Beyond outliers and on to micro-clusters: Vision-guided Anomaly Detection

Wenjie Feng[1,2], Shenghua Liu[1,2], Christos Faloutsos[3], Bryan Hooi[3], Huawei Shen[1,2], Xueqi Cheng[1,2]

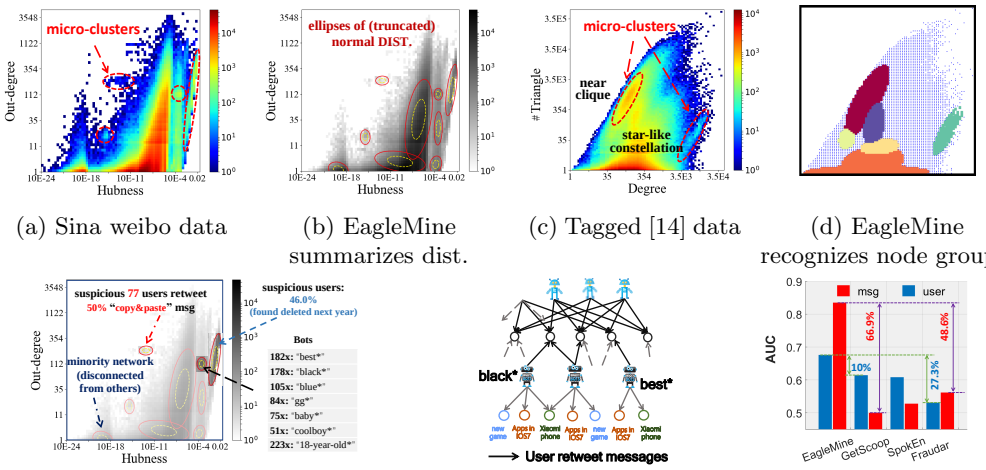
[1] CAS Key Laboratory of Network Data Science & Technology, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
[2] University of Chinese Academy of Sciences, Beijing 100049, China
[3] School of Computer Science, Carnegie Mellon University, PA, USA
fengwenjie@ict.ac.cn, liu.shengh@gmail.com, christos@cs.cmu.edu, bhooi@andrew.cmu.edu, {shenhuawei,cxq}@ict.ac.cn

Abstract. Given a heatmap for millions of points, what patterns exist in the distributions of point characteristics, and how can we detect them and separate anomalies in a way similar to human vision? In this paper, we propose a vision-guided algorithm, EagleMine, to recognize and summarize point groups in the feature spaces. EagleMine utilizes a water-level tree to capture group structures according to vision-based intuition at multiple resolutions, and adopts statistical hypothesis tests to determine the optimal groups along the tree. Moreover, EagleMine can identify anomalous micro-clusters (i.e., micro-size groups), which exhibit very similar behavior but deviate away from the majority. Extensive experiments are conducted for large graph scenario, and show that our method can recognize intuitive node groups as human vision does, and achieves the best performance in summarization compared to baselines. In terms of anomaly detection, EagleMine also outperforms state-of-the-art graph-based methods by significantly improving accuracy in synthetic and microblog datasets.

1 Introduction

Given real-world graphs with millions of nodes and connections, the most intuitive way to explore the graphs is to construct a correlation plot [25] based on the features of graph nodes. Usually a heatmap of those scatter points is used to depict their density, which is a two-dimensional histogram [20]. In the histogram, people can visually recognize nodes gathering into disjointed dense areas separately as groups (see Fig. 1), which help to explore patterns (like communities, co-author association behaviors) and detect anomalies (e.g., fraudsters, attackers, fake-reviews, outlier etc.) in an interpretable way [22].

In particular, a graph can represent friendships in Facebook, ratings from users to items in Amazon, or retweets from users to messages in Twitter, even they are time-evolving. Numerous correlated features can be extracted from graph, like degree, triangles, spectral vectors, and PageRank etc. and combination of these generate correlation plots. It becomes, even, labor-intensive to manually monitor and recognize patterns from heatmaps of the snapshots of temporal graphs. So, this raises the following questions: ***Given*** *a heatmap (i.e., histogram) of the scatter points in some feature space, how can we design an algorithm to automatically* ***recognize*** *and* ***monitor*** *the point groups as*



(a) Sina weibo data (b) EagleMine summarizes dist. (c) Tagged [14] data (d) EagleMine recognizes node groups.

(e) Micro-clusters highlighted and suspicious patterns. (f) Graphical view of anomaly Jellyfish pattern. (g) AUC for suspicious users and msgs.

Fig. 1: Heatmaps of correlation plots for some feature spaces of two real datasets and the performance of **EagleMine** algorithm. The bottom figures focus on the Sina weibo data. (a). Out-degree vs. Hubness feature space for weibo. (b). EagleMine summarizes the distribution of graph nodes for (a) with truncated Gaussian distributions. The ellipses denote the 1.5 and 3 times covariance of corresponding Gaussian. (c). # Triangle vs. Degree feature space for Tagged. (d). depicts the recognized node groups for (c). (e). highlights some micro-clusters in Fig.1b, including a disconnected small network, and very suspicious ones. A username list on the right side shows the name patterns of bots in a micro-cluster, where 182x: "best*" means 182 bots share prefix "best". (f). The structure of identified anomalous Jellyfish patterns. (g). shows the AUC performance for detecting suspicious users and msgs compared with state-of-the-art competitors.

human vision does, ***summarize*** *the points distribution in the feature space and* ***identify*** *suspicious micro-clusters?*

'Micro-cluster' refers to relatively small group of points (like users, items) with similar behavior in the feature space. Here we demonstrate some possible feature spaces, namely

i out-degree vs hubness - Fig. 1a - this can spot nodes with high out-degree, but low hubness score (i.e., fraudsters, which have many outgoing edges to non-important nodes, probably, customers, that paid them) [24].

ii #triangle vs degree - spotting a near-clique group (too many triangles, for their degree), as well as star-like constellations (too few triangles for such high degree) [23].

In this paper, we propose EagleMine, a novel tree-based mining approach to recognize and summarize the point groups in the heatmap of scatter plots, and can also identify anomalous micro-clusters. Experiments show that EagleMine outperforms baselines and achieves better performance both in quantitative (i.e., the code length for compact model description) and qualitative (i.e., consistent with vision-based judgment) comparisons, detects a micro-cluster of hundreds of bots in microblog data, Sina weibo[4], which presents

[4] One of the largest microblog websites in China.

Table 1: Comparison between algorithms.

	Density-based clustering	SpokEn	GetScoop	Fraudar	EagleMine
Micro-cluster detection	✓	✓	✓		✓
Micro-cluster suspiciousness			✓		✓
Linear scalability	?			✓	✓

strong signs of sharing unusual login-name prefixes, e.g., 'best*', 'black*' and '18-year-old*', and exhibiting very similar behavior in the feature space (see Fig. 1e).

In summary, the proposed EagleMine has the following advantages:

- **Anomaly detection:** can spot and explain anomalies on real data by identifying suspicious micro-clusters. Compared with the graph-based anomaly detection methods, EagleMine achieves higher accuracy for finding suspiciousness in Sina weibo.
- **Automated summarization:** automatically summarizes a histogram plot derived from correlated graph features (see Fig. 1b), and recognizes node groups forming disjonted dense areas as human vision does (see Fig. 3e).
- **Effectiveness:** detects interpretable groups, and outperforms the baselines and even those with *manually tuned parameters* in qualitative experiments (see Fig. 3).
- **Scalability:** is scalable with nearly linear time complexity in the number of graph nodes, and can deal with more correlated features in multi-dimensional space.

Our code is open-sourced at `https://github.com/wenchieh/eaglemine`, and most of the datasets we use are publicly available online. The supplementary material [1] provides proof, detailed information and additional experiments.

2 Related Work

Supported by human vision theory, including visual saliency, color sensitive, depth perception and attention of vision system [17], visualization techniques [38,5] and HCI tools help to get insight into data [35,2]. Scagnostic [35,6] diagnoses the anomalies from the plots of scattered points. [39] improves the detection by statistical features derived from graph-theoretic measures. Net-Ray [22] visualizes and mines adjacency matrices and scatter plots of a large graph, and discovers some interesting patterns.

For graph anomaly detection, [30,21] find communities and suspicious clusters with spectral-subspace plots. SpokEn[30] considers the "eigenspokes" on EE-plot produced by pairs of eigenvectors, and is later generalized for fraud detection. As more recent works, dense block detection has been proposed to identify anomalous patterns and suspicious behaviors [26,18]. Fraudar[18] proposed a densest subgraph-detection method that incorporates the suspiciousness of nodes and edges during optimization.

Density based methods, like DBSCAN [13] can detect clusters of arbitrary shape and data distribution, while the clustering performance relies on density threshold. STING [37] hierarchically merges grids in lower layers to find clusters with a given density threshold; Clustering algorithms [31] derived from the watershed transformation [36], treat pixel region between watersheds as one cluster, and only focus on the final results and ignores the hierarchical structure of clusters. [7] proposed hierarchical clustering method, "HDBSCAN", while its complexity is prohibitive for very large dataset (like graphs) and the "outlierness" score is not line with our expectations. Community detection algorithms [27], modularity-driven clustering, and cut-based methods [32] usually can't handle large graphs with million nodes or fail to provide intuitive and interpretable result when applying to graph clustering.

A comparison between EagleMine and the majority of the above methods is summarized in Table 1. Our method EagleMine is the only one that matches all specifications.

3 Proposed Model

Consider a graph $\mathcal{G}$ with node set $\boldsymbol{V}$ and edge set $\boldsymbol{E}$. $\mathcal{G}$ can be either homogeneous, such as friendship/following relations, or bipartite as users rating restaurants. In some feature space of graph nodes, our goal is to optimize the consistent node-group assignment with human visual recognition, and the goodness-of-fit (GoF) of node distribution in groups. So we map the node into a (multi-dimensional) histogram constructed based on a feature space, which can include multiple node features. Considering the histogram $\mathcal{H}$ with dimension $dim(\mathcal{H})$, we use h to denote the number of nodes in a bin, and $\boldsymbol{b}$ to denote a bin, when without ambiguity.

Model: To summarize the histogram $\mathcal{H}$ in a feature space of graph nodes, we utilize some statistical distributions as vocabulary to describe the node groups in $\mathcal{H}$. Therefore, our vocabulary-based summarization model consists of **Configurable vocabulary**: statistical distributions $\mathcal{Y}$ for describing node groups of $\mathcal{H}$ in a feature space; **Assignment variables**: $\mathcal{S} = \{s_1, \cdots, s_C\}$ for the distribution assignment of C node groups; **Model parameters**: $\Theta = \{\theta_1, \cdots, \theta_C\}$ for distributions in each node group, E.g. the mean and variance for normal distribution. **Outliers:** unassigned bins $\mathcal{O}$ in $\mathcal{H}$ for outlier nodes.

In terms of the configurable vocabulary $\mathcal{Y}$, it may include any suitable distribution, such as Uniform, Gaussian, Laplace, and exponential distributions or others, which can be tailored to the data and characteristics to be described.

4 Our Proposed Method

In human vision and cognitive system, connected components can be rapidly captured[11,28] with a top-to-bottom recognition and hierarchical segmentation manner [3]. Therefore, this motivates us to identify each node group as an inter-connected and intra-disjointed dense area in heat map, which guides the refinement for smoothing, and to organize and explore connected node groups by a hierarchical structure, as we will do.

Our proposed **EagleMine** algorithm consists of two steps:

- Build a hierarchical tree $\mathcal{T}$ of node groups for $\mathcal{H}$ in some feature space with WaterLevelTree algorithm.
- Search the tree $\mathcal{T}$ and get summarization of $\mathcal{H}$ with TreeExplore algorithm.

EagleMine hierarchically detects micro-clusters in the $\mathcal{H}$, then computes the optimal summarization including the model parameters Θ, and the assignment S for each node group, and outliers indices $\mathcal{O}$ in final. We elaborate each step in the following subsections.

4.1 Water-Level Tree Algorithm

In the histogram $\mathcal{H}$, we imagine an area consisting of jointed positive bins ($h > 0$) as an ***island***, and the other bins as ***water area***. Then we can flood the island areas, making those bins with $h < r$ to be underwater, i.e., setting those $h = 0$, where r is a water level. Afterwards, the remaining positive bins form new islands in condition of water level r.

To organize all the islands in different water levels, we propose a water-level tree structure, where each node represents an island and each edge represents the relationship:

Algorithm 1 WaterLevelTree Algorithm

Input: Histogram $\mathcal{H}$.
Output: Water-level tree $\mathcal{T}$.
1: $\mathcal{T} = \{$positive bins in $\mathcal{H}$ as root$\}$.
2: **for** $r = 0$ to $\log h_{max}$ by step ρ **do**
3: $\quad \mathcal{H}^r$: assign $h \in \mathcal{H}$ to zero if $\log h < r$.
4: $\quad \mathcal{H}^r = \mathcal{H}^r \circ \mathbf{E}$. ▷ *binary opening* to smooth.
5: $\quad$ islands $\mathcal{A}^r = \{$jointed bin areas in $\mathcal{H}^r\}$.
6: $\quad$ link each island in $\mathcal{A}^r$ to its parent in $\mathcal{T}$.
7: **end for**
8: **Contract** $\mathcal{T}$: iteratively remove each single-child island and link its children to its parent.
9: **Prune** $\mathcal{T}$: heuristically remove noise nodes.
10: **Expand** islands in $\mathcal{T}$ with extra neighbors.
11: **return** $\mathcal{T}$

where a child island at a higher water level comes from a parent island at a lower water level. Note that increasing r from 0 corresponds to raising the water level and moving from root to leaves.

The WaterLevelTree algorithm is shown in Alg. 1. We start from the root, and raise water level r in logarithmic scale from 0 to $\log h_{max}$ with step ρ, to account for the power-law-like distribution of h, where $h_{max} = \max \mathcal{H}$. We use the binary opening[5] operator ($\circ$) [15] for smoothing each internally jointed island, which is able to remove small isolated bins (treated as noise), and separate weakly-connected islands with a specific structure element. Afterwards, we link each island at current level r_{curr} to its parent at lower water level r_{prev} of the tree. The flooding process stops until r reaches the maximum level — $\log h_{max}$. Subsequently, we propose following steps to refine the raw tree $\mathcal{T}$ (the pictorial explanation for each step are given in the supplementary [1]):

Contract: The current tree $\mathcal{T}$ may contain many ties, meaning no new islands separated, which are redundant. Hence we *search the tree using depth-first search; once a single-child node is found, we remove it and link its children to its parent.*

Prune: The purpose of pruning is to smooth away noisy peaks on top of each island, arising from fluctuations of h between neighbor bins. Hence we *prune such child branches (including children's descendants) based on their total area size: the ratio of the sum of h in child bins to the sum of h in parent bins, is no less than* 95%.

Expand: We include additional surrounding bins into each island to avoid over-fitting for learning distribution parameters and to eliminate the possible effect of uniform step ρ for logarithmic scale. Hence we *iteratively augment towards positive bins around each island by a step of one-bin size until islands touch each other, or doubling the number of bins as contained in original island.*

Comparably in the Watershed formalization [36], the foreground of $\mathcal{H}$ are defined as catchment basins for clustering purpose, and can capture the boundaries between clusters as segmentation. We will see in experiments (sec. 5.3 and Fig. 3), the segmentation in

[5] Binary opening is a basic workhorse of morphological noise removal in computer vision and image processing. Here we use $\underbrace{2 \times \cdots \times 2}_{dim(\mathcal{H})}$ square-shape "probe".

Algorithm 2 TreeExplore Algorithm

Input: WaterLevelTree $\mathcal{T}$
Output: summarization $\{\mathcal{S}, \Theta, \mathcal{O}\}$.
1: $\Theta = \emptyset$.
2: $\mathcal{S}$ = decide the distribution type s_α from vocabulary for each island in $\mathcal{T}$.
3: Search $\mathcal{T}$ with BFS to iteratively conduct following to each node: use *DistributionFit* to determine the parameter; apply Hypothesis test to select optimal one; and insert result into Θ and update $\mathcal{S}$.
4: **Stitch** and replace promising distributions in $\mathcal{S}$, then update Θ.
5: Decide outliers $\mathcal{O}$ deviating from the recognized groups.
6: **return** summarization $\{\mathcal{S}, \Theta, \mathcal{O}\}$.

Watershed approximates the islands in one level of tree $\mathcal{T}$, with a threshold parameter for background. STING also selects clusters in the same level, and needs a density threshold; HDBSCAN extracts hierarchies with MST that can not capture trees with any branches. However, EagleMine has no tuning parameters, and then searches the water-level tree to find the best combination of islands, which may come from different levels (see sec. 4.2).

4.2 Tree Explore Algorithm

With the water-level tree and describing vocabulary, we can then determine the optimal node groups and their summarization. The main procedure is described in Alg. 2, where we decide the distribution vocabulary s_α for each tree node (island) α, search the tree with BFS, select the optimal islands with some criteria, and refine the final results using stitching. In addition, we believe the pictorial illustration in supplement [1] will offer intuitive explanation for the algorithm.

We now describe our vocabulary Θ. Truncated Gaussian distribution [34] is a flexible model for capturing clusters of different shapes, like line, circle, and ellipse, or their truncation in 2D case. Due to the discrete unit bins in $\mathcal{H}$, the *discretized, truncated, multivariate* Gaussian distribution (DTM Gaussian for short) with the mean $\boldsymbol{\mu}$ and co-variance $\boldsymbol{\Sigma}$ as parameter is used as one of the vocabulary. Observing the multi-mode distribution of islands (skewed triangle-like island in Fig. 1a) which exist in many different histogram plots and contains the majority of graph nodes, we add Mixture of DTM Gaussians as another vocabulary term to capture these complex structures.

In general, to decide the assignment $\mathcal{S}$ of vocabulary to each island, we can use distribution-free hypothesis test, like Pearson's χ^2 test, or other distribution specified approaches. Here, we heuristically assign Mixture of DTM Gaussians to the island containing the largest number of graph nodes at each tree level, and DTM Gaussian to other islands for simplicity. After vocabulary assignment, we use the maximum likelihood estimation to learn the parameters $\theta_\alpha \in \Theta$ for a island α, which $\theta_\alpha = \{\boldsymbol{\mu}_\alpha, \boldsymbol{\Sigma}_\alpha, \tilde{N}_\alpha\}$ and $\tilde{N}_\alpha = \sum_{(i_1,\cdots,i_F)\in\alpha} \log h_{i_1,\cdots,i_F}$. Let $DistributionFit(\alpha, s_\alpha)$ denote the step of learning the parameter θ_α.

Afterwards, we search along the tree $\mathcal{T}$ with BFS to select the optimal combination of clusters. In principle, metrics like AIC and BIC in machine learning and Pearson's χ^2 test and K-S test in statistics, can be adopted to determine whether to explore the children of $\mathcal{T}$. Here we utilize statistical hypothesis test to select models for its better

adaptation and performance in experiments, which measure the statistical significance of the null hypothesis [10,16]. The null hypothesis for searching the children of island α in $\mathcal{T}$ is:

$$\boldsymbol{H}_0\text{: the bins of island } \alpha \text{ come from distribution } s_\alpha.$$

If $\boldsymbol{H}_0$ is not rejected, we stop searching the island's children. Otherwise, we further explore the children of α.

Specifically, We apply this hypothesis test to an island based on its binary image, which focuses on whether the island's shape looks like a truncated Gaussian or mixture. Simply, we project the bin data to some dimensions and apply the test according to projection pursuit [19] and G-means [16]. We implement the Quadratic class 'upper tail' Anderson-Darling Statistic test[6] [9,33] (with 1% significance level) due to the truncation. And we accept the null hypothesis $\boldsymbol{H}_0$ only when the test is true for all dimension projections. If one of them is rejected, $\boldsymbol{H}_0$ will be rejected. Finally, we get the node groups to summarize the histogram until the BFS stops.

Stitch: some islands from different parents are physically close to each other. In such case, those islands can probably be summarized by the same distribution. So we use *stitch* process in step 4 to merge them by hypothesis test as well. The stitch process stops until no changes occur. When there are multiple pairs of islands to be merged at the same time, we choose the pair with the least average log-likelihood reduction after stitching:

$$(\alpha_{i^*}, \alpha_{j^*}) = \arg\min_{i,j} \frac{\mathcal{L}_i + \mathcal{L}_j - \mathcal{L}_{ij}}{\#\text{points of } \alpha_i \text{ and } \alpha_j}$$

where α_i and α_j are the pairs of islands to be merged, $\mathcal{L}_{(\cdot)}$ is log-likelihood of a island, and $\mathcal{L}_{ij}$ is the log-likelihood of the merged island.

Outliers and suspiciousness score: The outliers comprise of the bins far away from any distribution of the identified node groups (i.e. with probability $< 10^{-4}$). Intuitively, the majority island containing the most nodes is normal, so we define the weighted KL-divergence of an island from the majority island as its suspiciousness score:

Definition 1 (Suspiciousness). *Given the parameter θ_m for the majority island, the suspiciousness of the island α_i described by distribution with parameter θ_i is:*

$$\kappa(\theta_i) = \log \bar{d}_i \cdot \sum_{\boldsymbol{b} \in \alpha_i} N_i \cdot KL\left(P(\boldsymbol{b} \mid \theta_i) \,||\, P(\boldsymbol{b} \mid \theta_m) \right)$$

where $P(\boldsymbol{b} \mid \theta)$ is the probability in the bin $\boldsymbol{b}$ for the distribution with θ as parameter, N_i is the number of nodes in the island i, and we use the logarithm of $\bar{d}_i$, average degree of all graph nodes in the island i, as the weight based on the domain knowledge that if other features are the same, higher-degree nodes are more suspicious.

Time complexity: Given features associated with nodes $\boldsymbol{V}$, generating the histogram takes $O(|\boldsymbol{V}|)$ time. Let $nnz(\mathcal{H})$ be the number of non-empty bins in $\mathcal{H}$ and $\mathcal{C}$ be the number of clusters. Assume the the number of iterations for learning parameters in $DistributionFit(\cdot)$ is T, then we have (proofs are in our supplementary material [1]):

Theorem 1. *The time complexity of EagleMine is* $O(\frac{\log h_{max}}{\rho} \cdot nnz(\mathcal{H}) + \mathcal{C} \cdot T \cdot nnz(\mathcal{H}))$.

[6] This measures the goodness-of-fit of the left-truncated Gaussian distribution.

Table 2: Dataset statistics summary and synthetic settings.

	# of nodes	# of edges	Content	Injected Block
BeerAdvocate [29]	(33.37K, 65.91K)	1.57M	rate	1k × 500, 2k × 1k
Flickr	(1.4M, 466K)	1.89M	user to group	2k × 2k, 4k × 2k
Amazon	(2.14M, 1.23M)	5.84M	rate	-
Yelp	(686K, 85.54K)	2.68M	rate	-
Tagged	(2.73M, 4.65M)	150.8M	anonymized Links	-
Youtube	(3.22M, 3.22M)	9.37M	who-follow-who	-
Sina weibo	(2.75M, 8.08M)	50.1M	user-retweet-msg	-

5 Experiments

We design the experiments to answer the following questions: **[Q1] Anomaly detection**: How does EagleMine's performance on anomaly detection compare with the state-of-art methods? How much improvement does the visual-inspired information bring? **[Q2] Summarization**: Does EagleMine give significant improvement in concisely summarizing the graph? Does it accurately identify micro-clusters that agree with human vision? **[Q3] Scalability**: Is EagleMine scalable with regard to the data size?

The dataset[7] information used in our experiments is illustrated in Table 2. The Tagged[14] dataset was collected from Tagged.com social network website. It contains 7 anonymized types of links between users, and here we only choose the links of type-6, which is a homogeneous graph. The microblog Sina Weibo dataset was crawled in November 2013 from weibo.com, consisting of user-retweeting-message (bipartite) graph.

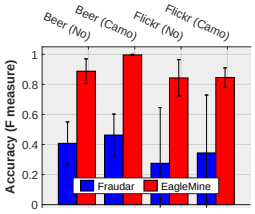
5.1 Q1. Anomaly detection

To demonstrate EagleMine can effectively detect anomalous, we conduct experiments on both synthetic and real data, and compare the performance with state-of-the-art fraud detection algorithms GetScoop[21], SpokEn [30], and Fraudar [18].

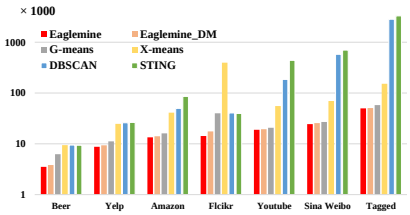
In the synthetic case, we inject different size fraud (as a block) with and without random camouflage into real datasets as Table 2 shows, where the ratio of camouflage is set to 50%, i.e. randomly selecting different objects as the same size as the targets. For BeerAdovate, the density of injected fraud is 0.05. For Flickr, the density of injected fraud are 0.05, 0.1, 0.2. We use F score for nodes on both sides of injected block to test the detection accuracy, and report the averaged result over above trials for each dataset in Fig. 2a. GetScoop and SpokEn are omitted since they fail to catch any injected object. It is obvious that EagleMine consistently outperforms Fraudar and achieves less variance for the injection cases with and without camouflages.

To verify that EagleMine accurately detects anomalies in Sina weibo data, we labeled these nodes, both user and message, from the results of baselines, and sampled nodes of our suspicious clusters from EagleMine, since that it is impossible to label

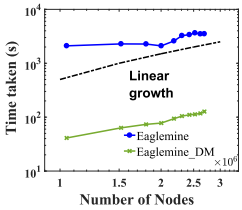
[7] The public datasets are available at: Amazon: `http://konect.uni-koblenz.de/networks/amazon-ratings`, Yelp: `https://www.yelp.com/dataset_challenge`, Flickr: `https://www.aminer.cn/data-sna#Flickr-large`, Youtube: `http://networkrepository.com/soc-youtube.php`, Tagged: `https://linqs-data.soe.ucsc.edu/public/social_spammer/`.



(a) Accuracy for detecting injected fraud.*



(b) MDL Quantitative Evaluation.†



(c) Running time

Fig. 2: **EagleMine performance for anomaly detection, summarization, and scalability**. (a) EagleMine achieves best accuracy for detecting injected fraud for BeerAdvocate ('Beer' as the abbr.) and Flickr data. *Note that GetScoop and spokEn are omitted for failing to catch any injected object. (b) MDL is compared on different datasets. EagleMine achieves the shortest description code length, which means concise summarization, and outperforms all other baselines (†Watershed clustering method is omitted due to its MDL results is even much larger than the worst case). (c) *blue* curve shows the running time of EagleMine v.s. # of node in graph in log-log scale.

all the nodes. Our labels were based on the following rules (like [18]): **1)** deleted user-accounts/messages by the online system[8] **2)** a lot of users that share unusual login-names prefixes, and other suspicious signals: approximately the same sign-up time, friends and followers count. **3)** messages about advertisement or retweeting promotion, and having lots of *copy-and-paste* text content. In total, we labeled 5,474 suspicious users and 4,890 suspicious messages.

The anomaly detection results are reported in Fig. 1g. Using AUC to quantify the quality of the ordered result from the algorithm, the sampled nodes from micro-clusters are ranked in descendant order of hubness or authority. The results show that EagleMine achieves more than 10% and about 50% improvement for anomalous user and msg detection resp., outperforming the baselines. The anomalous users detected by Fraudar and SpokEn only fall in the micro-cluster ① in Fig. 3e, since their algorithms can only focus on densest core in a graph. But EagleMine detects suspicious users by recognizing noteworthy micro-clusters in the whole feature space. Simply put, EagleMine detects more anomalies than the baselines, identifying more extra micro-clusters ②, ③, and ④.

5.2 Case study and found patterns

As discussed above, the micro-clusters ③ and ④ in out-degree vs hubness Fig. 3e contains those users frequently rewtweet non-important messages. Here we study the behavior patterns of micro-clusters ① and ② on the right side of the majority group. Note that almost half of the users are deleted by system operators, and many existing users share unusual name prefixes as Fig. 1e shown.

What patterns have we found? The Fig. 1f shows the 'Jellyfish' structure of the subgraph consisting of users from micro-clusters ① and ②. The head of 'Jellyfish' is the densest core (①), where the users created unusual dense connections to a group of messages, showing high hubness. The users (spammers or bots) aggressively 'copy-and-paste' many advertising messages a few times, which includes 'new game', 'apps in

[8] The status is checked three years later (May 2017) with API provided by Sina weibo service.

IOS7', and 'Xiaomi Phone', Their structure looks like 'Jellyfish' tail. Thus the bots in ② shows lower hubness than those in ①, due to the different spamming strategies, which are overlooked by density-based detection methods.

5.3 Q2. Summarization Evaluation on Real Data

We select X-means, G-means, DBSCAN and STING as the comparisons, the setting details are described in supplements. We also include EagleMine (DM) by using multivariate Gaussian description. We chose the feature spaces as degree vs pagerank and degree vs triangle for Tagged dataset, and choose in-degree vs authority and out-degree vs hubness for the rest.

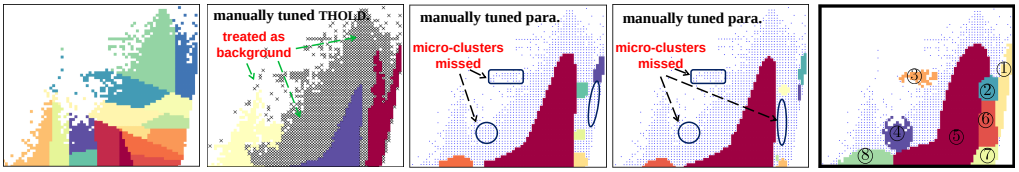
We use Minimum Description Length (MDL) to measure the summarization as [4] do, by envisioning the problem of clustering as a compression problem. In short, it follows the assumption that the more we can compress the data, the more we can learn about its underlying patterns. The best model has the smallest MDL length. The MDL lengths for the baselines are calculated as [12,8,4]. With the same principle, the MDL of EagleMine is: $L = \log^*(C) + L_{\mathcal{S}} + L_{\Theta} + L_{\mathcal{O}} + L_{\epsilon}$; details are listed in the supplementary [1].

The comparison results of MDL are reported in Fig. 2b. We can see that EagleMine achieves the shortest description length, indicating a concise and good summarization. Compared with the competitors, EagleMine reduces the MDL code length more 26.2% at least and even 81.1% than G-means and STING resp. on average, it also outperforms EagleMine (DM) over 6.4%, benefiting from a proper vocabulary selection. Therefore, EagleMine summarizes histogram with recognized groups in the best description length.

Besides the quantitative evaluation for summarization, we illustrate the results on 2D histogram for vision-based qualitative comparison. Due the the space limit, here we only exhibit the results for Sina Weibo dataset. As Fig. 3 shows, the plot features are user's out-degree and hubness indicating how many important messages retweeted. Without removing some low-density bins as background, Watershed algorithm easily identified all the groups into one or two huge ones. Hence we manually tuned the threshold of background to attain a better result, which is similar to the groups in a level of our water-level tree. The background for Watershed is shown with gray color in Fig. 3b. As we can see, Watershed only recognized a few very dense groups while failing to separate the two groups on the right and leaving other micro-clusters unidentified. Our EagleMine recognized groups in a more intuitive way, and identify those micro-clusters missed by DBSCAN and STING. Note that the user deletion ratio in the missed micro-clusters ① and ③ is unusually high, and they were suspended by the system operators for anti-spam. Besides, those micro-clusters ③ and ④ include the users have high out-degree but low-hubness, i.e., users retweeting many non-important messages (e.g., advertisements). Hence, EagleMine identify very useful micro-clusters automatically as human vision does.

5.4 Q3. Scalability

Fig. 2c shows the near-linear scaling of EagleMine's running time in the numbers graph nodes. Here we used Sina weibo dataset, we selected the snapshot of the graph, i.e., the reduced subgraph, according to the timestamp of edge creation in first $3, 6, \ldots, 30$ days. Slope of *black dot* line indicates linear growth.



(a) G-means (b) Watershed (c) DBSCAN (d) STING (e) EagleMine

Fig. 3: EagleMine visually recognizes better node groups than clustering algorithms for the feature space in Fig.1a. Watershed (with a threshold for image background), DBSCAN, and STING are mannaully tuned to have a relatively better results. The blue scattering points in (c)-(e) denote individual outliers. Even though DBSCAN and STING are extensively manually tunned, some micro-clusters of low density are missed.

6 Conclusions

We propose a tree-based approach EagleMine to mine and summarize all point groups in a heatmap of scatter plots. EagleMine finds optimal clusters based on a water-level tree and statistical hypothesis tests, and describes them with a configurable model vocabulary. EagleMine can automatically and effectively summarize the histogram and node groups, detects explainable anomalies on synthetic and real data, and can scale up linearly. In general, the algorithm is applicable to any two/multi-dimensional heatmap.

Acknowledgments

This material is based upon work supported by the Strategic Priority Research Program of CAS (XDA19020400), NSF of China (61772498, 61872206, 61425016, 91746301), and the Beijing NSF (4172059).

References

1. Supplementary document (proof and additional experiments). `https://goo.gl/ZjMwYe`
2. Akoglu, L., Chau, D.H., Kang, U., Koutra, D., Faloutsos, C.: Opavion: Mining and visualization in large graphs. In: SIGMOD. pp. 717–720 (2012)
3. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. PAMI pp. 898–916 (2011)
4. Böhm, C., Faloutsos, C., Pan, J.Y., Plant, C.: Robust information-theoretic clustering. In: KDD
5. Borkin, M., Gajos, K., Peters, A., Mitsouras, D., Melchionna, S., Rybicki, F., Feldman, C., Pfister, H.: Evaluation of artery visualizations for heart disease diagnosis. IEEE Trans. on Visualization and Computer Graphics pp. 2479–2488 (2011)
6. Buja, A., Tukey, P.A.: Computing and graphics in statistics. Springer-Verlag New York.
7. Campello, R.J.G.B., Moulavi, D., Zimek, A., Sander, J.: Hierarchical density estimates for data clustering, visualization, and outlier detection. ACM TKDD
8. Chakrabarti, D., Papadimitriou, S., Modha, D.S., Faloutsos, C.: Fully automatic cross-associations. In: SIGKDD. pp. 79–88 (2004)
9. Chernobai, A., Rachev, S.T., Fabozzi, F.J.: Composite goodness-of-fit tests for left-truncated loss samples. In: Handbook of Financial Econometrics and Statistics. Springer
10. Cubedo, M., Oller, J.M.: Hypothesis testing: a model selection approach (2002)

11. DiCarlo, J.J., Zoccolan, D., Rust, N.C.: How does the brain solve visual object recognition? Neuron (2012)
12. Elias, P.: Universal codeword sets and representations of the integers. IEEE trans. on information theory pp. 194–203 (1975)
13. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: KDD (1996)
14. Fakhraei, S., Foulds, J., Shashanka, M., Getoor, L.: Collective spammer detection in evolving multi-relational social networks. In: SIGKDD. KDD '15, ACM (2015)
15. Gonzalez, R.C., Woods, R.E.: Image processing. Digital image processing (2007)
16. Hamerly, G., Elkan, C.: Learning the k in k-means. NIPS (2004)
17. Heynckes, M.: The predictive vs. the simulating brain: A literature review on the mechanisms behind mimicry. Maastricht Student Journal of Psychology and Neuroscience (2016)
18. Hooi, B., Song, H.A., Beutel, A., Shah, N., Shin, K., Faloutsos, C.: Fraudar: Bounding graph fraud in the face of camouflage. In: SIGKDD. pp. 895–904 (2016)
19. Huber, P.J.: Projection pursuit. Annals of Statistics **13**(2), 435–475 (1985)
20. Jiang, M., Cui, P., Beutel, A., Faloutsos, C., Yang, S.: Catchsync: catching synchronized behavior in large directed graphs. In: SIGKDD (2014)
21. Jiang, M., Cui, P., Beutel, A., Faloutsos, C., Yang, S.: Inferring strange behavior from connectivity pattern in social networks. In: PAKDD (2014)
22. Kang, U., Lee, J.Y., Koutra, D., Faloutsos, C.: Net-ray: Visualizing and mining billion-scale graphs. In: PAKDD (2014)
23. Kang, U., Meeder, B., Faloutsos, C.: Spectral analysis for billion-scale graphs: Discoveries and implementation. In: PAKDD (2011)
24. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. JACM
25. Koutra, D., Jin, D., Ning, Y., Faloutsos, C.: Perseus: an interactive large-scale graph mining and visualization tool. VLDB (2015)
26. Kumar, R., Novak, J., Tomkins, A.: Structure and evolution of online social networks. In: Link mining: models, algorithms, and applications. Springer (2010)
27. Lancichinetti, A., Fortunato, S.: Community detection algorithms: a comparative analysis. Physical review E p. 056117 (2009)
28. Liu, X.M., Ji, R., Wang, C., Liu, W., Zhong, B., Huang, T.S.: Understanding image structure via hierarchical shape parsing. In: CVPR (2015)
29. McAuley, J.J., Leskovec, J.: From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In: WWW (2013)
30. Prakash, B.A., Sridharan, A., Seshadri, M., Machiraju, S., Faloutsos, C.: Eigenspokes: Surprising patterns and scalable community chipping in large graphs. In: PAKDD (2010)
31. Roerdink, J.B., Meijster, A.: The watershed transform: Definitions, algorithms and parallelization strategies. Fundamenta informaticae (2000)
32. Schaeffer, S.E.: Graph clustering. Computer science review (2007)
33. Stephens, M.A.: Edf statistics for goodness of fit and some comparisons. Journal of the American statistical Association pp. 730–737 (1974)
34. Thompson, H.R.: Truncated normal distributions. Nature **165**, 444–445 (1950)
35. Tukey, J.W., Tukey, P.A.: Computer graphics and exploratory data analysis: An introduction. Nat Computer Graphics Association (1985)
36. Vincent, L., Soille, P.: Watersheds in digital spaces: an efficient algorithm based on immersion simulations. PAMI pp. 583–598 (1991)
37. Wang, W., Yang, J., Muntz, R., et al.: Sting: A statistical information grid approach to spatial data mining. In: VLDB. pp. 186–195 (1997)
38. Ware, C.: Color sequences for univariate maps: theory, experiments and principles. IEEE Computer Graphics and Applications pp. 41–49 (Sept 1988)
39. Wilkinson, L., Anand, A., Grossman, R.: Graph-theoretic scagnostics. Proceedings - IEEE Symposium on Information Visualization, INFO VIS pp. 157–164 (2005)