowledge base of our platform, and (2) a channel for users’ interaction with our technology.

The participatory information provided by users in Foursquare will be used to populate our items database; similarly, we will use geopositioned tweets to calculate users’ Area Knowledge. Therefore, the social network monitor is the first layer of our architecture and it is composed by two crawlers: a Foursquare crawler and a Twitter crawler. The Foursquare crawler is in charge to scan the target city for new venues. Once a new venue is identified, it is stored in the items database with its associated metainformation such as its specific coordinates, the address or the category. The Twitter crawler is in charge to capture all the tweets generated in the target city. Its scope is threefold: (1) build and update the users’ social network, (2) update the user area knowledge using its geopositioned tweets, and (3) permits users’ communication with the system.

Moreover, and given its popularity, we use Twitter as the communication channel of our recommender system through a bot account managed by our intelligent agent.

5.2 Persistence Infrastructure: Urban data Model

Any recommender system bases its functioning in three main elements: users, items and ratings. These three elements have to be stored according to the inherent properties of the system, which in this case, imply real-time information access and update. Fed by the crawlers, the data required for our recommendation solution arrives to the persistence manager and each of these elements are stored in a persistence infrastructure in the following way: Users: One of the main functional requirements of the recommendation algorithm is the access to the social network of users. In order to effectively store this information, we opt for using a graph-oriented database, namely Neo4j. These type of databases allow us to persist users’ social network in the form of a directed weighted graph. In this database, we persist users as nodes and then establish edges amongst nodes if there exist a social relation amongst them. Consequently, an edge between two nodes is created if there exists a social relation amongst them, according to users’ Twitter profiles; specifically, an edge is created amongst from user $A$ towards user $B$ if user $A$ follows user $B$ in Twitter. At this edge level, the edge’s weight will be defined depending on the users interactions: different types of Twitter interactions (such as mentions, retweets or favourited) will affect the weight differently. Another important information about users is saved, namely his “Check-ins” (as described in Sec. ). These “check-ins” (the specific coordinates of each user geopositioned tweet) is stored into a MongoDB database using its location within space, that is proffited from when using geo-spatial indexing. Given these two characteristics (rapid information access and geo-spatial indexing), as well as the potential for distributed computing, we opt to implement this database using MongoDB.

Ratings: The notion of rating is classically treated as an explicit evaluation of users about an item. However, in this work, we take an alternative approach for ratings: we consider as a constant rating value the users presence in a location, sensed through the geopositioned tweets posted from or close to the venue location. This value is not obtained directly, the Ratings will be part of knowledge obtained by the recommendation engine and their information update process capturing user’s visits to specific locations but this will be explained on the Section 5.3.

5.3 Recommendation Engine: Information Update Process

The last component in our proposed architecture is the recommendation engine containing the implementation of the theoretical algorithms previously explained in the Section 3. Once we have our social networks monitor as an urban data sensor, and the ability to persist all the raw data required by the system, this component will be the responsible of the knowledge extraction process and the bussines logic triggered
to generate a recommendation.

Because of the real-time aspect of our system, our recommendation platform (whose workflow is detailed in Figure 3) needs to continuously update some information elements such as users’ Area Knowledge and Location Frequency, the creation or update of social relationships amongst users or the appearance of new locations. Specifically, we envision the users’ communication with our system through a Twitter personality that encapsulates our recommendation platform; everytime a user mentions our system’s username, the platform will capture this tweet (through the Mention’s Service sketched in Figure 2) and identify it as an explicit request for a recommendation that will trigger the whole intelligent process. Eventhough our technological platform allows us to generate recommendations everytime a user’s location is captured (with every geopositioned tweet), we rather restrict its functioning with a mention system reducing the overall intrusiveness.

After the recommendation is generated, it is returned to the user also through Twitter with a message posted by our intelligent agent.

6 Conclusion and Future Work

The designed recommender system plans to profit from the information proactively shared by users in the analyzed participatory platforms. However, as recently argued in Jeske, 2013, these type of crowdsourced systems is sensible to malicious attacks: in our case, and given the lack of restrictions to post geo-positioned content from Twitter, someone could easily envision the method to create a fake user to become the one with higher area knowledge in every area of the city, and then influence directly the resulting recommendations to his own will.

Despite this potential problem associated to the publishing policy of Twitter and Foursquare, and as we have analyzed in Sec. 2, many others have used information from these sources to generate location-based recommendations. However, and to the best of our knowledge, the presented algorithm is the first to include explicitly the user’s expertise about one of the fundamental properties of the items: the area where it is located. By combining this information, with some social information, we hypothesize that our system will be able to outperform other location-based recommender systems.

Our main long term research task to be performed is the development of a user profiling in term of the type of venues the user attends to, with the overall objective of combining the area expertise and with specific user profiles.

Acknowledgments

This work has been completed with the support of ACC1Ó, the Catalan Agency to promote applied research and innovation.

References


Road-quality classification and bump detection with bicycle-mounted smartphones

Marius Hoffmann, Michael Mock, Michael May
Fraunhofer IAIS
Schloss Birlinghoven, 53754 St Augustin, Germany
first.lastname@iais.fraunhofer.de

Abstract
The paper proposes an embedded surface road classifier for smartphones used to track and classify routes on bikes. The main idea is to provide, along with the route tracking, information about surface quality of the cycling route (is the surface smooth, rough or bumpy?). The main problem is the quantity of accelerometer data that would have to be uploaded along with GPS track, if the analysis was done off-line. Instead, we propose to classify road surfaces online with an embedded classifier, that has been trained off-line. More specifically, we rely on the accelerometer of a bicycle-mounted smartphone for online classification. We carry out experiments to collect cycling tracks consisting of GPS and accelerometer data, label the data and learn a model for classification, which again is deployed on the smartphone. We report on our experiences with classification accuracy on and runtime performance of the classifier on the smartphone.

1 Introduction
The main motivation of this work is to provide a community based cycling route road quality classification service. There are many community based services providing cycling routes together with altitude profiles, but none of them is providing information about the road quality of the route, i.e. whether the road is smooth, rough, or bumpy. Many bicycling community web-portals like [http://www.bikemap.net] offer facilities for uploading and downloading GPS tracks for cycling routes. To our knowledge, none of them provides information about road surface quality of the cycling route. Route quality information could be gathered together with GPS track using the accelerometer data coming from bicycle mounted smartphone. Obviously, including all accelerometer raw data in the data upload would increase data traffic significantly and may not be tolerable for the user, especially when gathering long tracks. The solution is to implement a road surface classification algorithm on the smartphone and to upload the classification results together with the GPS track. Similar approaches already have been successfully applied for other vehicles than bicycles. Pothole detection using GPS data and accelerometer data with dedicated hardware devices mounted in Taxi cabs has already been successfully explored in [Ericsson et al., 2008], [Strazdins12 et al., 2011] and [Mednis et al., 2012] investigate road condition monitoring for vehicular sensor networks based on time series analysis. We investigate experimentally, whether we can achieve a road surface classification using smartphones mounted on bicycles. In order to cope with the restricted computational power of these devices, we apply a machine learning approach: we learn a classifier off-line on a standard PC and apply the classifier online on the smartphone.

We collected GPS tracks and acceleration data (based on the mobile phone’s accelerometer sensor) and applied two different approaches for classification of road surface quality, both based on standard machine learning classifiers: in a direct segmentation classification approach, we used manual labeling of road segments of fixed length (smooth, rough, bumpy) to train a classifier, based of various parameter settings for feature extraction. The best result that we obtained in a cross-validation was a 20% increase of accuracy against a standard Kappa-Statistics. In a second approach, we trained a classifier for detecting bumps. Here we achieved an accuracy of 97%. Using this bump detector, we performed a threshold-based road segment classification, which delivered much more comprehensible results. A closer look at input data, manual labeling, classification results, and comparison with the real-world, revealed that the manual labeling was error prone. We conclude that the simple bump-detector based classification approach can be used for road surface quality classification and even does not require further manual labeling of road segments.

2 Classification approach and Results
Most of today’s smartphones are equipped with GPS and accelerometer sensors. In order to ensure that our algorithm performs not only on today’s top range models, we carried out the experiments with a 2-years old Nokia 5800, one of the first mass models providing accelerometer data. Figure 1 illustrates a track of accelerometer data collected with a smartphone.

Figure 1 shows the length of the accelerometer vector plotted over a track. We see that the data provides a more or less continuous signal (at 37 Hz in our case) over the complete track. As we want to explore a machine learning classification approach for road surface classification, we first have to
define the features which are used to train the classifier. The
raw data consists of GPS positions and their time stamps, and
acceleration values only. Acceleration values are represented
by a three-dimensional vector. In a first step, we extract as
many features from the data as possible and evaluate experi-
mentally, which feature selection yields the best classification
result.

In our first approach to classification described in section
2.1, we divide the road into segments of varying length. In our
second approach described in section 2.2, we just consider
two subsequent GPS points as boundary of a segment. In
both cases, we get a segmentation of the cycling route in a
sequence of segments as shown in Figure 2. As a result, the
recorded acceleration data is associated to a certain segment.

![Figure 2: This figure shows how a track will be segmented
into a set of segments](image)

The segmentation shown in 2 allows to indicate, which po-

tion of the road has a certain surface property or would even
contain potholes. We can now analyze segment by segment.
At the end each segment contains GPS and acceleration data
which can be used for creating features for this segment.

Features which can be extracted from the GPS data are speed
and inclination. To simplify the handling of the accel-
eration data provided by the accelerometer, which is made up
of an 3D vector, we will further use L2-norm of this vector
which is defined as \( ||x|| = \sqrt{x_1^2 + \ldots + x_n^2} \). The example
shown in Figure 1 already illustrates, that changes in these
values and sometimes even potholes can be detected by (hu-
man) visual inspection of the data. For our machine learning
approach, we extract the mean, the variance and the standard
deviation of the acceleration values of a segment as features
for this segment.

2.1 Direct road surface classification
In this section, we apply standard classification methods to
segments of varying length, based on the features described
above. For our analysis we consider a number of previ-
ous segments which are before the segment that we want
to classify. We define a whole road as a set of segments
\( S = \{ s_1, s_2, \ldots, s_n \} \). We consider the previous \( x \) segments
\( s_{i-x}, s_{i-(x-1)}, \ldots, s_i \) of the segment \( i \) which we want
to classify as features for \( s_i \). In this case the features of the
previous segments serve us (primarily our machine learning
algorithm) as additional information for our analysis. How
much these feature information’s are relevant and how many
segments we must consider has be analyzed experimentally.

The organization of the training data is shown in Table 1.
Now we use all extracted features of such a set of segments
as training data. Every segment has its own row with its own
features and additional features of previous segments. Each
row in this table also contains the class as entry in the column
called label. This column contains class which later on will
be learned by the machine learning algorithm. For example
row 1 has the label smooth as class for segment \( S_1 \).

<table>
<thead>
<tr>
<th>( f_{S_{i-2}} )</th>
<th>( f_{S_{i-1}} )</th>
<th>( f_{S_i} )</th>
<th>label ( (S_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>( f_{S_0} )</td>
<td>( f_{S_1} )</td>
<td>smooth</td>
</tr>
<tr>
<td>( f_{S_0} )</td>
<td>( f_{S_1} )</td>
<td>( f_{S_2} )</td>
<td>smooth</td>
</tr>
<tr>
<td>( f_{S_1} )</td>
<td>( f_{S_2} )</td>
<td>( f_{S_3} )</td>
<td>smooth</td>
</tr>
<tr>
<td>( f_{S_2} )</td>
<td>( f_{S_3} )</td>
<td>( f_{S_4} )</td>
<td>rough</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
<td>\vdots</td>
</tr>
</tbody>
</table>

Table 1: This table illustrates how the features of each seg-
ment are arranged in order to generate a training set of data

We want to evaluate how well the classifiers can learn from
the provided data and which features and parameters influ-
ence the performance of these classifiers. The goal is to eval-
uate whether it is possible at all to learn from the data and
if so, which are the best parameters (for example segment
length, number of segments to be included in the table).

As raw data we recorded one route several times. The route
for direct surface classification was recorded 16 times and
leads through urban terrain mostly the city of Bonn and they
have a length of approximate 13-14km per track (the devia-
tion in length results from the GPS inaccuracy). Each track
was labeled for classification by hand with the tool presented
in [Guc et al., 2008].

The previously mentioned segment arrangement
\( s_{i-x}, s_{i-(x-1)}, \ldots, s_i \) will further be called \( S_{\text{line}} \), which
only consist of previous segments and where \( i \) is our current
position. For the segments length, the other Fixed segment
length is fixed from the beginning (during the evaluation
fixed values of 1m, 2m, 5m, 10m, 15m and 20m are used).
For the fixed length parameter the amount of acceleration

40
values can vary, because the amount of values is speed dependant. For classification we will use two different Algorithms the K-Nearest-Neighbor and the Naive Bayes Classifier. Five different features were extracted from the training data: speed, inclination, acceleration mean, acceleration variance, acceleration standard deviation.

The following table shows a compact overview of all parameters which were evaluated.

<table>
<thead>
<tr>
<th>Parameter type</th>
<th>Parameter Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML algorithm</td>
<td>K-NN, Naive Bayes</td>
</tr>
<tr>
<td>segments length</td>
<td>variable length: gps</td>
</tr>
<tr>
<td>number of segments</td>
<td>fixed length: 1m, 2m, 5m, 10m, 15m, 20m</td>
</tr>
<tr>
<td>extracted features</td>
<td>inclination, speed, acceleration (mean, variance, std)</td>
</tr>
</tbody>
</table>

Table 2: This table gives an overview of all parameters which were changed during evaluation. The acceleration contains three features, acceleration-mean, -variance and -standard deviation.

To measure the performance of the classification algorithm on the evaluation data, a 10-folded cross-validation was included. A N-folded cross validation splits the test data into N equally large sets and then uses N − 1 set for training to classifier and 1 set for validating the learned concept this is repeated N times where for every iteration a different set of the N sets is used for validation. At the end a confusion matrix is provided from the cross-validation module which consists of the average performance values of the classification.

Additionally we performed a feature selection optimization in order to find the best feature combination. This optimization allows to find a feature combination which only contains features which influence the learning algorithm positively and result in high accuracy. Features which confuse the learning scheme will not be selected anymore. We found that the previously mentioned speed and inclination feature confuses the learning scheme and results in performances which are worse than the corresponding kappa statistics.

<table>
<thead>
<tr>
<th>true smooth</th>
<th>true bumpy</th>
<th>true rough</th>
<th>class precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>pred. smooth</td>
<td>5785</td>
<td>882</td>
<td>92</td>
</tr>
<tr>
<td>pred. bumpy</td>
<td>836</td>
<td>1052</td>
<td>62</td>
</tr>
<tr>
<td>pred. rough</td>
<td>151</td>
<td>110</td>
<td>492</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>class recall</th>
<th>accuracy: 77.45%</th>
</tr>
</thead>
<tbody>
<tr>
<td>85.425%</td>
<td>51.468%</td>
</tr>
</tbody>
</table>

Table 3

The classification performance for the Naive Bayes and the K-NN were almost similar, but the K-NN performed (on average) slightly better than the Naive Bayes.

For K-NN algorithm, the performance increases with an increasing number of the segments which are considered for classification. The Naive Bayes classifier, however, has a more constant performance, independently of the number of segments included in the table. The evaluation also showed that the classification results which use longer segments lengths (15m and 20m) perform much better than the ones with short segment length’s (2m). When looking at the influences of all features, we observed that the speed feature does not contribute to the classification. The inclination feature, even worse, confuses the classifier.

The best results (table 3) are achieved with the features acceleration (mean, variance, standard deviation) and a segment length of 20m and 13 segments must be considered for classification. The used segment setup is the Stine setup. The corresponding kappa statistic achieves an accuracy of 56.357% which makes a difference of 21.101% between the classifier and its kappa statistic.

The overall results of the classification (at best 78%) are not very satisfying for a classification model. We will see in section 2.2 that the bump detection just based on GPS-defined segments performs much better.

2.2 Bump detection based classification

In this approach, we first consider the detection of single bumps or potholes. The classifier in this first just distinguishes the two classes: “bump” and “no bump”. For the bump classification a different route was selected and recorded 15 times. Each of them has a length between 110m and 130m per track (here the deviation in length also results from the GPS inaccuracy). Again each track was labeled for classification by hand via the already mentioned annotator tool.

The performance of the bump classification works out much better compared to the highest accuracy of the surface classification. Again, the feature “speed” turned out to be irrelevant and the feature “inclination” was confusing the classifier. It was also observed that (for surface- not bump-classification), the more segments are considered the more the accuracy declines. The reason for this is that the longer the considered area the more unimportant information is contained in the data which should be classified. In comparison to the surface classification, the bump classification needs shorter segment length’s (1m to 5m) to reach high classification accuracy. The longer the segment lengths, the worse the classification performance gets. The long segment also confuses the classification algorithm, this was verified by comparing the results of the classification with the corresponding kappa statistic. The best result were achieved with the segment length GPS parameter. This is quite expected, because “bumps” are short term events and GPS-based segmentation (i.e. every two succeeding GPS points define a segment) is the smallest achievable spatial granularity.

<table>
<thead>
<tr>
<th>true no bump</th>
<th>true bump</th>
<th>class precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>pred. no bump</td>
<td>404</td>
<td>404</td>
</tr>
<tr>
<td>pred. bump</td>
<td>2</td>
<td>29</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>class recall</th>
<th>accuracy: 98.186%</th>
</tr>
</thead>
<tbody>
<tr>
<td>99.507%</td>
<td>82.857%</td>
</tr>
</tbody>
</table>

Table 4

As we can see it is indeed possible to do pothole and bumpy detection with a very high accuracy, just using the Naive Bayes Classifier on a single segment. This led us to extend this simple approach to be applicable in road surface
classification, with the three classes "smooth", "rough", and "bumpy", as described in the following.

**Extended bump classification** The bump detection can be altered slightly to derive another concept for surface classification. The main idea is to count the number of bumpy segments in a certain road section. Depending on that number, one of the classes "smooth", "rough", and "bumpy" is assigned as follows:

- For \( \frac{0}{N} \leq \|bumps\| \leq \frac{N}{3} \), the class smooth is assigned.
- For \( \frac{N}{3} < \|bumps\| \leq \frac{2N}{3} \), the class rough is assigned
- For \( \frac{2N}{3} < \|bumps\| \leq N \), the class bumpy is assigned

Not surprisingly, the best results were achieved for \( N=3 \), i.e. just considering the GPS-Segments \( S_{i-1}, S_i \), and \( S_{i+1} \) for the classification of GPS-segment \( S_i \). In other words, a GPS segment is considered as, for example, smooth, if at most one of its preceding, the GPS-segment itself, and the succeeding GPS-segment have a bump. The results are shown in Table 5.

<table>
<thead>
<tr>
<th>pred. smooth</th>
<th>true smooth</th>
<th>true bumpy</th>
<th>true rough</th>
<th>class precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>27865</td>
<td>2249</td>
<td>1823</td>
<td>3315</td>
<td>89.579%</td>
</tr>
<tr>
<td>pred. bumpy</td>
<td>1488</td>
<td>537</td>
<td>2893</td>
<td>58.825%</td>
</tr>
<tr>
<td>pred. rough</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>class recall</th>
<th>accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>88.175%</td>
<td>52.491%</td>
</tr>
</tbody>
</table>

Table 5: Confusion matrix of a the best performing classification which considered 3 segments during its classification

The classifier with the best accuracy for surface classification achieves \( \approx 75\% \) the classifiers from the previous sections which directly learn the labels from the training data perform much worse. For the extended bump classification the K-NN classifier achieves 61\% accuracy. A random classifier with the same label distribution performs with \( \approx 57\% \) accuracy.

The confusion matrix of the extended bump classifier explains why the accuracy is not higher. The classifier is quite good for smooth data, but it confuses rough and bumpy data. A closer look and comparison with the recorded variances in Figure 3 reveals that most probably, the labeling was not consistent in assigning the labels "rough" and "bumpy".

For each label class the figure shows the manual labels (light gray bars) and the predicted labels (dark gray bars). It can be seen that the light gray labels for the rough class are not modeled with sufficient detailness (on the left side of the diagram). It can also be seen from the acceleration values that this label contains parts of different labels like smooth and bumpy which were not correctly labeled. The diagram shows that the classifier indeed is more often correct than the manual label which is unfortunately the reference for the performance. This is the main reason for the "bad" performance of the classifiers and explains also the confusion matrix (table 5).

### 2.3 Classifier implementation on the smartphone

In this section we will discuss the runtime of the whole classification process which was implemented in J2ME. The one of initial goals of this work is to make the classification process possible in the online mode of the client.

Once learned, the classifier has to execute the following steps online on the smartphone:

- calculate the mean, variance and standart deviation of all previous absolute acceleration vector values
- assemble classification data
- applies Naive Bayes classifier for bump detection
- put prediction to bump LIFO (these LIFO stores previous classifications, which are needed to calculate surface prediction)
- put GPS coordinates and prediction for this segment to ObservationBuilder
- builds observation
- sends observation

The execution time of the learned classifier took less than 2 ms in a JME implementation on a Nokia 5800 with an ARM CPU execution at 400 Mhz. The accelerator delivered data at 37 Hz, resulting in 37 values which must be evaluated at each GPS point (given that GPS is running at 1Hz). This means that the overall impact of the classifier on the device performance was very low and that classifier execution finished safely before the next accelerometer values came in.

### 3 Conclusion

It was shown that in general a surface and a bump classification can be realized via a machine learning approach. It was shown how the data must be preprocessed to achieve good classification results and which features play an important role in this classification process. At the current state, the classification is not as good as it could be. We showed that the correctness and accuracy of the labels in training data should be improved for training a machine learning algorithm. However, we also achieved very good bump detection. The learned classifier is fast enough to be executed online on a moderately fast smartphone hardware and needs no further learning or labeling. Surface classification may derived from this. As the classifier performed best for short segments, mainly based on the variance of the length of the acceleration vector, we...
also see a good chance for just time-series based analysis approaches such as used in [Mednis et al., 2012] or [Mladenov and Mock, 2009] to be applied for road surface classification. As application, biking communities can profit from the presented approach for displaying route quality information on a community portal, or cycling-friendly cities can monitor the surface quality of their cycling route network for detecting damage and initiating road repair.

Acknowledgements

The research leading to these results has received funding from the European Union’s Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 255951 (LIFT Project).

References


Simulating Price Interactions by Mining Multivariate Financial Time Series

Bruno Silva  
DSI/ESTSetúbal  
Instituto Politécnico de Setúbal  
Portugal  
bruno.silva@estsetubal.ips.pt

Luis Cavique  
Universidade Aberta  
Portugal  
lcavique@univ-ab.pt

Nuno Marques  
DI/FCT - UNL  
Portugal  
nmm@di.fct.unl.pt

Abstract

This position paper proposes a framework based on a feature clustering method using Emergent Self-Organizing Maps over streaming data (UbiSOM) and Ramex-Forum – a sequence pattern mining model for financial time series modeling based on observed instantaneous and long term relations over market data. The proposed framework aims at producing realistic monte-carlo based simulations of an entire portfolio behavior over distinct market scenarios, obtained from models generated by these two approaches.

1 Introduction

Grasping the apparently random nature of financial time series has proven to be a difficult task and countless methods of forecasting are presented in literature. Nowadays, this is even more difficult due to a global economy with strong interconnections. Most traders forecast future price using some combination of fundamentals, indicators, patterns and experience in the expectation that recent history will forecast the probable future often enough to make a profit. Detecting correlations between financial time series and being able to simulate both short and long term interactions in virtual scenarios using models extracted from observed market data can provide an increasingly needed tool to minimize risk exposure and volatility for a given portfolio of securities. This position paper argues that feature clustering methods using Emergent Self-Organizing Maps over streaming data (UbiSOM) [Silva et al., 2012], [Silva and Marques, 2010b] can be conjoined with Ramex-Forum – a sequence pattern mining model [Marques and Cavique, 2013], for financial time series modeling based on observed instantaneous and long term relations over market data. Since the lower the correlation among the individual securities, the lower the overall volatility of the entire portfolio, this makes possible to propose a tool to minimize risk exposure and volatility for a given portfolio of securities. The proposed framework aims at producing more realistic Monte Carlo-based simulations of the entire portfolio behavior over distinct market scenarios, obtained from models generated by these two approaches.

2 Proposed Framework

The proposed modular framework is depicted in Figure 1 and consists of i) The UbiSOM, an ESOM algorithm tailored from streaming data ii) The Ramex-Forum, a sequence pattern mining model and iii) A Monte Carlo-based simulator. The first two are fed with a stream of log-normalized raw asset prices, which are then used by the third module to produce future different and possible market scenarios, based on the observed data. The UbiSOM can model instantaneous short-term correlations between the various assets and its topological map (Section 2.1) can be used as a starting point to generate alternate time-series based on a trajectory model (Section 3.1) by the simulation module. The input from the Ramex-Forum module should be useful to incorporate in the simulations long-term dependencies between the assets to produce more realistic market scenarios.

2.1 Emergent Self-Organizing Maps

Self-Organizing Maps [Kohonen, 1982] can use the ability of neural networks to discover nonlinear relationships in input data and to derive meaning from complicated or imprecise data for modeling dynamic systems such as the stock market. The Self-Organizing Map (SOM) is a single layer feed-forward network where the output neurons are arranged in a 2-dimensional lattice, whose neurons become specifically tuned to various input vectors (observations) in an orderly fashion. Each input is connected to all output neurons and attached to each neuron there is a weight vector with the same dimensionality as the input vectors. These weights represent prototypes of the input data. The topology preservation of the SOM projection is extensively used by focusing SOM on using larger maps – ESOM [Ultsch and Her-
A previous work [Silva and Marques, 2010a] showed that ESOMs provide a way of representing multivariate financial data on two dimensions and use a viable tool to detect instantaneous short-term correlations between time-series. We illustrate this in Section 3, within our preliminary results. Additionally, and supported by the detected correlations, the topological ordered map can be used as a good starting point to generate realistic multivariate financial data based on the short-term relationships.

2.2 The Ramex-Forum Algorithm

Ramex-Forum solves the problem of huge number of rules that avoid a global visualization in many pattern discovery techniques (e.g., [Agrawal and Srikant, 1995]). Ramex-Forum is a sequential pattern mining algorithm that includes a two-phase: the transformation phase and the search phase. In the transformation phase the dataset is converted into a graph where cycles are allowed. The raw data must be sorted in such a way that each time interval can be identified. In the search phase the maximum weighted spanning poly-tree is found. A poly-tree is a direct acyclic graph with one path between any pair of nodes at the most. The in-degree of any vertex of a tree is 0 (the root) or 1. On the other hand, the in-degree of a vertex can be greater than 1. A maximum weighted spanning poly-tree is the spanning poly-tree with a weight that is upper than or equal to the weight of every other spanning poly-tree. The Ramex-Forum algorithm develops a new heuristic inspired in Prim’s algorithm [Prim, 1957] and assures a new way of visualization long term patterns in polynomial execution time.

3 Preliminary Results

In this section we provide a proof-of-concept of the proposed methodology within the framework. The proposed method is illustrated with historical data representing the world economy in the recent past (years 2006 to 2012) – Figure 2. The huge economic changes during this period are good to show the usefulness of data mining algorithms over financial data. Top financial products such as average Indexes for companies based in different countries (DJI – Dow Jones, in the United States; BVSP – Bovespa, in Brazil; FCHI, Euronext in Paris; N225, Nikkei in Japan; the HSI — Hang Seng Index, in Hong Kong; and DAX, German Index) and relevant commodities exchange-traded funds (ETF) such as United States Oil Fund (USO) and GLD for a physically backed gold ETF, were considered.

Each time series is considered a feature of the training data, i.e., observations are the prices of the financial products for consecutive days. After performing a logarithmic normalization of the values, so that the specific range of each asset price is disregarded, the historical data forms the training dataset that is fed into the UbiSOM and Ramex-Forum modules. The trained UbiSOM map contains a topologically organized projection of the historical data. Correlations between individual time-series are extracted through a visualization technique for the UbiSOM map. By component plane representation we can visualize the relative component distributions of the input data. Component plane representation can be thought as a sliced version of the UbiSOM. By component plane representation we can visualize the relative component distributions of the input data. Component plane representation can be thought as a sliced version of the UbiSOM. Each component plane has the relative distribution of one data vector component. In this representation, dark values represent relatively large values while white values represent relatively small values. By comparing component planes we can see if two components correlate. If the outlook is similar, the components strongly correlate. The component planes for the resulting trained map are represented in Figure 3. Visual inspection
may suffice to detect correlations, but in [Silva and Marques, 2010a] we provided an automated algorithm to cluster time-series based on a distance metric computed for pairs of component planes (Figure 4). However, this ability is only of relevance in this paper to justify the use of ESOM maps to generate multivariate time-series based on a trajectory model (Section 3.1).

Figure 5 presents a Ramex-Forum generated graph for this financial data, considering interactions with a latency of up to 160 (long-term) trading days over a period of 2000 days – results presented in [Marques and Cavique, 2013]. Each arc represents the number of synchronous positive price tendencies (buying signals given by a moving average indicator). During the studied period Hong Kong HSI Index has a behavior that was preceded by 273 times by similar variations in American Dow Jones (DJI) and 179 times by German DAX. We should notice that USA DJI is influencing most major assets in the world. The only exception to this is European German DAX, that strongly co-influences Chinese HSI. Another correlation found is between HSI and GLD tendencies in these long term dependencies. This is something that UbiSOM cannot capture and can be incorporated when generating more realistic market scenarios.

3.1 Generating Scenarios

By projecting again the historical data over the UbiSOM we get a set of trajectories over the map that are used to generate these alternate time series. A trajectory is formed by projecting two consecutive observations from the training data and storing the pair of neurons that were activated, in the form of a trajectory (bare in mind that loops are frequent, because two similar observations are prone to be projected in the same neuron). Figure 6 depicts these trajectories in the form of a directed graph in which each vertex represents a neuron and the edges the obtained trajectories. The weight of the edge indicates how many times the trajectory was followed in the projection of the training data.

Based on this trajectory model, we can generate alternate time series using Monte Carlo simulations. Starting at a vertex with edges and randomly choosing the next trajectory of the model to follow we can create paths of arbitrary lengths (dependent of the desired number of daily prices). Each vertex/neuron contains the prototype of data that contains the set of daily prices for similar observed days. The totality of the path then gives us the multivariate time series. Details on how to generate the path are currently being studied, it must not be totally random, i.e., the weight on the edges must be taken into account so as to give more importance to trajectories that are more common. Also, when creating the trajectory model we store at each vertex/neuron the statistical variation of the training vectors that are projected on that particular neuron. This allows generating a Gaussian around each prototype vector component to introduce variability on a particular virtual daily price. This is particularly important when loops are being followed in the path, so that generated time series doesn’t contain “flat” lines.

Figure 7 depicts a sample of a generated outcome that can be obtained from trajectories over the trained UbiSOM. It can be seen that the multivariate time series maintain the observed correlations in the original historical data. This can be very useful in generating possible scenarios for risk estimation.

3.2 Discussion

Visual inspection of the similarity of component planes in Figure 3 shows that the results of the UbiSOM model are coherent, e.g., DJI and DAX are strongly correlated in their historical behavior. GLD, as it was expected, is very far from any other financial product. Its historical behavior is extremely different in the analysis period, mainly always in an upward movement. All the other assets maintain a significant distance from the others, showing that the correlation is not that strong. Interesting additional long term relations were found by Ramex-Forum algorithm. The example shown in Figure 5, presents the USA DJI index influencing most other indexes. Also China (HSI) is detected as a major player in world economy and seems to be the major influence on the price of gold (GLD). Indeed, during the analyzed period, People’s Republic of China was one of the major buyers of gold in the world and has the largest reserves of Gold and Foreign Exchange in the world (CIA World Factbook, 2013).

However, it is the conjunction of the long term relations detected in Ramex-Forum with the magnitude of short time multivariate dependencies of UbiSOM, that should be the most interesting application. Different long term trajectories can now be generated on the UbiSOM map, based on the long term sequences detected by Ramex-Forum. E.g., neurons cor-
Figure 6: Trajectories generated for the projection of the historical training data over a $15 \times 10$ trained UbiSOM.
responding to highest increases in gold values can be easily selected from a SOM map. The same could be done for high values for the Chinese (HSI), or USA (DJI) economy. Highly probable pathways should then be made among those neurons. In practice these will encode the Ramex-Forum graph as a probable pathway among distinct UbiSOM neurons. Then for a given trading day (e.g. today) starting point, we can then generate random walks in the map. Since each neuron represents a possible market state, we can easily generate for each neuron a possible virtual trading day that is strongly related with observed data. However it will be the pathways to provide the most interesting effect on this map. Indeed, in average, virtual trading days will follow possible sequences given (and measured) by Ramex-Forum graph.

4 Conclusions

Both algorithms should provide a realistic and easily usable framework to study and simulate possible effects of either economic or political decisions. Even extreme events, e.g., Acts of God, may then be given some probability. We believe that such a time-series based model is a much needed tool in todays strongly interdependent and complex world where over-simplistic assumptions frequently lead to poor decisions.

On one hand UbiSOM provides the daily correlation between products and can be made self-adjustable to continuously changing streams of data (i.e., both collaborative learning [Silva and Marques, 2010b] and detecting concept drift [Silva et al., 2012]). On the other hand Ramex-Forum graphs shows sequences of the more representative events and can be easily used to model the dynamic of the occurrences. So, we believe that these complementary tools, one more static and the other more dynamic, can intrinsically guarantee realistic modeling on different scenarios and provide a major breakthrough in decision support systems.

References


Trend template: mining trends with a semi-formal trend model

Olga Streibel, Lars Wißler, Robert Tolksdorf, Danilo Montesi
streibel@inf.fu-berlin.de, lars.wissler@googlemail.com, tolk@ag-nbi.de
Networked Information Systems Group, Freie Universität Berlin, Berlin, Germany
montesi@cs.unibo.it
University of Bologna, Bologna, Italy

Abstract
Predictions of uprising or falling trends are helpful in different scenarios in which users have to deal with huge amount of information in a timely manner, such as during financial analysis. This temporal aspect in various cases of data analysis requires novel data mining techniques. Assuming that a given set of data, e.g. web news, contains information about a potential trend, e.g. financial crisis, it is possible to apply statistical or probabilistic methods in order to find out more information about this trend. However, we argue that in order to understand the context, the structure, and explanation of a trend, it is necessary to take a knowledge-based approach. In our study we define trend mining and propose the application of an ontology-based trend model for mining trends from textual data. We introduce the preliminary definition of trend mining as well as two components of our trend model: the trend template and the trend ontology. Furthermore, we discuss the results of our experiments with trend ontology on the test corpus of German web news. We show that our trend mining approach is relevant for different scenarios in ubiquitous data mining.

1 Introduction
When discussing trends some of us may think about the ups and downs of NASDAQ\(^1\), or DAX\(^2\) curves, or changes in public opinion on politics before elections. Likewise, one can think about web trends, life style trends or daily trends, i.e. hot topics, in the news or on social networks. Changes in a mobile data stream also fall within the definition of a trend. Understanding a trend as a hot topic is related to the research in Emerging Topic Detection (EDT) and Topic Detection and Tracking (TDT), the subfields of information retrieval [Allan, 2002][Kontostathis et al., 2003]. A trend is defined there as a topic that emerges in interest and utility over time. Accordingly, common examples of trends may be the “Arab Spring” which emerged in political news worldwide in the beginning of 2011, as well as the financial and real estate crisis which started to emerge on business news worldwide in 2008. A graphical representation of a trend, based on GoogleTrends\(^3\), is shown in Fig. 1.

Figure 1: This graph shows a search volume index for the terms “financial crisis” (blue curve) and “insolvent” (red curve) in Germany from 2006 to 2011. Source: GoogleTrends

Several methods have been proposed for detecting trends in texts or discovering trends in the web news (see Section 3). Other works provide approaches from statistics and time series analysis that can be applied for analyzing trends in non-textual data. Our work contributes to the general understanding of trend mining that we see as highly relevant to ubiquitous data mining. In this paper, we explain our abstract concept of a trend template and go on to describe a trend ontology which is an instance of the trend template.

2 Ubiquitous data mining and trend mining
The Ubiquitous Data Mining (UDM) is defined as the essential part of the ubiquitous computing [Witten and Eibe, 2005]. The UDM techniques help in extracting useful knowledge from data that describes the world in movement, including the aspects of space and time. Time is the necessary dimension for trend mining—there is no trend without time. And a trend is one of the aspects of a world in movement. Before we discuss general trend characteristics, we want to mention the sociological and statistical perspectives on the trend, as well as define trend mining. This helps in understanding the trend characteristics that create the basis for the definition of our trend template later in this paper.

---

\(^1\)http://www.nasdaq.com/ online accessed 04-17-2013
\(^3\)http://www.google.com/trends/ online accessed 04-17-2013
2.1 Trend from different perspectives

Detecting trends from the sociological point of view is an analytical method for observing changes in peoples behavior over time with regard to “six attitudes towards trends” [Vejlgaard, 2008]. The definition of these six attitudes is based on eight different personality profiles of groups who participate in the trend process: trend creators, trend setters, trend followers, early mainstreamers, mainstreamers, late mainstreamers, conservatives and anti-innovators.

Detecting trends from the statistics perspective is based on trend analysis of time-series data with two goals in mind: “modeling time series (i.e. to gain insight into the mechanisms or underlying forces that generate the time series) and forecasting time series (i.e. to predict the future values of the time-series variables)” [Han and Kamber, 2006]. The trend analysis process consists of four major components: trend or long-term movements, cyclic movements or cyclic variations, seasonal movements or seasonal variations, and irregular or random movements [Han and Kamber, 2006]. A trend, in this context, is an indicator for a change in the data mean [Mitsa, 2010].

2.2 Trend mining

Since data mining can be described as “the extraction of implicit, previously unknown, and potentially useful information from data” [Witten and Eibe, 2005], we propose the use of the term trend mining as defined below:

**DEF 2.1 Trend mining** is the extraction of implicit, previously unknown and potentially useful knowledge from time-ordered text or data. The trend mining techniques can be used for capturing trend in order to support user in providing previously unknown information and knowledge about the general development in users field of interests.

3 Related Research

In general, when mining trends from textual data, at least the following three research areas should be mentioned: emergent trend detection, topic detection and tracking, and temporal data mining.

In [Kontostathis et al., 2003] several systems that detect emerging trends in textual data are presented. These ETD systems are classified into two main categories: semi-automatic and fully-automatic. For each system there is a characterization based on the following aspects: input data and attributes, learning algorithms and visualization. This comparison includes an overview over the research published in [Allan et al., 1998][Lent et al., 1997][Agrawal et al., 1995][Swan and Jensen, 2000][Swan and Allan, 1999][Watts et al., 1997]. TDT research [Allan, 2002] is predominantly related to the event-based approaches. Event-based approaches for trend mining underlie the assumption that trends are always triggered by an event, which is often defined as “something happening” or “something taking place” [Lita Lundquist, 2000] in the literature. Considering a trend from the event research perspective means that trend detection has to be understood as a monitoring task. This is mostly the case for so-called short-term trends that are indeed triggered by some events and in order to detect them we have to monitor the stream in which they occur, e.g. the occurrence of “Eyjafjallajökull eruption” which was reported in social networks and on the news in March 2010. However, so-called long-term trends, e.g. “financial crisis”, that started to be on-topic in 2008 are not necessarily conjoined with one specific event. It is more a chain of events or even the “soft” indicators as public opinion or news. No sharp distinction has been made between the TDT and ETD research fields, which means that some research such as [Swan and Allan, 1999] or [Lavrenko et al., 2000] can be in fact classified into both fields. Temporal data mining research [Mitsa, 2010] offers methods for clustering, classification, dimension reduction and processing of time-series data [Wang et al., 2005]. It addresses in general the temporal data and the techniques of time series analysis on these data. One definition of temporal data is “time series data which consist of real valued sampled at regular time intervals” [Mitsa, 2010]. Temporal data mining applies the data mining methodology and deals with the same approaches for classification or clustering, that are relevant for mining trends in textual data.

4 Trend template

Based on our experiments and considerations, we outline the following assumptions about trends in the general context of this work;

A trend can be described by the following characteristics: trigger, context, amplitude, direction, time interval, and relation. Fig. 2 illustrates the trend template. In 4.1, we more precisely define each characteristic.

4.1 Definitions

**Trigger** is a thing. They can be: an event, a person, or a topic anything that triggers the trend. A trigger can but does not have to cause a trend. A trigger makes the trend visible. An example of a trigger is *Lehman Brothers*’ insolvency that can be classified as both a topic and an event.

**Context** is the area of the trigger. If the trigger is a topic then the context is this topic’s area, e.g. *Lehman Brothers insolvency* is mentioned in the context of real estate market.

**Amplitude** is the strength of a given trend. It can be expressed by a number, the higher the number the more impact the trend has or by a qualitative value that describes the trend phase, e.g. beginning (setter), emerging (follower), mainstream, fading (conservative).

**Time** is necessary while spotting trend, since there can be no trend without time. It is the interval in which the trend is appearing, independent from the amplitude, e.g. the real estate crisis appeared between the years 2008-2011.

**Relation** expresses the dependency between the trigger and the context, it puts the given trigger, e.g. *Lehman Brothers insolvency* within the given context of the real estate crisis in a relation, e.g. *Lehman Brothers insolvency is part of the*...
4.2 Formal description

The trend template is an abstract model that describes the main concepts that are important and necessary for knowledge-based trend mining. In following, we more explicitly define the trend template:

DEF. 4.1: Trend template (TT) is a quintuple:

$$TT := (T, C, R, TW, A)$$

where: $T$ is trigger, $C$ is context, $R$ is relation, $TW$ is time window, and $A$ is amplitude.

DEF. 4.2: Trigger is set of concepts:

$$T := \{t_0, \ldots, t_n\}, n \in \mathbb{N} \land t \in T$$

so that if $E$, $P$, $T_o$ are the sets defining:

- events: $E := \{e_0, \ldots, e_n\}, n \in \mathbb{N} \land e \in E$
- persons: $P := \{p_0, \ldots, p_n\}, n \in \mathbb{N} \land p \in P$
- locations: $L := \{l_0, \ldots, l_n\}, n \in \mathbb{N} \land l \in L$
- topics: $T_o := \{t_{o0}, \ldots, t_{on}\}, n \in \mathbb{N} \land t_o \in T_o$

then:

$$T := E \cup P \cup T_o \cup L$$

DEF. 4.3: Context is a union set consisting of a set of concepts and a set of relations between them where $c$ is a context element:

$$C := C_{co} \cup R_{co}, c \in C$$

with $C_{co}$ the set of concepts

$$C_{co} := \{c_{co0}, \ldots, c_{con}\}, n \in \mathbb{N} \land c_{co} \in C_{co}$$

and $R_{co}$ the set of relations:

$$R_{co} := \{r_{co0}, \ldots, r_{con}\}, n \in \mathbb{N} \land r_{co} \in R_{co} \land R_{co} \subseteq C_{co} \times C_{co}$$

where $r_{co}$ defines a binary relation:

$$r_{co} : c_{coz}, c_{coy} \rightarrow r_{co}(c_{coz}, c_{coy}) \land c_{coz} \neq c_{coy}$$

and the context element is defined by:

$$c = c_{co} \cup (c_{co1}, c_{co2})$$

$$C = C_{co} \cup C_{co} \times C_{co}$$

DEF. 4.4: Relation is a set of relations:

$$R := \{r_0, \ldots, r_n\}, n \in \mathbb{N} \land r \in R \land R := \{T \times C\}$$

with

$$r_i : t_i, c_i \rightarrow r_i(t_i, c_i)$$

DEF. 4.5: Time window is a function that assigns time slice to the time points:

$$TP := \{t_{point}\mid t_{point} = ms \lor second \lor minute \lor hour \lor day \lor month \lor year\}$$

$$TS := \{t_{point0}, \ldots, t_{pointn}\}$$

$$TW : TP \rightarrow TS$$

DEF. 4.6: Amplitude is a function that assigns a value to the quadruple of $(T, C, R, TW)$

$$A : T \times C \times R \times TW \rightarrow \mathbb{N} \cup V$$

where $N$ is the set of natural numbers and $V$ is the set of categorical values

$$a : (t, c, r, tw) \rightarrow n \lor v$$
5 Trend Ontology

One way of implementing the trend template is the realization of this model in the form of an ontology. We can understand the ontology as an instance of the trend template.

Based on the trend template described above, we created an applicable model, using SKOS\(^6\) and RDFS/OWL\(^7\) concepts and properties. Our model serves as a general model that can be extended regarding the particular application domain and applied for annotating a text corpus in order to retrieve the trend structure. The trend ontology is divided into levels meta, middle and low which correspond to three abstract layers of the model. Whereas the low level and the middle level relate to the corresponding application domain (in our case it is the German Stock Exchange, DAX), the meta level is the most interesting one. Meta ontology incorporates the general trend characteristics and can be applied to any application domain.

The central concepts of the ontology are Trigger, TriggerCollection, Indication, Relational and ValuePartition and have been modeled as subconcepts of skos:Concept, skos:Collection and time:TemporalEntity, with different semantic construction, e.g. skos:related, skos:member. The concepts mirror the composition of the trend template. Trigger consists of three subconcepts: event, person, location. The main goal of the meta ontology is to offer all necessary concepts and relations in order to span the trend template as a structure over a text corpus. To actually translate a specific document corpus into such a structure, meta ontology needs to be combined with a domain specific trend ontology which defines domain specific concepts, their keywords and possibly also their relations. This can either be done manually by extracting common terms as keywords and linking them to their respective concepts, or automatically by entity recognition. The pseudocode 6.1 describes the algorithm that we applied to build up the trend description on the test corpus.

6 Experiments

The text corpus which we call German finance data\(^8\) that served as our test corpus consists of about 40,500 news articles related to the fields of business and finance, provided as XML files. The corpus is available in German and provides news articles from January 2007 to May 2008. The text was parsed in cooperation with neofonie\(^9\) from the following sources: comdirect\(^10\), derivatecheck\(^11\), Handelsblatt\(^12\), GodmodeTrader\(^13\), Yahoo\(^14\), Financial Times Deutschland\(^15\), and finanzen.net\(^16\) in German and provides news articles from January 2007 to May 2008. The text was parsed in cooperation with neofonie\(^9\) from the following sources: comdirect\(^10\), derivatecheck\(^11\), Handelsblatt\(^12\), GodmodeTrader\(^13\), Yahoo\(^14\), Financial Times Deutschland\(^15\), and finanzen.net\(^16\).

In general, the content of the corpus is focused on finance and business information concerning German companies and stocks. It focuses on the situation at DAX, as well as on reviews and ratings of German companies and shares. For evaluation purposes regarding usefulness and practicability, the trend ontology has been filled with two different parts of the test corpus: stock market specific documents in Part 1 and the general business news in Part 2 (subsequently first and second part). They contain over 5,000 and 16,000 documents respectively. We specified several basic questions and respective queries as relevant for trends in general and specifically for stock market trends. Querying the ontology for the total occurrence of concepts yields the following output (shortened to some of the most relevant concepts): Germany (9,137), USA (4,808), Deutsche Telekom (442), Allianz (433), Switzerland (382), Starbucks (104). The output corresponds directly to the corpus of German stock news with a clear focus on German companies followed by the still dominant US market. A similar query for often mentioned lines of business in the context of Germany in contrast to the USA yields a major focus on the industry for Germany. 4.5% to 7.1% of the total occurrences of Germany appear in the context of different lines of industry. The USA is strong in the context of IT (9%) and services (6.9%). Moreover, we checked so-called topic structure by using our ontology. Here a general example for the concept Germany:

```
trendonto:#Germany (9137) has Topic
trendonto:#Financial : 1142
trendonto:#buy : 1003
trendonto:#MachineBuildingIndustry : 650
trendonto:#Share : 606
trendonto:#StockPrice : 562
trendonto:#Up : 520
trendonto:#Industry : 510
trendonto:#Investment : 468
trendonto:#Supplier : 422
trendonto:#AutomobilIndustry : 414
```

Algorithm 6.1: createTrendDescription(c, o)

```
comment: parse ∀ document ∈ corpus

comment: into ontology

parse(c, inO, outO) {
    model.read(inO)
    create.reasoner(inO)
}

for each d ∈ c do {
    pass(keywords);
    matches.add(match.model(keywords, inO)) {
        for keyword ← 0 to i {
            if inO.concept.label == keyword or
               keyword ∈ inO.concept.label
               keyword prefix or keyword postfix ==
               inO.concept.label prefix or postfix
            then matches.add(keyword)
        }
        relate.model(matches, inO)
        if model.getRelation(matches).is.Empty
        then model.createRelation(matches)
        else model.incCounter(matches)
    }
    model.write(outO)
}
```

6 http://www.w3.org/2004/02/skos/ online accessed 04-17-2013
7 http://www.w3.org/TR/owl-features/ online accessed 04-17-2013
8 Currently (May 2013) in the publishing process at Linguistic Data Consortium http://www.ldc.upenn.edu/
9 http://www.neofonie.de, online accessed 04-25-2012
10 http://www.comdirect.de/index.html, online accessed 04-25-2012
11 http://derivatecheck.de, online accessed 04-25-2012
12 http://www.handelsblatt.com/weblog/, online accessed 04-25-2012
13 http://www.godmode-trader.de, online accessed 04-25-2012
14 http://de.biz.yahoo.com, online accessed 04-25-2012
15 http://www.finanzen.net, online accessed 04-30-2012
16 Online accessed 04-25-2012
In Fig. 3 we show the comparison of the performance values for the stock markets as ranked by ontology (test based on time window: July 2007 to April 2008) and reported in real (time window July 2007 to July 2011). Applying the trend ontology to the test set enables to find out specific information about the certain trend that is described in the documents of the test set. Our preliminary experiments results that we partially present in this paper show that our idea of a trend template could help in harvesting knowledge from the given test data in a timely manner.

7 Conclusions and future work

This paper presents our research on knowledge-based trend mining, wherein the main contribution is our semi-formal model of a trend template. We showed that the implementation of the trend template in the form of a trend ontology allows for capturing the trend structure out of a test document set. Our experiments confirm that a knowledge-based approach for mining trends out of data allows for extended trend explanations. Currently we are comparing the trend ontology experiment results with the results from adapted K-Means clustering and LDA-based topic modeling algorithms applied on our test set.

Acknowledgments

This work has been partially supported by the “InnoProfile-Corporate Semantic Web” project funded by the German Federal Ministry of Education and Research (BMBF) and the BMBF Innovation Initiative for the New German Länder - Entrepreneurial Regions.

References


Ubiquitous Self-Organizing Maps

Bruno Silva
DSI/ESTSetúbal
Instituto Politécnico de Setúbal
Portugal
bruno.silva@estsetubal.ips.pt

Nuno Marques
DI/FCT - UNL
Portugal
nmm@di.fct.unl.pt

Knowledge discovery in ubiquitous environments are usually conditioned by the data stream model, e.g., data is potentially infinite, arrives continuously and is subject to concept drift. These factors present additional challenges to standard data mining algorithms. Artificial Neural Networks (ANN) models are still poorly explored in these settings.

State-of-the-art methods to deal with data streams are single-pass modifications of standard algorithms, e.g., K-means for clustering, and involve some relaxation of the quality of the results, i.e., since the data cannot be revisited to refine the models, the goal is to achieve good approximations [Gama, 2010]. In [Guha et al., 2003] an improved single pass k-means algorithm is proposed. However, k-means suffers from the problem that the initial k clusters have to be set either randomly or through other methods. This has a strong impact on the quality of the clustering process. CluStream [Aggarwal et al., 2003] is a framework that targets high-dimensional data streams in a two-phased approach, where an online phase produces micro-clusterings of the incoming data, while producing on-demand offline models of data also with k-means.

In this position paper we address the use of Self-Organizing Maps (SOM) [Kohonen, 1982] and argue its strengths over current methods and directions to be explored on its adaptation to ubiquitous environments, which involve dynamic estimation of the learning parameters based on measuring concept drift on, usually, non-stationary underlying distributions. In a previous work [Silva and Marques, 2012] we presented a neural network-based framework for data stream mining that explored the two-phased methodology, where the SOM produced offline models. In this paper we advocate the development of a standalone Ubiquitous SOM (UbiSOM), that is capable of producing models in an online fashion, to be integrated in the framework. This allows derived knowledge to be accessible at any time.

The Self-Organizing Map is a well-established data-mining algorithm with hundreds of applications throughout enumerate scientific domains for tasks of classification, clustering and detection of non-linear relationships between features [Oja et al., 2003]. It can be visualized as a sheet-like neural-network array, whose neurons become specifically tuned to various input vectors (observations) in an orderly fashion. The SOM is able to project high-dimensional data onto a 2D lattice, while preserving topological relationships among the input data, thus electing it as a data-mining tool of choice [Vesanto, 1999], either for clustering, data inspection and/or classification. The powerful visualization techniques for SOM models allow the detection of complex cluster structures, detection of non-linear relationships between features and even allow the clustering of time series.

As with most standard data mining methods, classical SOM training algorithms are tailored to revisit the data several times to build good models. As training progresses, existing learning parameters are decreased monotonically over time through one of a variety of decreasing functions. This is required for the network to converge to a topological ordered state and to estimate the input space density. The consequence is that the maps lose plasticity over time, i.e., if a training sample presented at a later point in time is very different from what it has learned so far it does not have the ability to represent this new data appropriately because these parameters do not allow large updates at that time. In ubiquitous environments data is expected to be presented to the network over time, i.e., the network should be learning gradually and derived knowledge should be accessible at any time. This means that the SOM must be able to retain an indefinite plasticity over time, with the ability to incorporate very different data from what it has learned at a particular time, i.e., to be in conformance with the “dynamic environment” requirement.

Ubiquitous SOMs, i.e., self-organizing maps tailored for ubiquitous environments with streaming data, should define those parameters not based on time t, but in the error on the network for a particular observation.

An underused variant of the SOM, called the parameterless SOM (PLSOM) [Berglund, 2010], was first introduced to address the difficulty of estimating the initial learning parameters. The PLSOM has only one parameter β (neighborhood range) that needs to be specified in the beginning of the training process, after which α (learning rate) and σ (neighborhood radius) are estimated dynamically at each iteration. The basic idea behind the PLSOM is that for an input pattern that the network already represents well, there is no need for large adjustments – learning rate and neighborhood radius are kept small. On the other hand, if an input vector is very dissimilar of what was seen previously, then those parameters are adjusted to produce large adjustments. However, in its current form, it fails in mapping the input space density...
Figure 1: Learned Gaussian distribution for the classical SOM (left) and for the PLSOM (right). The later does not maintain the density of the input space, which undermines the use of visualization techniques for cluster detection and feature correlation.

onto the 2D lattice (Figure 1). This undermines the visualization capabilities of the PLSOM, namely for cluster detection. Also, by estimating the values of the learning parameters solely based on the network error for a given observation, it is very sensible to outliers.

Nevertheless, this variant of the SOM retains an indefinite plasticity, which allows the SOM to react to very different input samples from what has been presented to it, at any point in time; and converges faster to an initial global ordered state of the lattice. These two capabilities makes PLSOM an interesting starting point for the proposed goal.

Concept drift means that the concept about which data is being collected may shift from time to time, each time after some minimum permanence. Changes occur over time. The evidence of drift in a concept is reflected in the training samples (e.g., change of mean, variance and/or correlation). Old observations, which reflect the behavior in nature in the past, become irrelevant to the current state of the phenomena under observation [Gama, 2010]. In [Silva et al., 2012] we addressed concept drift detection using a different type of neural network, namely Adaptive Resonance Theory (ART) networks. Figure 2 illustrates its applicability to financial time series. It works by measuring the quantization error of the last built micro-cluster ART model, over a predefined number of previous ones. We propose to use the ideas of the PLSOM algorithm using the network error as an input to a concept drift module, either ANN-based or not. While the concept is stable, the learning parameters are being decreased monotonically so as to map the input space density; when the concept begins to drift the parameters are adjusted to higher values so as to cope with the different observations. If the, possibly, underlying non-stationary distribution is drifting rapidly, maintaining higher learning parameters will, consequently, make the model “forget” old and irrelevant observations to the current state.

Conclusion ANN methods exhibit some advantages in ubiquitous data mining: they have the ability to adapt to changing environments; have the ability to generalize from what they have learned; and through the ANN error it is possible to determine if the new information that arrives is very different from what it has learned so far. A purely online Ubiquitous Self-Organizing Map (UbiSOM) that can learn non-stationary distributions is relevant for data stream mining, namely because of its. The SOM mining capabilities greatly surpass K-means, without introducing a big overhead in computation needs. Measuring concept drift as a way to estimate the learning parameters of the learning algorithm is, in our belief, a promising path.

References