Integrating Community Matching and Outlier Detection for Mining Evolutionary Community Outliers

Manish Gupta
University of Illinois at Urbana-Champaign
gupta58@illinois.edu

Jing Gao
State Univ. of New York at Buffalo
jing@buffalo.edu

Yizhou Sun
University of Illinois at Urbana-Champaign
sun22@illinois.edu

Jiawei Han
University of Illinois at Urbana-Champaign
hanj@illinois.edu

ABSTRACT

Temporal datasets, in which data evolves continuously, exist in a wide variety of applications, and identifying anomalous or outlying objects from temporal datasets is an important and challenging task. Different from traditional outlier detection, which detects objects that have quite different behavior compared with the other objects, temporal outlier detection tries to identify objects that have different *evolutionary behavior* compared with other objects. Usually objects form multiple communities, and most of the objects belonging to the same community follow similar patterns of evolution. However, there are some objects which evolve in a very different way relative to other community members, and we define such objects as *evolutionary community outliers*. This definition represents a novel type of outliers considering both temporal dimension and community patterns. We investigate the problem of identifying evolutionary community outliers given the discovered communities from two snapshots of an evolving dataset. To tackle the challenges of community evolution and outlier detection, we propose an integrated optimization framework which conducts outlier-aware community matching across snapshots and identification of evolutionary outliers in a tightly coupled way. A coordinate descent algorithm is proposed to improve community matching and outlier detection performance iteratively. Experimental results on both synthetic and real datasets show that the proposed approach is highly effective in discovering interesting evolutionary community outliers.

Categories and Subject Descriptors

H.1.0 [**Information Systems**]: Models and Principles; H.4.m [**Information Systems Applications**]: Miscellaneous

General Terms

Algorithms, Experimentation

Keywords

community matching, evolutionary community outliers, anomaly detection, ECOutlier, temporal outliers

KDD'12, August 12–16, 2012, Beijing, China.

1. INTRODUCTION

For a large number of applications, systems can be modeled as temporal datasets. Each snapshot in such a temporal dataset could describe the attributes of a collection of objects or a network of connected objects. For example, consider a database of employees in a company. Each employee can be associated with a large number of temporal attributes like salary, address, telephone number, designation, etc. As another example, consider co-authorship networks like DBLP where each author node is associated with temporal data and links. In daily life, various kinds of records like credit, personnel, financial, judicial, medical, etc. are all temporal. Given a snapshot of such a temporal dataset, analysts often perform clustering of objects with the goal of determining intrinsic grouping of objects in an unsupervised manner. We can refer to each group as a *community*. By analyzing a pair of snapshots from such a temporal dataset, we can observe that these communities evolve, often contracting, expanding, splitting or merging with each other. Most of the objects within a community follow similar evolution trends and their average defines the evolution trend of the community. However, evolutionary behavior of certain objects is quite different from the average evolutionary behavior of its community. Our goal is to detect such anomalous objects as *Evolutionary Community Outliers* (or *ECOutliers*) given a pair of snapshots from a temporal dataset.

***ECOutlier* Examples**

ECOutliers are interesting because they evolve against the trend, and often times they prove to be trend-setters. Interesting examples of *ECOutliers* can be commonly observed in real-life scenarios. We list a few below.

- A stockbroker who suddenly changes his portfolio and starts investing in another sector against his historical investments even when other similar stockbrokers continue to invest in the same old sector.
- Conglomerate diversification, e.g., "Walt Disney" moved from producing animated movies to construction of theme parks and vacation properties.

Case Studies

Now let us study two specific cases in detail.

Computer Science Research Evolution. An *ECOutlier* author may change his research area across time while others in his community continue to research in the same area. For example, consider the two snapshots 1997 and 1998 when Soumen Chakrabarti changed his research area from "Parallel Systems" (characterized by work in ICALP, PLDI, SPAA) to "Data and Information Systems" (characterized by work

in VLDB, SIGMOD). In a bibliographic network, a conference may change the sub-topics it deals with while other conferences in the area continue to focus on the same sub-topics. For example, in 2010, there were 13 papers in CIKM about personalization, while none of the other IR conferences like WWW, WSDM, SIGIR or ECIR focused so much on personalization in that year.

Employee Promotion. Suppose we have collected the communication frequencies among employees in a company. Also, each employee has temporal attributes like salary, designation, location, etc. We can identify the natural communities based on their attribute values and communication links. Across two snapshots of this small company, say only one employee E gets promoted from a developer two levels up to a team lead. Also, E's salary increases, and his communication pattern changes because he starts communicating more with managers. Due to change in links and the associated data, the community membership of E changes too. As E has behavior (community membership) changes much different from other members in his original community, he can be considered as an *ECOutlier*. E can be detected by identifying and comparing the communities of the two snapshots.

Existing Work

Here we briefly mention the difference between this work and existing outlier detection techniques. More discussions can be found in Section 2. (1) *Single-snapshot outlier detection*: Distance-based [22, 30] and density-based [7, 24] methods have been proposed but they work on only single snapshot data and hence cannot detect temporal changes. (2) *Temporal outlier detection*: Traditional time series literature [13] defines two types of outliers (Type I/additive and Type II/innovative) based on the data associated with an individual object across time, ignoring the community aspect completely. (3) *Stream outlier detection*: Recent work on outlier detection on data streams has focused on distance-based local outliers [29] or on graph outliers [4], while we focus on outliers in the community context. In general, existing work ignores time or community information in outlier detection, and thus the outliers detected traditionally are not evolutionary community outliers as proposed in this paper.

Integrated Framework

Evolutionary community outliers are objects that do not follow the evolutionary trend compared with other objects in the same community. Therefore, one natural way to identify such *ECOutliers* is to detect objects which change their community belongingness against the trend across time. However, community discovery is an unsupervised procedure, so communities discovered at different snapshots may not match with each other. Moreover, communities evolve too, and thus communities must be matched across snapshots. *ECOutliers* can be detected with high accuracy if the community matching across two snapshots is of high quality. However, community matching suffers from the presence of *ECOutliers* itself. Outliers are objects that defy the trend, and the trend must be obtained from community matching. Therefore, community matching and outlier detection cannot be separated. This motivates us to present an integrated framework which models outlier detection and community matching as a joint optimization problem. As we will show in Section 5, such an integrated approach is much more effective compared to a two-stage approach, which detects outliers after community matching is done.

Brief Overview of ECOutlier Detection

Consider a temporal dataset represented by the snapshot series $X_1, X_2, \ldots, X_T$. In this paper, we focus on the problem of detecting evolutionary outliers from any of the two snapshots X_i and X_j. Given community detection results on all pairs of snapshots, evolutionary outliers across multiple snapshots can be easily defined based on simple post-processing. For the sake of simplicity, let us denote the pair of snapshots as X_1 and X_2. Latent community discovery on X_1 (or X_2) leads to a matrix $\boldsymbol{P}$ (or $\boldsymbol{Q}$). Each element of $\boldsymbol{P}$ (or $\boldsymbol{Q}$) denotes the probability with which a particular object belongs to a particular community in snapshot X_1 (or X_2). Next, one needs to match a particular community in X_1 to one or more communities in X_2. Note that community mismatch happens due to permutation of community labels and community evolution. Such matching should try to minimize the distance between the matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$. To ignore the effect of *ECOutliers* from such an optimization, the distance function should appropriately discount the matrix entries ((object, community) pairs) corresponding to outliers, thereby also learning an outlierness score for every (object, community) pair. Hence, we formulate the problem as an optimization problem over both outlierness degree and community correspondence. Community matching and outlier detection are improved through iterative updating procedures, and upon convergence, meaningful outliers are output.

Summary

We make the following contributions in this paper:

- We introduce the notion of Evolutionary Community Outliers *ECOutliers*. To the best of our knowledge, this is the first work on detecting evolutionary outliers considering both time and community information.
- We formulate the problem using an integrated optimization framework and develop an iterative algorithm that performs community matching and evolutionary outlier detection simultaneously.
- We show the interesting and meaningful outliers detected from multiple real and synthetic datasets.

Our paper is organized as follows. We discuss related work in Section 2. In Section 3, we define the *ECOutlier* detection problem, and present the optimization framework and solution. In Section 4, we present analysis and discussions related to the algorithm. We discuss experimental setup and results with detailed insights in Section 5. The paper is summarized in Section 6.

2. RELATED WORK

Our work is related to the areas of community matching and outlier detection.

Community Matching: The problem of community matching appears mainly when multiple different clusterings need to be integrated into a single clustering. Community mismatch happens because of 'label switching' or 'genuine multimodularity' [20]. Community matching (consensus clustering) can be done in a hard or a soft way. Hard community matching can be performed by selecting the best matching pair of communities one by one, avoiding conflict with already selected pairs [10, 11] or using greedy algorithms like CLUMPP [20]. Soft community matching can be done so as to minimize the distance between the two matrices [27]. In computer vision, community matching has been

Notation	Meaning
o	Index for objects
i	Index for a community in X_1
j	Index for a community in X_2
N	Number of objects
K_1	Number of communities in X_1
K_2	Number of communities in X_2
$\boldsymbol{P}^{N\times K_1} = [p_{oi}]$	Belongingness matrix for snapshot X_1
$\boldsymbol{Q}^{N\times K_2} = [q_{oj}]$	Belongingness matrix for snapshot X_2
$\boldsymbol{S}^{K_1\times K_2} = [s_{ij}]$	Correspondence matrix
$\boldsymbol{A}^{N\times K_2} = [a_{oj}]$	Outlierness matrix
μ	Estimated sum of outlierness

Table 1: Table of Notations

done by using cluster features like position, intensity, shape and average gray-scale difference [23], and degree of match between surrounding clusters [28]. Words-based communities could be matched using TF-IDF similarity [9]. Different from these studies, we introduce a new soft community matching technique which is evolutionary-outlier-aware.

Outlier Detection: Outlier (or anomaly) detection is a very broad field and has been studied in the context of a large number of application domains. Chandola et al. [8] and Hodge et al. [18] provide extensive overview of outlier detection techniques. Four main types of outliers studied in literature are point outliers, contextual outliers, collective outliers and evolutionary outliers. A variety of supervised, semi-supervised and unsupervised techniques have been used for outlier detection. These include mixture of models [12], neural networks, stat profiling using histograms, SVMs [19], rule based systems, parametric stat modeling, non-parametric stat modeling, bayesian networks, etc. The problem studied in this paper has connections with distance-based outlier detection algorithms [21] in the sense that we are trying to search outliers in a space with community change trends as dimensions. Outliers have been discovered in high-dimensional data [2], uncertain data [3], stream data [4], network data [15] and time series data [13]. Recently, there has been significant interest in detecting outliers in evolving datasets [16, 17], but none of them explores the outliers with respect to communities in a general evolving dataset.

Trajectory outlier detection and community tracking methods [5, 26] (1) usually work well for *longer* time series (unlike two timestamps in our setting), (2) and need crisp correspondence among clusters across timestamps. Also, often such methods cannot model link dependencies between objects (e.g. in networks) in the same timestamp. Such techniques usually use Markov models, which could lead to expensive computation or approximate solutions; our method provides high accuracy in time linear in the number of objects.

3. ECOUTLIER DETECTION APPROACH

In this section, we will present our integrated framework for *ECOutlier* detection. Let us start by first introducing the notations. We represent a matrix using the notation $\boldsymbol{B}$. We use $\vec{b_{i\cdot}}$ and $\vec{b_{\cdot j}}$ to represent the i^{th} row and j^{th} column of $\boldsymbol{B}$ respectively. We will represent the $(i,j)^{th}$ element of the matrix $\boldsymbol{B}$ using b_{ij}. Given two vectors $\vec{a}$ and $\vec{b}$, we will use $\vec{a}\cdot\vec{b}$ to denote their dot product. Table 1 shows the important notations that we use in this paper. Next, we use these notations to define our problem.

3.1 Problem Definition

We start with an introduction to some basic concepts.

Community

A community is a probabilistic collection of similar objects, such that similarity between objects within the community is higher than the similarity between objects in different communities. For example, a research area is a community in a co-authorship network. We will use K_1 and K_2 to denote the number of communities in the two snapshots respectively.

Belongingness Matrix

Each entry in the belongingness matrix corresponds to the probability with which an object o belongs to a community i. The rows of the matrix correspond to objects while the columns correspond to communities. Let us denote the belongingness matrices for the N objects in X_1 and X_2 by $\boldsymbol{P}$ and $\boldsymbol{Q}$ respectively. Thus, $\boldsymbol{P} \in [0,1]^{N\times K_1}$, $\boldsymbol{Q} \in [0,1]^{N\times K_2}$, $\sum_{i=1}^{K_1} p_{oi} = 1$ and $\sum_{j=1}^{K_2} q_{oj} = 1$ for every object o.

Correspondence Matrix

We propose to use a soft matching of communities across snapshots. Soft correspondence means that a community of a given clustering corresponds to every community in another clustering with different weights. Hence, the match between two clusterings may be formulated as a matrix called as the correspondence matrix $\boldsymbol{S}^{K_1\times K_2}$. Also, $\sum_{j=1}^{K_2} s_{ij} = 1$ $(\forall i = 1 \ldots K_1)$.

Outlierness Matrix

We denote the outlierness matrix by $\boldsymbol{A}^{N\times K_2}$. a_{oj} represents the outlierness score for the (object, community) entry (o,j).

Evolutionary Community Outlier

An (object, community) pair (o,j) is an *ECOutlier* if change in p_{oi} to q_{oj} is quite different from the average change trend for community i in X_1 and j in X_2. An object can be considered as an outlier if the change in its probability distribution with respect to community belongingness is quite different from that of its X_1 community members.

Evolutionary Community Outlier Detection Problem

Given two snapshots ($\boldsymbol{P}$ and $\boldsymbol{Q}$), our problem is to estimate $\boldsymbol{S}$ and $\boldsymbol{A}$ and thereby derive *ECOutliers* with respect to the two snapshots.

To perform community matching between the matrices $\boldsymbol{P}$ and $\boldsymbol{Q}$, one needs to estimate a correspondence matrix $\boldsymbol{S}^{K_1\times K_2}$ such that the distance (sum of entrywise squared differences) between the matrices $\boldsymbol{Q}$ and $\boldsymbol{P}\times\boldsymbol{S}$ is minimized. However, such an approach will perform biased community matching if we take into account the contribution from outlier entries too, when estimating the correpondence matrix $\boldsymbol{S}$. For higher quality matching, one needs to ignore evolutionary outlier entries. Hence, we need to incorporate the outlierness score matrix $\boldsymbol{A}$ into community matching. In the remainder of this section, we will develop an integrated approach to compute $\boldsymbol{S}$ and $\boldsymbol{A}$.

3.2 Integrated Framework

Let μ be the estimated sum of outlierness in the snapshot. Then, we can incorporate the outlierness score into the community matching formulation as shown in Eqs. 1 to 5. We will discuss the estimation of μ in Section 3.5.

In the objective function (Eq. 1), $\boldsymbol{S}$ and $\boldsymbol{A}$ are the variables to be learned. $(q_{oj} - \vec{p_{o\cdot}}\cdot\vec{s_{\cdot j}})^2$ denotes the squared error incurred in community matching and $\log\left(\frac{1}{a_{oj}}\right)$ determines the outlier weight associated with the $(o,j)^{th}$ entry. Note that even if there were no outliers, there would still be some matching error, because each object evolves somewhat differently from the community averages. However,

outliers that evolve very differently from community averages are penalized using a higher a_{oj} value. Using $\log\left(\frac{1}{a_{oj}}\right)$ in the objective function allows us to smooth out outlierness values. The log function makes sure that the weights for individual entry matching across snapshots lie within a small range. The more anomalous a particular (object, community) entry (o, j) is, the higher will be the value of a_{oj} and so $\log\left(\frac{1}{a_{oj}}\right)$ will be lower. This would mean that lower weight will be associated with the $(o, j)^{th}$ outlier entry when performing community matching. While there could be other ways of formulating the objective function, we use this particular formulation for ease of computation.

$$\min_{\boldsymbol{S},\boldsymbol{A}} \sum_{o=1}^{N}\sum_{j=1}^{K_2} \log\left(\frac{1}{a_{oj}}\right)(q_{oj} - \vec{p_{o\cdot}} \cdot \vec{s_{\cdot j}})^2 \tag{1}$$

subject to the following conditions

$$s_{ij} \geq 0 \quad \forall i = 1 \ldots K_1, \forall j = 1 \ldots K_2 \tag{2}$$

$$\sum_{j=1}^{K_2} s_{ij} = 1 \quad \forall i = 1 \ldots K_1 \tag{3}$$

$$1 \geq a_{oj} \geq 0 \quad \forall o = 1 \ldots N, \forall j = 1 \ldots K_2 \tag{4}$$

$$\sum_{o=1}^{N}\sum_{j=1}^{K_2} a_{oj} \leq \mu \tag{5}$$

Total Amount of Outlierness

If the total amount of outlierness is unbounded, one can simply mark all entries as outliers and then there will be no useful community matching. This corresponds to the trivial solution of setting all $\boldsymbol{A}$ elements to very high values, for the optimization (Eq. 1). Hence, we need to put in a constraint based on how many outliers we expect. Note that normal entries have small a_{oj} values, while outlier entries have large a_{oj} values. Thus, we would like to bound the total sum of all a_{oj} values to be within a maximum level of outlierness we expect in the snapshot. This can be achieved using the constraint $\sum_{o=1}^{N}\sum_{j=1}^{K_2} a_{oj} \leq \mu$ (Eq. 5). We replace it by an equality constraint to simplify computation. In fact, the semantic meanings of outlierness scores will not change by this action because we only care about the relative ranking of the scores. Essentially the equality claims that there is a certain level of outlierness in the entire snapshot. We will show later how we can estimate μ.

The objective function can be minimized (local minimum) by alternately optimizing one of $\boldsymbol{S}$ and $\boldsymbol{A}$ while fixing the other. Next, we will derive iterative update rules for the correspondence entry (s_{ij}) and the outlierness scores (a_{oj}). Using the method of Lagrangian Multipliers, we can rewrite the problem as follows. Here, β_i and γ are Lagrangian variables.

$$\min_{\boldsymbol{S},\boldsymbol{A}} \quad f = \sum_{o=1}^{N}\sum_{j=1}^{K_2} \log\left(\frac{1}{a_{oj}}\right)(q_{oj} - \vec{p_{o\cdot}} \cdot \vec{s_{\cdot j}})^2$$
$$+ \sum_{i=1}^{K_1} \beta_i \left[\sum_{j=1}^{K_2} s_{ij} - 1\right] + \gamma\left[\sum_{o=1}^{N}\sum_{j=1}^{K_2} a_{oj} - \mu\right] \tag{6}$$

subject to the following conditions

$$s_{ij} \geq 0 \quad \forall i = 1 \ldots K_1, \forall j = 1 \ldots K_2 \tag{7}$$

$$1 \geq a_{oj} \geq 0 \quad \forall o = 1 \ldots N, \forall j = 1 \ldots K_2 \tag{8}$$

3.3 Derivation of Update Rules

Taking the partial derivative of Eq. 6 with respect to a particular a_{oj} and setting it to 0, we obtain the following.

$$a_{oj} = \frac{(q_{oj} - \vec{p_{o\cdot}} \cdot \vec{s_{\cdot j}})^2}{\gamma} \quad \textit{and} \quad \sum_{o=1}^{N}\sum_{j=1}^{K_2} \frac{(q_{oj} - \vec{p_{o\cdot}} \cdot \vec{s_{\cdot j}})^2}{\mu} = \gamma \tag{9}$$

This gives us the update rule for a_{oj} as follows.

$$a_{oj} = \frac{(q_{oj} - \vec{p_{o\cdot}} \cdot \vec{s_{\cdot j}})^2 \mu}{\sum_{o'=1}^{N}\sum_{j'=1}^{K_2}(q_{o'j'} - \vec{p_{o'\cdot}} \cdot \vec{s_{\cdot j'}})^2} \tag{10}$$

The numerator $(q_{oj} - \vec{p_{o\cdot}} \cdot \vec{s_{\cdot j}})^2$ represents the squared error for the $(o, j)^{th}$ entry, which represents the error incurred by matching the community detection results between two snapshots on the o^{th} object with respect to the j-th community in $\boldsymbol{Q}$. The denominator in Eq. 10 represents the overall error across all the entries in matrix $\boldsymbol{Q}$, given a particular $\boldsymbol{S}$, which serves as a normalization factor. Intuitively, we assign a higher outlierness score to objects and communities that incur higher community matching error, while objects that are matched well with respect to certain communities across two snapshots are considered normal and thus receive lower outlierness score.

Now, we will obtain the update rule for s_{ij}. Taking partial derivative of f with respect to s_{ij}, we obtain the following.

$$\sum_{o'=1}^{N}\left[2\log\left(\frac{1}{a_{o'j}}\right)(q_{o'j} - \vec{p_{o'\cdot}} \cdot \vec{s_{\cdot j}})(-p_{o'i})\right] + \beta_i = 0 \tag{11}$$

After some algebraic simplifications, we obtain the following update rule for s_{ij}.

$$s_{ij} = \frac{\sum_{o'=1}^{N} 2\log\left(\frac{1}{a_{o'j}}\right) p_{o'i}\left[q_{o'j} - \sum_{\substack{k=1 \\ k\neq i}}^{K_1} p_{o'k}s_{kj}\right] - \beta_i}{\sum_{o'=1}^{N} 2\log\left(\frac{1}{a_{o'j}}\right) p_{o'i}^2} \tag{12}$$

The intuition behind Eq. 12 is as follows. Our goal is to compute s_{ij} when other elements of the matrix $\boldsymbol{S}$ are fixed. s_{ij} involves matching of all objects with respect to the i^{th} column of $\boldsymbol{P}$ and the j^{th} column of $\boldsymbol{Q}$. Contributions from each object o' are weighted by $\log\left(\frac{1}{a_{o'j}}\right)$ such that highly outlying objects contribute little to matching. Fixing other elements of $\boldsymbol{S}$, $\left[q_{o'j} - \sum_{\substack{k=1 \\ k\neq i}}^{K_1} p_{o'k}s_{kj}\right]$ represents the part of $q_{o'j}$ that needs to be explained by $p_{o'i}s_{ij}$. β_i and the denominator of Eq. 12 are used to make sure that $\sum_j s_{ij} = 1$. β_i's can now be computed easily using Eq. 12 and the constraints $\sum_{j=1}^{K_2} s_{ij} = 1$ $(\forall i = 1 \ldots K_1)$. This value of β_i can then be substituted back in Eq. 12 to obtain the update rule for s_{ij}.

3.4 Interpretation of S and A

$\boldsymbol{S}$ captures the average evolution trend for the communities in the two snapshots. It also captures the permutation effect when matching two clusterings. s_{ij} represents the degree to which the community i in snapshot X_1 contributes to the community j in snapshot X_2. Thus, s_{ij} averages the evolution/match across all the objects belonging to i in X_1 and j in X_2. If a community i splits into two parts j_1 and j_2, s_{ij_1} and s_{ij_2} will have non-zero values. Similar to this split case, s_{ij} can be used to represent community merges, community expansion, community shrinking and a mix of such scenarios. Apart, a community i may die out with all s_{ij}'s set to 0. Similarly, $\boldsymbol{S}$ can also capture birth of new communities.

Now, if there are evolutionary outliers, they will possess values quite different from the average. For example, when

Algorithm 1 OneStageμ Outlier Detection Algorithm

Input: $\boldsymbol{P}$, $\boldsymbol{Q}$
Output: Estimates of $\boldsymbol{S}$ and $\boldsymbol{A}$
1: Initialize μ to 1
2: Initialize all $s_{ij} \Leftarrow \frac{1}{K_2}$ and all $a_{oj} \Leftarrow \frac{1}{NK_2}$.
3: **while** NOT converged **do**
4: Update $\boldsymbol{A}$ using Eq. 10 (Outlier Detection Step).
5: Update $\boldsymbol{S}$ using Eq. 12 (Community Matching Step).
6: **end while**
7: $\mu \Leftarrow \frac{\sum_{o'=1}^{N}\sum_{j'=1}^{K_2}(q_{o'j'} - \vec{p_{o'}} \cdot \vec{s_{\cdot j'}})^2}{\max_{o,j}(q_{oj}^2)}$
8: Repeat Steps 2 to 6.

s_{ij}=1, community i in X_1 gets merged with, or is renamed as community j in X_2. However, an outlier will have most of its mass moving from community i in X_1 to communities other than j in X_2. Presence of such outliers can affect the computation of s_{ij} itself because outliers can lead to significantly different average values. In our formulation, a weight is given to an object o when computing the community evolution values for community j, which is a function of a_{oj}. For normal (object, community), a_{oj} should be low, while for outlying (object, community), a_{oj} should be high. $\boldsymbol{A}$ is designed to capture such evolutionary outlierness.

3.5 Estimation of Overall Outlierness μ

Note that μ indicates the total sum of outlier scores (in Eq. 5). If we set μ to 1, a_{oj} is expected to be quite small for most of the (o, j) entries. There are N objects and K_2 communities, and thus the matching error of one entry is much smaller than the total squared sum of errors. The consequence is that the difference between a_{oj} is also small and it is hard to judge whether an object is an outlier or not based on a_{oj}. Techniques that transform and interpret outlierness scores [25] might help solve this problem. However, another problem is that a small μ causes overfitting of $\boldsymbol{S}$ to the outlier entries, when performing community matching. Hence, we would like to have μ as high as possible without violating constraints in the optimization. μ is upper-bounded by the combination of Eq. 10 and the constraint $a_{oj} \leq 1$. Therefore, we propose the following two-pass procedure to set μ. In the first pass, the overall snapshot outlierness μ is initialized to 1. After the first pass, μ is estimated as the ratio of overall error to the maximum entry value, as shown in Line 7 of Algorithm 1. The algorithm then uses this estimated μ for the second pass. Since μ increases compared to the first pass, a_{oj} values are relatively large. This reduces the overfitting of $\boldsymbol{S}$ to the outlier entries. Hence, matching for non-outlier entries improves, and so outlier detection improves too. Also, due to relaxation in matching, the overall error, i.e., the denominator in Eq. 10 is higher than the overall error in the first pass, and thus the value of a_{oj} remains ≤ 1, ensuring that the constraint is not violated.

Algorithm 1 summarizes the proposed *ECOutlier* detection algorithm. We name it *OneStage*μ to distinguish it from several baselines we evaluated, which will be discussed in detail in Section 5. "OneStage" indicates that the proposed method conducts outlier detection and community matching together, and μ is used to represent the two-pass procedure that estimates μ. As can be seen, the proposed method iteratively learns both parameters representing the outlierness scores (a_{oj}) and the community correspondence values (s_{ij}). At every iteration, a_{oj} values are first updated assuming all s_{ij} are fixed, and then s_{ij} values are sequentially updated based on the values of all a_{oj} and s_{ij} from other entries. The algorithm terminates when the change in the value of the objective function is less than a threshold ϵ.

4. ANALYSIS AND DISCUSSIONS

In this section, we analyze the convergence property and time complexity of the proposed *ECOutlier* detection method. We also discuss several important issues in implementing the method.

Convergence

LEMMA 4.1. *Algorithm 1 converges and the final solution is a stationary point of the optimization problem presented in Eqs. 1 to 5.*

PROOF. As stated in [6, p. 267-271], a block coordinate descent algorithm converges to a stationary point if the function achieves minimum at each step with respect to the set of working variables when the values of other variables are fixed. To prove this, we first check the convexity of the function. We notice that the constraints (Eqs. 2 to 5) are all linear. When we consider $\boldsymbol{S}$ as constant, the optimization function is a linear combination of negative log of a_{oj}'s. As $\log(\frac{1}{a_{oj}})$ is convex, its linear combination is also convex. When $\boldsymbol{A}$ is fixed and s_{ij}'s are updated sequentially, the objective function can be written as a quadratic function in s_{ij}. Since p_{ij} are all positive, it can be derived that the function is convex in each s_{ij}. As the function to be minimized is convex at each step, setting derivatives with respect to the working variables to zero will give unique minimum. Hence, Algorithm 1 is guaranteed to converge. □

Computational Complexity

Recall that N is the number of objects, K_1 and K_2 are the number of communities in the two snapshots respectively. Sum of all error values in Eq. 10 can be computed once for all a_{oj}'s in $O(NK_1K_2)$ time. Then, computation of each a_{oj} takes $O(K_1)$ time. $\boldsymbol{S}$ consists of K_1K_2 entries. When computing $\boldsymbol{S}$, at every iteration, for each s_{ij}, one needs to use Eq. 12 which is $O(NK_1)$ itself. Thus, computational complexity of the *OneStage*μ Algorithm is $O(NK_1^2K_2I)$ where I is the number of iterations. As can be seen, the running time is linear with respect to the number of objects. Usually the number of communities is small, and thus the proposed method scales well to large data sets.

Identifying Outliers

Algorithm 1 allows us to obtain outlierness scores for every (object, community) pair, given two snapshots of a temporal dataset. These scores are useful to identify interesting objects that are outliers with respect to a particular community. One could also aggregate such outlierness scores in a variety of ways to obtain the outlierness score for an object across all communities. Different aggregation functions could be used; weighted sum of scores or maximum (object, community) entry corresponding to an object may be a good choice, depending on the application. Also, when reporting top outlier objects in the snapshot one may want to consider the overall activity of the object along with its outlierness score. For example, an author publishing 50 papers in a snapshot with a relatively lower level of outlierness may still be more interesting than an author publishing only 5 papers but with a relatively higher level of outlierness.

Initialization

We initialize all a_{oj} values to $\frac{1}{NK_2}$, i.e., we begin by considering all entries to be equally anomalous. We initialize

all s_{ij} entries to $\frac{1}{K_2}$, i.e., we assume that each community in snapshot X_2 evolves equally from each of the communities in snapshot X_1.

Series of Snapshots

In this paper, we focused on finding *ECOutliers* given a pair of snapshots. However, it is quite natural to extend our approach given a series of snapshots. We can perform community matching and outlier detection across every pair of consecutive snapshots, using the proposed module. Also, when the evolution is gentle, one can smooth out community matching over several consecutive snapshots. This can help in improving the community matching accuracy, thereby also improving the outlier detection.

5. EXPERIMENTS

Evaluation of outlier detection algorithms is quite difficult due to lack of ground truth. We perform experiments on multiple synthetic datasets, each of which simulates real scenarios. We will evaluate outlier detection accuracy of the proposed algorithm based on outliers injected in synthetic datasets. We evaluate the results on real datasets using case studies. We perform comprehensive analysis of objects to justify the top few outliers returned by the proposed algorithm. The code and the data sets are available at: http://blitzprecision.cs.uiuc.edu/ECOutlier

5.1 Baselines

We compare the proposed algorithm with three baseline methods: *OneStage (1S)*, *TwoStage (2S)* and *NearestNeighbor (NN)*. As discussed, our method is named as *OneStageμ (1Sμ)* because it integrates outlier detection and community matching, and μ is estimated using a two-pass procedure. The baseline methods are explained as follows.

OneStage (1S): *1S* is the one pass version of *1Sμ* (Steps 1 to 6 of Algorithm 1), in which total outlier score μ is set to 1. Thus, comparison with *1S* will help us understand improvement in accuracy by improved estimation of μ.

TwoStage (2S): Outliers are obtained by looking at the top values in $\boldsymbol{Q} - \boldsymbol{P} \times \boldsymbol{S}$ where $\boldsymbol{S}$ is computed by matching $\boldsymbol{P}$ and $\boldsymbol{Q}$. Comparison with *2S* will help us understand which method is better – performing outlier detection after community matching (*2S*) or doing them in an integrated way (*1Sμ*).

NearestNeighbor (NN): For every object o, we find its k-Nearest Neighbors set, $NN_{X_1}(o)$, in $\boldsymbol{P}$ using the KDTree implementation in Java-ML [1]. Recall that $\boldsymbol{P}$ and $\boldsymbol{Q}$ are the belongingness matrices for snapshots X_1 and X_2. We exclude the object o from $NN_{X_1}(o)$. The outlierness score for (object, community) pairs (o, j) can then be computed as $a_{oj} = \left| q_{oj} - \frac{\sum_{o' \in NN_{X_1}(o)} q_{o'j}}{|NN_{X_1}(o)|} \right|$. Note that this means that an outlier entry (o, j) has a high score if $q_{o,j}$ (i.e., belongingness of object o to community j in the second snapshot) is quite different from the average belongingness of its X_1 nearest neighbors to the same community j in X_2. *NN* seems to follow the exact definition of *ECOutliers* and so one would expect it to get high accuracy but we will discuss later on why it fails to perform better than other methods.

5.2 Synthetic Datasets

Dataset Generation

We generate a variety of synthetic datasets to capture different spatial cases of evolution (Figs. 1 and 2). For each dataset, the figures show two snapshots. In each snapshot, we generate multiple clusters, each of which represents a community. Each cluster is modeled using a 2D Gaussian distribution, and evolution is modeled by changing the means and the variances of the Gaussians. For example, to model contraction (expansion) of a cluster, we reduce (increase) the standard deviation of the corresponding cluster (the first two plots in Fig. 1). *NoEvolution* is represented by Plots 3 and 4 in Fig. 1. Plots 5 and 6 in Fig. 1 show the case of *Merge* where top four clusters merge to form one big cluster, and *Split* is shown in the last two plots in Fig. 1 where the upper cluster splits into four small clusters. We model a mix of cluster evolution in Fig. 2, which consists of cluster expansion, contraction, merging and splitting. Thus, we try to incorporate all forms of possible cluster evolution in our synthetic datasets. The circles represent 2σ boundary for the cluster Gaussian. For each cluster, we generate a set of N/K_1 points, each of which is sampled from the cluster Gaussian. For each of the N points, we obtain p_{oj} and q_{oj} as the probability with which the point can be generated from its cluster's Gaussian distribution. Using $\boldsymbol{P}$ and $\boldsymbol{Q}$, we obtain $\boldsymbol{S}$ as the community matching matrix in absence of any outlierness. $\boldsymbol{S}$ captures evolution without the effect of outliers.

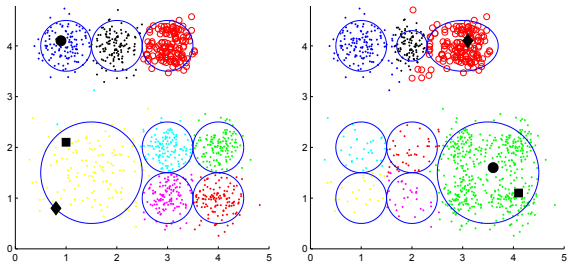

Figure 2: *SynMix* Dataset (X_1 and X_2)

Next, we inject outliers as follows. First we set an outlierness factor Ψ and choose a random set of objects, R with $N \times \Psi$ objects. For each object o in R, we swap the $q_{oj_{min}}$ and $q_{oj_{max}}$ values where j_{min} and j_{max} are the communities with the min and the max q_{oj} values. Thus, we inject two outliers per object in R. Finally, we set $a_{oj} = (q_{oj} - \vec{p_{o\cdot}} \cdot \vec{s_{\cdot j}})^2$. For each of the datasets, for X_2, we show the top three outliers (using different black filled shapes) discovered by *OneStageμ*. We show their original positions in X_1 and their new positions in X_2.

Results on Synthetic Datasets

We experiment using a variety of different settings. For each setting, we perform 100 experiments and report the mean values. We fix the threshold ϵ for convergence to 10^{-6}. We vary the number of objects as 1000, 5000 and 10000. We vary the percentage of outliers injected into the dataset as 1%, 2%, 5% and 10%. Using these settings, we compare the actual outlier objects with the top outliers returned by various algorithms. For each algorithm, we show the precision of outlier detection with respect to the ranking of outliers. The results of the three baselines and the proposed method are shown in Fig. 3 for *SynMix* dataset (10000 objects, Ψ=10%). Note that the proposed algorithm (*1Sμ*) outperforms the others in finding the top few outliers most precisely. The area under this curve (AUC) is a good measure of the effectiveness of the algorithm in identifying the outliers. We report the AUC values in Table 2 (Average variances are .0012 for *NN*, .0021 for *2S*, .0017 for *1S*, and .0005

N	Ψ (%)	SynContractExpand				SynNoEvolution				SynMerge				SynSplit				SynMix			
		NN	2S	1S	1Sμ	NN	2S	1S	1Sμ	NN	2S	1S	1Sμ	NN	2S	1S	1Sμ	NN	2S	1S	1Sμ
1000	1	.755	.947	**.966**	**.966**	.832	.791	.853	**.965**	.720	.774	.835	**.926**	.786	.918	.929	**.931**	.606	.891	.904	**.925**
	2	.729	.920	.948	**.957**	.812	.733	.789	**.961**	.702	.715	.781	**.908**	.779	.865	.920	**.924**	.675	.823	.860	**.915**
	5	.710	.853	.913	**.956**	.726	.712	.752	**.928**	.645	.654	.719	**.849**	.697	.799	.891	**.920**	.631	.770	.817	**.920**
	10	.619	.766	.833	**.960**	.657	.684	.706	**.881**	.580	.617	.656	**.801**	.630	.749	.832	**.918**	.594	.730	.776	**.917**
5000	1	.778	.945	**.970**	**.970**	.938	.793	.848	**.971**	.713	.762	.801	**.928**	.796	.913	**.942**	**.942**	.691	.881	.895	**.918**
	2	.756	.930	.947	**.961**	.864	.772	.815	**.962**	.677	.752	.791	**.903**	.768	.885	.938	**.940**	.646	.862	.876	**.919**
	5	.689	.901	.929	**.964**	.742	.750	.779	**.941**	.626	.698	.749	**.827**	.689	.806	.913	**.924**	.608	.831	.860	**.921**
	10	.622	.778	.829	**.964**	.656	.730	.747	**.912**	.579	.643	.679	**.795**	.624	.762	.834	**.929**	.593	.783	.824	**.919**
10000	1	.769	.949	.973	**.974**	.926	.807	.856	**.974**	.707	.788	.817	**.933**	.789	.938	.955	**.960**	.665	.882	.897	**.921**
	2	.752	.937	.949	**.963**	.851	.788	.828	**.964**	.681	.762	.796	**.898**	.758	.898	.948	**.951**	.670	.869	.881	**.916**
	5	.695	.900	.930	**.964**	.738	.763	.788	**.951**	.627	.719	.756	**.826**	.683	.807	.914	**.922**	.604	.847	.871	**.919**
	10	.622	.771	.825	**.965**	.660	.753	.769	**.926**	.583	.645	.681	**.795**	.621	.769	.827	**.934**	.584	.812	.845	**.917**

Table 2: Syn Dataset AUC (*NN=NearestNeighbor, 2S=TwoStage, 1S=OneStage, 1Sμ=OneStageμ*)

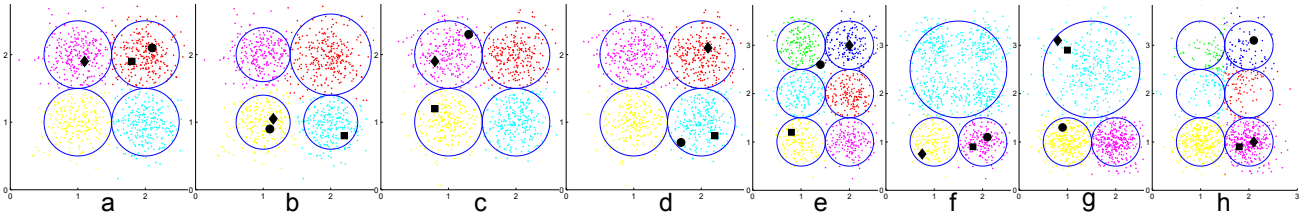

Figure 1: Two Snapshots (X_1 & X_2) each for *SynContractExpand* (a→b), *SynNoEvolution* (c→d), *SynMerge* (e→f), *SynSplit* Datasets (g→h)

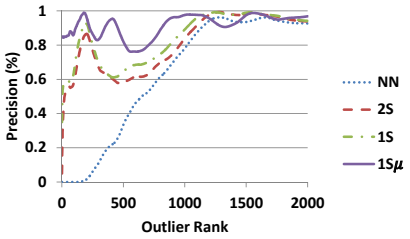

Figure 3: Precision for SynMix (*NN=NearestNeighbor, 2S=TwoStage, 1S=OneStage, 1Sμ=OneStageμ*)

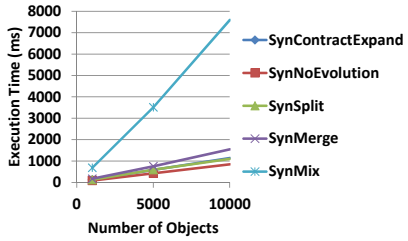

Figure 4: Running Times (ms) for *OneStageμ* (Scalability with Increasing Dataset Size)

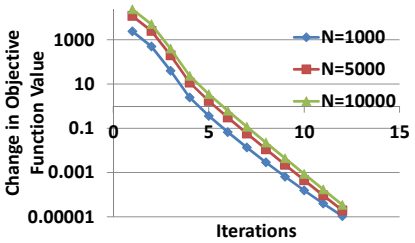

Figure 5: Convergence of OneStageμ Algorithm

for *1Sμ*). For *NearestNeighbors*, we tried k=5, 10, 50 and 100. We found the best AUC at k=10 and hence report the values for k=10. As the table shows, the proposed algorithm outperforms all the other algorithms for all the settings by a wide margin (sometimes as high as 30% better than the *TwoStage* method). *NearestNeighbors* performs fairly well for *SynNoEvolution* but not for other datasets. This is because it does not consider evolution and assumes that the set of nearest neighbors does not change across the snapshots. In contrast, the proposed algorithm detects outliers in the context of community evolution.

5.3 Real Datasets

Dataset Generation

We perform experiments using three real datasets: *IMDB*, *DBLP* and *Four Area* (subset of *DBLP*). We use *iTopicModel* [31] to perform community detection on the dataset since it uses both data and link information. It outputs the belongingness matrix $\boldsymbol{P}^{N\times K_1}$ ($\boldsymbol{Q}^{N\times K_2}$) for the snapshot X_1 (X_2).

***IMDB*:** We consider the co-starring graph (edge weight = co-starring frequency) created using actors who acted in at least 5 movies per snapshot. The two snapshots correspond to the years '06-'07 (15337 actors) and '08-'09 (13142 actors). Sets of genres of movies in which the actor acted constitutes the data associated with an actor. After community detection, we retain only those actors (6609) that are present in both snapshots. We consider number of communities as $K_1 = K_2 = 5$.

***DBLP*:** We consider the co-authorship graph (edge weight = co-authorship frequency) created using publications in the top 1500 conferences by authors, each of which published ≥ 10 papers per snapshot. The two snapshots correspond to the years '06-'07 (4784 authors) and '08-'09 (5633 authors). Sets of related conferences constitute the data associated with an author. After community detection, we retain only those authors (2666) that are present in both snapshots. We consider number of communities as $K_1 = K_2 = 5$. Similar to the author network, we also create a network of conferences (edge weight = #authors publishing at both conferences) where the words in paper titles constitute the data associated with each conference. 920 conferences are present in both snapshots and again, we set $K_1 = K_2 = 5$.

***Four Area*:** This is a subset of DBLP for the four areas of data mining (DM), databases (DB), information retrieval (IR) and machine learning (ML). It consists of papers from 20 conferences (5 per area). For details, read [14]. We obtain two snapshots corresponding to the years '01-'04 (601 authors) and '05-'08 (1206 authors). We build the co-authorship network similar to the one for *DBLP*. $K_1 = K_2 = 4$, and 374 authors are present in both snapshots.

Results on Real Datasets

Here, we discuss case studies obtained from these datasets.

Conference	Sim_1	Sim_2	Sim_3	Sim_4
ANTS	0.55	0.16	0.14	0.08
TLCA	0.60	0.53	0.14	0.08
INFOS	0.83	0.75	0.10	0.22
Sigite Conf.	0.66	0.33	0.11	0.27
Gil Jahrestagung	0.77	0.81	0.08	0.31

Table 3: Analysis of Top Outlier Conferences (*1S*μ)

***IMDB*:** Top two outlier actors returned by *OneStage*μ are discussed below.

1. Kelly Carlson (I): In X_1, she did many Sport, Thriller and Action movies, while in X_2 she switched to Drama, Music, Reality-TV. Most of her top ten closest collaborators in X_1 still do Documentary, Thriller, Action in X_2. Few of her co-stars collaborate with her in the second time snapshot. Also, her X_1 neighbors used to do Sport, Comedy, Documentary, Action while her X_2 neighbors do Drama, Documentary, Thriller, Music. Thus, clearly she changed her community from Sports, Thriller, Action genres to Drama, Music genres.

2. Josh Brolin: In X_1, he did a lot of Thriller, Drama, Crime and Mystery movies. In X_2, he acted in a lot of Documentary, Comedy, History, Music movies. Not only did he change his genres completely, but also not many other actors show such a change in the type of their movies. This is why, in X_2, his genres are quite different from that of his X_1 nearest neighbors. Hence, clearly he is an outlier.

***DBLP* (Authors Network):** We will discuss about the top two authors which are detected as evolutionary community outliers.

1. Georgios B. Giannakis. In X_1, his publications were mainly in CISS, ICC, GLOBECOM, INFOCOM. In X_2, he published in completely different conferences: ICASSP, ICRA. We looked at the conferences at which his X_1 community members published in X_2. This set of conferences (GLOBECOM, ICC, CISS, INFOCOM) is completely different from the set of conferences at which he published in X_2, but much similar to his own published venues in X_1.

2. Vassilios Peristeras. In X_1, his publications were mainly in HICSS, ICEGOV, IEEE SCC, EDOCW, CSREA, etc. In X_2, he published in completely different conferences: WSKS, ICSC, OTM, ICDIM, SAC. We looked at the conferences at which his X_1 community members published in X_2. This set of conferences (HICSS, ISI, ICEGOV, etc.) is completely different from the set of conferences at which he published in X_2, but much similar to his own conferences in X_1. This clearly justifies him to be a community outlier.

***DBLP* (Conf Network):** Table 3 shows the top five conferences returned as outliers by *OneStage*μ. We performed some analysis and hence list four measures in Table 3: similarity between top 20 words (we removed most frequent 100 words from dataset) for the conference across the snapshots, similarity in top 20 words between the conference at X_2 and its ten closest X_1 community members, similarity in neighbor conferences across the two snapshots, similarity in the words shared by the neighbors in the first snapshot and neighbors in the second snapshot.

As the table shows, each of the conferences have very low similarity for at least one of the measures, justifying their detection as *ECOutliers*. For comparison with the baseline, we present the top five conferences returned as outliers returned by *TwoStage* in Table 4. As one can clearly see, the similarity values in Table 4 are much higher compared to Table 3. Thus, the outliers returned by *TwoStage* are not as good as the outliers returned by the proposed algorithm.

Conference	Sim_1	Sim_2	Sim_3	Sim_4
ESEC SIGSOFT FSE	0.26	0.71	0.72	0.70
ISORC	0.64	0.66	0.75	0.74
Communications in Computing	0.16	0.57	0.58	0.70
ICDE Workshops	0.27	0.65	0.74	0.78
HICSS	0.78	0.60	0.83	0.79

Table 4: Analysis of Top Outlier Conferences (*2S*)

***Four Area* (Authors Network):** In this dataset, we observe a trend of people moving from ML community to DM community. Also, many authors often publish in both DB and IR. For this dataset, we will discuss two outliers returned by *OneStage*μ, who behave quite different from these trends.

1. Jérôme Lang. For this author, most of his community members moved from logical reasoning and related areas to other areas like DM and IR but he stays in AI field. He published 5 and 11 papers in the two snapshots respectively. Words in the titles of his papers are mainly "logic, planning, representation, action, uncertainty, propositional". We looked at the words which his X_1 community members (other authors having similar word distributions) use in X_2. The top words were "data, retrieval, xml, web, learning, mining". This clearly shows that his community members moved from logical reasoning to other areas while Jérôme decided to stay in the area, opposed to the trend. While he continued to publish in pure ML and AI conferences, his community members publish in a lot of IR and DM conferences in X_2.

2. Georg Gottlob. Generally the observed trend is that ML authors move to other related areas like DM and IR. Sometimes, some DM authors publish in ML conferences. But Georg has been a DB author in X_1, who started publishing in ML community in X_2. In X_1, he published frequently in PODS, VLDB, ICDE. In X_2, apart from PODS, he published heavily in IJCAI and AAAI. His set of collaborators also changed by a large extent across the two snapshots. A lot of his X_2 collaborators publish in ML conferences.

5.4 Running Time and Convergence

The experiments were run on a Linux machine with 4 Intel Xeon CPUs with 2.67GHz each. The code was implemented in Java. Fig. 4 shows the execution time for *OneStage*μ on different synthetic datasets in ms. Note that the algorithm is linear in the number of objects. These times are averaged across 100 runs of the algorithm. On an average *OneStage*μ needed $\sim$13 iterations per pass to converge on both real and synthetic datasets. Fig. 5 shows the change in the objective function value with iterations for the *SynMix* dataset for different number of objects, using a log-linear plot. The figure shows that *OneStage*μ converges fairly quickly.

6. CONCLUSIONS

We introduced the notion of evolutionary outliers with respect to latent evolving communities, i.e., *ECOutliers*. Such outliers represent the objects which disobey the common evolutionary trend among the majority of the objects in a community. The challenge is that both community evolution patterns and outliers are unknown. Outliers should be derived based on community matching across different snapshots, but need to be ignored when conducting community matching. We proposed an optimization framework which integrates community matching and outlier detection. The objective function is to minimize community matching error, in which the contributions from outlier objects are weighed lower. An iterative algorithm *OneStage*μ is developed to

solve the optimization problem, which improves community matching and *ECOutlier* detection gradually. Experiments on a series of synthetic data show the proposed algorithm's capability of detecting outliers under various types of community evolution. Case studies on *DBLP*, *IMDB* and *Four Area* datasets reveal some interesting and meaningful evolutionary outliers. Although the proposed algorithm focuses on two snapshots, it can detect both short-term and long-term trends and outliers, as snapshots can consist of short or long intervals. Moreover, it can be extended to handle multiple snapshots.

7. ACKNOWLEDGEMENTS

We thank Vinod Vydiswaran and anonymous KDD reviewers for discussions and insightful comments. Research was sponsored in part by the Cyber Security project (W911NF-11-2-0086) of U.S. Army Research Laboratory, NSF IIS-0905215, and U.S. Air Force Office of Scientific Research MURI award FA9550-08-1-0265. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

8. REFERENCES

[1] T. Abeel, Y. Van de Peer, and Y. Saeys. Java-ML: A Machine Learning Library. *Journal of Machine Learning Research*, 10:931–934, Jun 2009.

[2] C. C. Aggarwal and P. S. Yu. Outlier Detection for High Dimensional Data. *SIGMOD Records*, 30:37–46, May 2001.

[3] C. C. Aggarwal and P. S. Yu. Outlier Detection with Uncertain Data. In *Proc. of the SIAM Intl. Conf. on Data Mining (SDM)*, 483–493, 2008.

[4] C. C. Aggarwal, Y. Zhao, and P. S. Yu. Outlier Detection in Graph Streams. In *Proc. of the* 27^{th} *Intl. Conf. on Data Engineering (ICDE)*, 399–409. 2011.

[5] J. Alon, S. Sclaroff, G. Kollios, and V. Pavlovic. Discovering Clusters in Motion Time-Series Data. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, Volume 1, 375–381. IEEE Computer Society, 2003.

[6] D. P. Bertsekas. *Non-Linear Programming (2nd Edition)*. Athena Scientific, 1999.

[7] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. LOF: Identifying Density-Based Local Outliers. In *Proc. of the 2000 ACM SIGMOD Intl. Conf. on Management of Data (SIGMOD)*, 93–104. ACM, 2000.

[8] V. Chandola, A. Banerjee, and V. Kumar. Anomaly Detection: A Survey. *ACM Surveys*, 41(3), 2009.

[9] W. W. Cohen and J. Richman. Learning to Match and Cluster Large High-Dimensional Data Sets for Data Integration. In *Proc. of the* 8^{th} *ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, 475–480. ACM, 2002.

[10] E. Dimitriadou, A. Weingessel, and K. Hornik. Voting-Merging: An Ensemble Method for Clustering. In *Proc. of the Intl. Conf. on Artificial Neural Networks (ICANN)*, 217–224. Springer, 2001.

[11] S. Dudoit and J. Fridlyand. Bagging to Improve the Accuracy of a Clustering Procedure. *Bioinformatics*, 19(9):1090–1099, 2003.

[12] E. Eskin. Anomaly Detection over Noisy Data using Learned Probability Distributions. In *Proc. of the* 17*th Intl. Conf. on Machine Learning (ICML)*, 255–262. Morgan Kaufmann Publishers Inc., 2000.

[13] A. J. Fox. Outliers in Time Series. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(3):350–363, 1972.

[14] J. Gao, F. Liang, W. Fan, Y. Sun, and J. Han. Graph-based Consensus Maximization among Multiple Supervised and Unsupervised Models. In *Proc. of the* 23^{rd} *Annual Conf. on Neural Information Processing Systems (NIPS)*, 585–593. Curran Associates, Inc., 2009.

[15] J. Gao, F. Liang, W. Fan, C. Wang, Y. Sun, and J. Han. On Community Outliers and their Efficient Detection in Information Networks. In *Proc. of the* 16^{th} *ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining (SIGKDD)*, 813–822, 2010.

[16] Y. Ge, H. Xiong, Z. Zhou, H. Ozdemir, J. Yu, and K. C. Lee. Top-Eye: Top-K Evolving Trajectory Outlier Detection. In *Proc. of the* 19^{th} *ACM Conf. on Information and Knowledge Management (CIKM)*, 1733–1736, 2010.

[17] A. Ghoting, M. E. Otey, and S. Parthasarathy. LOADED: Link-Based Outlier and Anomaly Detection in Evolving Data Sets. In *Proc. of the* 4^{th} *IEEE Intl. Conf. on Data Mining (ICDM)*, 387–390, 2004.

[18] V. J. Hodge and J. Austin. A Survey of Outlier Detection Methodologies. *AI Review*, 22(2):85–126, 2004.

[19] W. Hu, Y. Liao, and V. R. Vemuri. Robust Anomaly Detection Using Support Vector Machines. In *Proc. of the Intl. Conf. on Machine Learning (ICML)*, 282–289. Morgan Kaufmann Publishers Inc, 2003.

[20] M. Jakobsson and N. A. Rosenberg. CLUMPP: A Cluster Matching and Permutation Program for Dealing with Label Switching and Multimodality in Analysis of Population Structure. *Bioinformatics*, 23:1801–1806, Jul 2007.

[21] E. M. Knorr and R. T. Ng. Algorithms for Mining Distance-Based Outliers in Large Datasets. In *Proc. of the* 24^{th} *Intl. Conf. on Very Large Data Bases (VLDB)*, 392–403. Morgan Kaufmann, 1998.

[22] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-Based Outliers: Algorithms and Applications. *The VLDB Journal*, 8:237–253, Feb 2000.

[23] D. Kottke and Y. Sun. Motion Estimation Via Cluster Matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16:1128–1132, 1994.

[24] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. LoOP: Local Outlier Probabilities. In *Proc. of the* 18^{th} *ACM Conf. on Information and Knowledge Management (CIKM)*, 1649–1652. ACM, 2009.

[25] H.-P. Kriegel, P. Kröger, E. Schubert, and A. Zimek. Interpreting and Unifying Outlier Scores. In *Proc. of the* 11^{th} *SIAM Intl. Conf. on Data Mining (SDM)*, 13–24. SIAM / Omnipress, 2011.

[26] J.-G. Lee, J. Han, and X. Li. Trajectory Outlier Detection: A Partition-and-Detect Framework. In *Proc. of the* 24^{th} *Intl. Conf. on Data Engineering (ICDE)*, 140–149. IEEE Computer Society, 2008.

[27] B. Long, Z. M. Zhang, and P. S. Yu. Combining Multiple Clusterings by Soft Correspondence. In *Proc. of the* 5^{th} *IEEE Intl. Conf. on Data Mining (ICDM)*, 282–289. IEEE Computer Society, 2005.

[28] M. J. Miller, A. D. Olson, and S. S. Thorgeirsson. Computer Analysis of Two-Dimensional Gels: Automatic Matching. *ElectroPhoresis*, 5(5):297–303, 1984.

[29] D. Pokrajac, A. Lazarevic, and L. J. Latecki. Incremental Local Outlier Detection for Data Streams. In *IEEE Symposium on Computational Intelligence and Data Mining (CIDM)*, 504–515. IEEE, Apr 2007.

[30] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient Algorithms for Mining Outliers from Large Data Sets. *SIGMOD Records*, 29:427–438, May 2000.

[31] Y. Sun, J. Han, J. Gao, and Y. Yu. iTopicModel: Information Network-Integrated Topic Modeling. In *Proc. of the* 9^{th} *IEEE Intl. Conf. on Data Mining (ICDM)*, 493–502. IEEE Computer Society, 2009.