EagleMine: Vision-Guided Mining in Large Graphs

Wenjie Feng\textsuperscript{1,2}, Shenghua Liu\textsuperscript{1}, Christos Faloutsos\textsuperscript{3}, Bryan Hooi\textsuperscript{3}, Huawei Shen\textsuperscript{1}, Xueqi Cheng\textsuperscript{1}

\textsuperscript{1}CAS Key Laboratory of Network Data Science \& Technology, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
\textsuperscript{2}University of Chinese Academy of Sciences (UCAS), Beijing 100049, China
\textsuperscript{3}School of Computer Science, Carnegie Mellon University, PA, USA

\texttt{fengwenjie@software.ict.ac.cn, liu.shengh@gmail.com, christos@cs.cmu.edu}
\texttt{bhooi@andrew.cmu.edu, shenhuawei@ict.ac.cn, cxq@ict.ac.cn}

\section*{ABSTRACT}

Given a graph with millions of nodes, what patterns exist in the distributions of node characteristics, and how can we detect them and separate anomalous nodes in a way similar to human vision?

In this paper, we propose a vision-guided algorithm, EagleMine, to recognize and summarize graph node groups in feature spaces. EagleMine hierarchically discovers node groups, and each group is an internally connected dense area in some feature space. EagleMine utilizes a water-level tree to capture group structures according to vision-based intuition at multiple resolutions. EagleMine traverses the water-level tree, applying statistical hypothesis test to determine the optimal node groups that should be fitted along the path. Moreover, EagleMine can identify anomalous micro-clusters (i.e., micro-size groups), who exhibit very similar behavior in some feature space, and deviate away from the majority. Experiments on real-world data show that our method can recognize intuitive groups as human vision does, and achieve the best performance in summarization compared to baselines. In terms of anomaly detection, EagleMine also outperforms well-known state-of-the-art graph-based methods by significantly improving accuracy in a microblog dataset.

\section*{1 INTRODUCTION}

Given real-world graphs with millions of nodes and connections, the most intuitive way to explore the graphs is to construct a scatter-plot of graph nodes in coordinates of correlated features, known as a correlation plot \cite{15, 47}. Usually a heat map of those scatter-plot points is used to show their density, which is a two-dimensional histogram \cite{24}. In this histogram, people can visually recognize nodes gathering into disjointed dense areas separately as groups (Fig. 1), which help to explore patterns (like communities, lockstep behaviors) and detect anomalies (e.g., fraudsters, attackers, fake-reviews, outlier etc.) in a more interpretable way \cite{13, 26}.

In particular, a graph can represent friendships in Facebook, the most intuitive way to present the graphs is to construct a scatter-plot of graph nodes in coordinates of correlated features, known as a correlation plot. Usually a heat map of those scatter-plot points is used to show their density, which is a two-dimensional histogram. In this histogram, people can visually recognize nodes gathering into disjointed dense areas separately as groups (Fig. 1), which help to explore patterns (like communities, lockstep behaviors) and detect anomalies (e.g., fraudsters, attackers, fake-reviews, outlier etc.) in a more interpretable way. In the paper we demonstrate some of the 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., they have many outgoing edges to non-important nodes, probably, customers, that paid them) \cite{28}.

- **ii** #triangles vs degree - Fig. 1b - spotting a near-clique group (too many triangles, for their degree), as well as star-like constellations (too few triangles for such high degree) \cite{27}.

In this paper, we propose EagleMine, a novel tree-based mining approach to recognize and summarize the node groups in a correlation plot of graph nodes. EagleMine can identify anomalous micro-clusters. Experiments show that EagleMine outperforms baselines and achieves better performance both in quantitative (i.e., code length for compact model description) and qualitative (i.e., consistent with vision-based judgment) comparisons. EagleMine detects a micro-cluster of hundreds of bots in a real-world microblog data, Sina Weibo, which presents 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. 2b).

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{figure1.png}
\caption{Heat maps (histograms) of correlation plots.}
\end{figure}

\begin{itemize}
\item recognize and monitor the node groups as human vision does?
\item summarize the node groups in the feature space and identify suspicious micro-clusters?
\end{itemize}

‘Micro-cluster’ refers to relatively small group of people, that exhibit very similar behavior in the feature space. In the paper we demonstrate some of the possible feature spaces, namely:

\begin{itemize}
\item i out-degree vs hubness - Fig. 1a - this can spot nodes with high out-degree, but low hubness score (i.e., they have many outgoing edges to non-important nodes, probably, customers, that paid them) \cite{28}.
\item ii #triangles vs degree - Fig. 1b - spotting a near-clique group (too many triangles, for their degree), as well as star-like constellations (too few triangles for such high degree) \cite{27}.
\end{itemize}

In this paper, we propose EagleMine, a novel tree-based mining approach to recognize and summarize the node groups in a correlation plot of graph nodes. EagleMine can identify anomalous micro-clusters. Experiments show that EagleMine outperforms baselines and achieves better performance both in quantitative (i.e., code length for compact model description) and qualitative (i.e., consistent with vision-based judgment) comparisons. EagleMine detects a micro-cluster of hundreds of bots in a real-world microblog data, Sina Weibo, which presents 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. 2b).
Figure 2: Our proposed EagleMine achieves effective results on microblog Sina Weibo data. (a) EagleMine summarizes the graph nodes in a feature space with truncated Gaussian distributions, where we show the $1.5\Sigma$ and $3\Sigma$ ellipses. $\Sigma$ is the covariance matrix. The majority group (having the largest area and the most of nodes) in the middle is described by two overlapped and truncated Gaussians. (b) highlights some micro-clusters, including a disconnected small network, and very suspicious micro-clusters. A user name list on the right side shows the name patterns of bots in a micro-cluster, where 182x: 'best*' means 182 bots share unusual prefix 'best'. (c) The blue curve shows the running time of EagleMine, compared to a linear function, and the green curve shows the generalization vocabulary (DM-Gaussian) of EagleMine. (d)-(h) are the results of recognizing node groups by the baselines and our EagleMine respectively. A relatively good results of Watershed is achieved by manually tuning parameters. Still, some micro-clusters of low density are not recognized.

In summary, the proposed EagleMine has following advantages:

- **Automated summarization**: EagleMine automatically summarizes a histogram plot derived from correlated graph features (see Fig. 2a). EagleMine recognizes node groups forming disjointed dense areas as human vision does (see Fig. 2h).

- **Effectiveness**: EagleMine detects interpretable groups, and outperforms the baselines and even those with *manually tuned parameters* in qualitative (see Fig. 2d-2h), and quantitative experiments (see Fig. 6).

- **Anomaly detection**: EagleMine can spot, and even explain anomalies on real data by identifying suspicious micro-clusters (see Fig. 2b). Compared with the graph-based anomaly detection methods, EagleMine achieves higher accuracy for finding suspicious users in Sina Weibo data.

- **Scalability**: EagleMine is scalable, with nearly linear time complexity in the number of graph nodes, and can deal with more correlated features in multi-dimensional space.

**Reproducibility**: Our code is open-sourced at [https://github.com/wenchieh/eaglemine](https://github.com/wenchieh/eaglemine), and most of the datasets we use are publicly available online.

In Section 2, we provide notions and our proposed model. In Section 3, we propose EagleMine algorithm for recognizing and summarizing graph nodes. In Section 4, we present experimental results. After discussing related work in Section 5, we offer conclusions in Section 6.
3 OUR PROPOSED METHOD

Our method EagleMine is guided by the following traits of human vision and cognitive system:

TRAIT 1. *Human vision usually detects connected components, which can be rapidly recognized by eyes despite substantial appearance variation*\(^{[14, 32]}\).

This motivates us to identify each node group as an internally connected dense area in the heatmap, and different groups are disjointed from each other, which guides the refinement for smoothing.

TRAIT 2. *Top-to-bottom recognition and hierarchical segmentation*\(^{[5]}\). *Humans organize basic elements (e.g., words, shapes, visual-areas) into higher-order groupings to generate and represent complex hierarchical structures in human cognition and visual-spatial domains.*

This suggests that organizing and exploring connected node groups should be based on a hierarchical structure, as we will do.

Algorithm 1 describes the overall structure of EagleMine. It hierarchically detects micro-clusters in the histogram \(\mathcal{H}\), and outputs the optimal summarization including the model parameters \(\Theta\), and assignment \(S\) for each node group, and outliers indices \(O\). The following subsections will elaborate each step in detail.

### 3.1 Water-level tree

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. Assume that 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 can 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 \(T\), in which each node represents an island and each edge represents the relationship: 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.

In a 2D histogram, islands are candidate groups for Trait 1; flooding process intuitively reflects how human eyes hierarchically capture different objects from the color histogram \(\mathcal{H}\), as Trait 2. For example, the gradient colors in Fig 1a depict groups at different water levels.

The WaterLevelTree algorithm is shown in Alg. 2. We start from the root, and raise water level \(r\) in logarithmic scale from 0 to \(\log h_{\text{max}}\) with step \(\rho\), to account for a power-law-like distribution of \(h\). Let \(h_{\text{max}} = \max \mathcal{H}\). We use the binary opening\(^2\) operator \((=)\)\(^{[18]}\) for smoothing each internally jointed island, which is able to remove small isolated bins (treated as noise), and also separate weakly-connected islands with a specific structure element. Afterwards, we link each island at current level \(r_{\text{curr}}\) to its parent at lower water level \(l_{\text{prev}}\) of the tree (see Fig 4a). Finally, when \(r\) reaches \(\log h_{\text{max}}\), the flooding process stops, and we build a water-level tree.

We propose the following steps to refine raw tree \(T\):

**Contract:** The current tree \(T\) may contain many nodes with only one child, meaning no new islands separated, which are redundant. Hence we search the tree using depth-first search (DFS); once a single-child node is found, we remove it and link its children to its parent. This process is shown in Fig 4a, where the dashed lines with arrow depict that the single-child’s children are linked to its parent, and the gray links are removed.

**Prune:** The purpose of pruning is to smooth an island which has noise peaks on top of the island due to fluctuations of \(h\) in neighboring bins. An example is shown at bottom right of Fig 4b. The island \(a\) at water-level \(r\), contains fluctuation noises on the top.

\(^2\) Binary opening is a basic workhorse of morphological noise removal in computer vision and image processing. Here we use \(2 \times \cdots \times 2\) square-shape "probe".
So when water level raises to \( r' \), the noises become three separated 'tiny' islands, linking to their parent \( a \). Hence we prune such child branches (including children's descendants) based on their total area size: the ratio of sum of \( h \) in child bins over sum of \( h \) in parent bin, is no more than 95\%. The pruning branches are illustrated in Fig.4b (1 and 2).

**Expand:** We include additional surrounding bins to avoid overfitting for learning distribution parameters and to eliminate the possible effect of uniform step \( \rho \) 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 in original island. The expansion of node is illustrated with shadowed rings in Fig.4b. For the node 3, the light-blue labeled irregular part represents the original island area and the outer peripheral gray bins are 1-st step expanded part as the pictorial depiction shows, and the further expansion follows similar process until it gets to the above constraints.

Note that in the Watershed formalization [43], 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 (section 4.2), the segmentation in Watershed approximates the islands in one level of tree \( T \), with a threshold parameter for background. STING also selects clusters in the same level, and needs a density threshold. However, EagleMine has no tuning parameters, and searches water-level tree to find a better combination of islands with hypothesis test, which may from different levels (see section 3.3).

### 3.2 Describing islands

The vocabulary \( \mathcal{Y} \) can contain any proper user-defined distributions. In the feature space that we may concern, we use multivariate Gaussian as one of the vocabularies. Note that many node features, such as degrees and number of triangles, typically follow a power-law distribution. Hence some bins along histogram boundary may have larger number of nodes (see the bottom bins in Fig. 1). Thus truncated Gaussian distribution is a proper choice to describe such a truncated ellipse as far as we known. Moreover, since bins in \( \mathcal{H} \) are discrete units, distributions must be discretized, namely, defining a probability function in each bin instead of probability density function. Now the discretized, truncated, multivariate Gaussian distribution (DTM Gaussian for short) is defined as follows:

**Definition 1 (DTM GAUSSIAN).** The probability function in a bin with boundary matrix \( b \) is

\[
P(b; \mu, \Sigma, \beta) = \int_{\beta} \cdot \cdot \cdot \int_{\beta} \psi(x; \mu, \Sigma, \beta) dx
\]

where \( x \) is \( F \)-dimensional variable, \( \beta, \beta \in \mathbb{R}^{F \times 2}, \beta \) is the truncation bound, and \( \psi(\cdot) \) is density function of truncated normal distribution with mean \( \mu \) and co-variance \( \Sigma \).

For 2D histogram, \( \beta = \begin{bmatrix} [0, +\infty]; [0, +\infty] \end{bmatrix} \). With DTM Gaussian, we can describe islands with shapes of lines, circles, and ellipses, and their truncation. The red-yellow oval circles in Fig. 2a depict some node clusters of DTM Gaussian described.

Observing the multi-mode distribution of islands (skewed triangle-like island in Fig. 1a), we also add mixture DTM Gaussian as another vocabulary. In our data study, this triangle-like island exists in many different histogram plots, and contains the majority of graph nodes. For example, Fig. 1a depicts users’ distribution over out-degree and hubness. The power-law of out-degree makes the density decrease along the vertical axis. Meanwhile, users with similar degree shares similar hubness, forming a nearly normal distribution in horizontal. Therefore, those majority users forms a triangle-like island in the feature space, and we use mixture DTM Gaussian for it.

To decide the assignment \( S = \{ s_1, \ldots, s_C \} \) of vocabulary to each island, we can use distribution-free statistical hypothesis test, like Pearson's \( \chi^2 \) test, or other distribution specified approaches. After vocabulary assignment, we use maximum likelihood estimation to learn the parameters \( \theta_{\alpha} \in \Theta \) for an island \( \alpha \), which \( \theta_{\alpha} = \{ \mu_{\alpha}, \Sigma_{\alpha}, \bar{N}_{\alpha} \} \) and \( \bar{N}_{\alpha} = \sum_{i} \log h_{i_1}, \ldots, i_F \). For denotation, we define function \( DistributionFit(\alpha, s_{\alpha}) \) as learning process which returns \( \theta_{\alpha} \).
Algorithm 3 TreeExplore Algorithm

**Input:** WATERLEVELTree $T$

**Output:** summarization $\{S, \Theta, O\}$.

1. $\Theta = \emptyset$.
2. $S = $ decide the distribution type $s_\alpha$ from vocabulary for each island in $T$.
3. queue $Q = $ root node of $T$.
4. while $Q \neq \emptyset$ do
   » breath-first search (BFS).
   5. $\alpha \leftarrow$ dequeue of $Q$.
   6. $\theta_\alpha = $ DistributionFit($\alpha, s_\alpha$)
   7. Hypothesis test $H_0 = $ bins of island $\alpha$ come from the distribution $s_\alpha$.
   8. if $H_0$ is rejected then
      9. enqueue all the children of $\alpha$ into $Q$.
   10. else
      11. $\Theta = \Theta + \{\theta_\alpha\}$
   12. end if
5. end while
6. Stitch and replace promising distributions in $S$, then update $\Theta$.
7. Decide outliers $O$ deviating from the recognized groups.
8. return summarization $\{S, \Theta, O\}$.

3.3 Tree explore Algorithm

With the hierarchical water-level tree and describing vocabulary, we can then determine the optimal node groups and their summarization. The procedure is described in Alg. 3, where we explore the tree with BFS, decide the distribution vocabulary $s_\alpha$ for each tree node (island) $\alpha$, select the optimal islands by using hypothesis test as a metric, and refine the final results with stitching in final.

In general, one can decide the assignment of distribution by Pearson’s $\chi^2$. However, we heuristically assign mixture DTM Gaussian to the island with the largest number of nodes in each tree level, and DTM Gaussian to other islands for simplicity.

BFS explore and hypothesis test: Afterwards, we search along the tree $T$ with BFS to select the optimal combination of clusters (see step 4 to 14). Starting from the root node, we use the following null hypothesis to decide whether to explore the children:

$H_0$: the bins of island $\alpha$ come from distribution $s_\alpha$.

Due to the variability of the value of bins in each island, we have tried Pearson’s $\chi^2$ test, BIC and AIC criteria but the extreme heights made the test and other criteria unstable. Thus, we test an island based on its binary image, which focuses on whether the island’s shape looks like a truncated Gaussian or mixture. With projection pursuit [23], we simplify the hypothesis test by projecting the bins data to one dimension where the test can be applied. The author of [19] also use this approach to perform standard Gaussian test for determining the optimal $k$ for $k$-means. And we accept the null hypothesis only when $H_0$ is true for all axes projections. We then use Quadratic class ‘upper tail’ Anderson-Darling Statistic test$^3$ [12, 41] (with 1% significant level) to test on all axes. If one of the tests is rejected, the null hypothesis will be rejected.

If $H_0$ is not rejected, we stop exploring the island’s children.

Otherwise, we further explore the children of island $\alpha$. The dashed lines with an arrow in Fig. 4c demonstrate this process. Finally, the final optimal combination of islands to summarize the histogram is achieved until the BFS search stops.

Stitch: Furthermore, some micro-clusters (islands) from different parents, e.g., $\alpha_1$ and $\alpha_2$ in Fig. 4c, are physically close to each other. In such case, those islands, separated due to sparsity and non-smoothness of data distribution or be over-split by WATERLEVELTree, can potentially be summarized by one distribution. Therefore, we use stitch process in step 15 to merge them by hypothesis test. We test and stitch every pair of such islands from different parents in $T$, until no changes occur. When there are multiple pairs of islands that be merged at the same time, we heuristically choose the pair with the least average log-likelihood reduction after stitching:

$$\arg\min_{i,j} \frac{L_i + L_j - L_{ij}}{\text{#points of } a_i \text{ and } a_j}$$

where $a_i$ and $a_j$ are the pair of islands to be merged, $L_{i(\cdot)}$ is log-likelihood of a island, and $L_{ij}$ is the log-likelihood of the merged island.

Outliers and suspiciousness score: We assign the bins that are close to a distribution (i.e., have probability at least $10^{-4}$) of a island to the group. Other bins are outliers $O$.

Furthermore, since the only one majority island fitted by mixture distribution contains the majority and normal nodes, we give the suspiciousness of other islands by the weighted KL-divergence as,

$$\kappa(\theta_i) = \log \hat{d}_i \cdot \sum_b N_i \cdot \text{KL}(P_{\theta_i}(b) || P_{\theta_m}(b)),$$

where $P_{\theta}(b)$ is the probability in the bin $b$ for the distribution with $\theta$ as parameter, $N_i$ is the number of samples in the island $i$, and we use the logarithm of $\hat{d}_i$, 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.

3.4 Time complexity

Given the features associated with nodes $V$, the time complexity for generating histogram $H$ is $O(|V|)$.

Let $M$ be the number of non-empty bins in the histogram $H$, and $C$ be the number of clusters. We use gradient-descent to learn parameters in DistributionFit(·) of EagleMine algorithm, so we assume that the number of iterations is $T$, which is related to the differences between initial and optimal objective values, and $h_{max} = \max H$ as defined before. Then we have:

**Theorem 3.1.** The time complexity of EagleMine algorithm is $O(\frac{\log h_{max}}{\rho} \cdot M + C \cdot T \cdot M)$.

4 EXPERIMENTS

We design the experiments to answer the following questions:

1. **Quantitative evaluation on real data**: Does EagleMine give significant improvement in concisely summarizing the graph?
2. **Qualitative evaluation (vision-based)**: Does EagleMine accurately identify micro-clusters that agree with human vision?
3. 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?

4. Scalability: Is EagleMine scalable with regard to the data size? The data information used in our experiments is illustrated in Table 1. The Tagged [17] 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.

4.1 Q1. Quantitative Evaluation on Real Data

The comparison algorithms and their settings are listed as follows.

- X-means: initialize with k-means and 5 clusters.
- G-means: set max_depth = 5, limiting no more than 16 clusters to avoid too many clusters; set p-value = 0.01 which is insensitive.
- DBSCAN: set Eps=1, and use \[|h_i - h_j|_{\infty}\] as distance function; we searched MinPts from the average number of nodes in a histogram bin until the max number by step 50, and manually select the one consistent well with human-inspired insight.
- STING: \[c = \frac{\text{MinPts} + 1}{2E_{\text{eps}}}\] with DBSCAN’s tuned optimal MinPts and Eps for clusters as initial, and refine the visual result by fine-tuning.
- EagleMine and EagleMine (DM): are our proposed EagleMine with DTM Gaussian and whole multivariate Gaussian respectively.

Envisioning the problem of clustering as a compression problem, we use Minimum Description Length (MDL) as the metric to measure the summarization as [7, 21] do. 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:

\[L = \log^2(C) + L_S + L_Q + L_O + L_e\]

This description model consists of the following terms:

- The number of clusters requires \[\log^2(C)\] bits.
- The assignment \(S\) of all non-outlier bins in \(H\) requires \(L_S = |C| \cdot \log(\mathcal{Y})\) bits.
- The fitting parameters of each DTM Gaussian need \((F + 2)\) free parameters and one fitted values \(\tilde{h}\). So its encoding requires \(L_0 = (0.5F^2 + 1.5F + 1)\) bits, where \(L_0\) is the floating point cost, we used \(4 \times 8\) bits in our setting. So, the total parameter code-length is \(L_0 = |C| \cdot L_0 \cdot (0.5F^2 + 1.5F + 1)\) bits.
- The outliers \(O\) require \(L_O\) bits to encode their indices.
- The model error requires \(L_e\) bits. For a bin \(b\) in group (island) \(a_i\), the expected number of nodes is \(\tilde{h} = \begin{cases} 2^{N_i}P_i(b), & \text{if } x_{hi} < \tilde{h} + \epsilon \\ \tilde{h} + \epsilon, & \text{otherwise} \end{cases}\) where \(1\) denotes the sign bit.

As for the other methods in comparison, we calculated the MDL length with the same principle [7, 10, 15]. And for the different feature space, we chose the degree vs pagerank and degree vs outdegree vs hubness for the rest.

The comparison of MDL length is reported in Fig. 6a. We can see that EagleMine achieves the shortest description length, indicating a concise and good summarization. On average, EagleMine reduces the MDL code length more than 81.6%, 79.0%, 65.5%, 20.2% compared with STING, DBSCAN, X-means and G-means respectively. We also computed the MDL for Watershed clustering method on all datasets, it is even much larger than DBSCAN, so we do not show them in Fig. 6 for a better chart view. EagleMine also outperforms EagleMine (DM) over 4.6%, benefiting from a proper vocabulary selection. Therefore, EagleMine summarizes the histogram with recognized groups in the best description length.

\[\text{Here, } \log^2(x) = \log(\log_x(x)) + \log(\log_x(x)) + \cdots \text{ where only the positive terms are included in the sum.} \]
Due to the complexity of geometric properties of distributions in high-dimensional space, we conducted EagleMine in 3-, 4- and 5-dimensional feature spaces on Amazon and Yelp datasets, and use the out-degree vs. top-k hubness (top-k left singular vectors) as features. Here, EagleMine used only digitized multivariate Gaussian as the vocabulary for simplicity. We can roughly draw the conclusion that more feature dimensions will lead to larger cost to describe for the histogram.

4.2 Q2. Qualitative evaluation (vision-based)

In this part, we illustrate the results on 2D histogram for vision-based qualitative comparison. Due to the space limit, we try to exhibit the comparisons with different baselines and in different feature spaces in an alternative way. The baselines include X-means, G-means, DBSCAN, STING, and Watershed. The feature spaces include out-degree vs. hubness, in-degree vs. authority, and triangle vs. degree.

Fig. 2d-2h shows the results on user-retweet-message graph in Sina Weibo data. 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. 2c. 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 unidentifiable. 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, they were suspended by the system operators for anti-spam. Besides, those micro-clusters 2 and 4 include the users that have high out-degree but low-hubness, i.e., users retweeting many non-important messages (e.g., advertisements). Therefore, EagleMine identify very useful micro-clusters automatically as human vision does.

Moreover, Fig. 5 illustrates more examples on different datasets and feature spaces in four groups. The original histogram plots are given at the beginning of each group following with label figures of different methods. Outlier nodes are labeled with the blue color. Different colors represent different groups by corresponding methods. From original plots, people will naturally expect that bins with the similar color (density) and jointed locations should be in one group. Hence we can see that G-means and X-means produce a number of groups, over-separating the groups recognized by human vision. Although manually tuned DBSCAN and STING can capture the majority dense region in each plot, while overlooking some suspicious micro-clusters, e.g., micro-clusters ② and ③ in Fig. 5c. Thus, EagleMine illustrates its advantages of recognizing groups, especially identifying micro-clusters, which is more consistent with human vision.

4.3 Q3. Anomaly detection

To compare the performance for anomaly detection, we labeled these nodes, both user and message, from the results of baselines, and sample nodes of our suspicious clusters from EagleMine, considering that it is impossible to label all the nodes. Our labels were based on the following rules [22]:

- user-accounts/messages which are deleted from the online system (system operators found the spams) ⑥.
- a lot of users that share unusual login-names prefixes, and other suspicious signals: approximately the same sign-up time, friends and followers count.
- messages on 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. We compared with state-of-the-art fraud detection algorithms GetScoop[25], Spoken[38], and Fraudar [22].

EagleMine returns the micro-clusters with suspiciousness scores. Table 2 lists the ranking orders of micro-clusters identified by EagleMine for Fig. 2h and 5c respectively, according to their scores.

The anomaly detection results are reported in Fig. 6b, 6c. Using the AUC (area under the ROC curve) to quantify the quality of the ordered result from the algorithm, the sampled nodes from micro-clusters are ranked in descending order of hubness or authority. The results show that EagleMine achieves more than 10% improvement in anomalous user detection, and about 50% improvement in anomalous message detection, outperforms the baselines. The anomalous users detected by Fraudar and SpokEn only fall in the micro-cluster ① in Fig. 2h, 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, by identifying extra micro-clusters ②, ③, and ④ in the feature space.

4.4 Q3. Pattern case study

As discussed above, the micro-clusters ② and ③ in out-degree vs. hubness Fig. 2h contains those users frequently retweet advertisements. 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. 2h shown.

What pattern have we found? The Fig. 6d shows the ‘Jellyfish’ structure of the subgraph consisting of users from micro-clusters.

Table 1: Dataset mining in large graphs

<table>
<thead>
<tr>
<th>Feature space</th>
<th>Content</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amazon rating [1]</td>
<td>(2.14M, 1.23M) 5.84M Rate</td>
</tr>
<tr>
<td>Android [35]</td>
<td>(1.32M, 61.27K) 2.64M Rate</td>
</tr>
<tr>
<td>BeerAdvocate [34]</td>
<td>(33.37K, 65.91K) 1.57M Rate</td>
</tr>
<tr>
<td>Yelp [2]</td>
<td>(686K, 85.54K) 2.68M Rate</td>
</tr>
<tr>
<td>Tagged [17]</td>
<td>(2.75M, 8.08M) 50.1M User-retweet-msg</td>
</tr>
<tr>
<td>Sina weibo</td>
<td>(2.75M, 8.08M) 50.1M User-retweet-msg</td>
</tr>
</tbody>
</table>

Table 2: Suspicious ranking of micro-clusters in Sina Weibo.

<table>
<thead>
<tr>
<th>Feature space</th>
<th>Suspiciousness score k(·) rankings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Out-degree vs Hubness</td>
<td>①, ②, ③, ④, ⑤</td>
</tr>
<tr>
<td>In-degree vs Authority</td>
<td>⑥, ⑦, ⑧, ⑨</td>
</tr>
</tbody>
</table>

To compare the performance for anomaly detection, we labeled these nodes, both user and message, from the results of baselines, and sample nodes of our suspicious clusters from EagleMine, considering that it is impossible to label all the nodes. Our labels were based on the following rules [22]:

- user-accounts/messages which are deleted from the online system (system operators found the spams) ⑥.
- a lot of users that share unusual login-names prefixes, and other suspicious signals: approximately the same sign-up time, friends and followers count.
- messages on 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. We compared with state-of-the-art fraud detection algorithms GetScoop[25], Spoken[38], and Fraudar [22].

EagleMine returns the micro-clusters with suspiciousness scores. Table 2 lists the ranking orders of micro-clusters identified by EagleMine for Fig. 2h and 5c respectively, according to their scores.

The anomaly detection results are reported in Fig. 6b, 6c. Using the AUC (area under the ROC curve) to quantify the quality of the ordered result from the algorithm, the sampled nodes from micro-clusters are ranked in descending order of hubness or authority. The results show that EagleMine achieves more than 10% improvement in anomalous user detection, and about 50% improvement in anomalous message detection, outperforms the baselines. The anomalous users detected by Fraudar and SpokEn only fall in the micro-cluster ① in Fig. 2h, 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, by identifying extra micro-clusters ②, ③, and ④ in the feature space.
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 retweeted the similar message collection (i.e., A in Fig. 5c). At the meantime, users from ② connected to some of messages in A, and ‘copy-and-paste’ many advertising messages on a few topics, e.g., ‘new game’, ‘apps in IOS7’, and ‘Xiaomi Phone’. Their structure looks like ‘Jellyfish’ tail. Thus the bots in ② shows lower hubness than those in ①, since of the different spamming strategies, which are overlooked by density-based detection methods.

4.5 Q4. 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 top 3, 6, . . . , 30 days. Slope of black dot line indicates linear growth.

5 RELATED WORK

For the Gaussian clusters, K-means, X-means [37], G-means [19], and BIRCH [49] (which is suitable for spherical clusters) algorithms suffer from being sensitive to outliers. Those methods are distance based, which prefer to cluster the points with a short distance in feature space forming a spherical area. Density based methods, such as DBSCAN [16] and OPTICS [4] are noise-resistant and can detect clusters of arbitrary shape and data distribution, while the clustering performance relies on density threshold for DBSCAN, and also for OPTICS to derive clusters from reachability-plot. RIC [7] enhances other clustering algorithms as a framework, using minimum description language as goodness criterion to select fitting distributions and separate noise. STING [44] hierarchically merges grids in lower layers to find clusters with a given density threshold. Clustering algorithms [39] derived from the watershed transformation [43], treat pixel region between watersheds as one cluster, and only focus on the final results and ignores the hierarchical structure of clusters. Community detection algorithms [31], modularity-driven clustering, and cut-based methods [40] usually cannot handle large graphs with million nodes or fail to provide intuitive and interpretable result when applying to graph clustering.

Supported by human vision theory, including visual saliency, color sensitive, depth perception and attention of vision system [20], visualization techniques [8, 45] and HCI tools help to get insight into data [3, 42]. SCAGNOSTIC [9, 42] diagnoses the anomalies from the plots of scattered points. [47] improves the detection by statistical features derived from graph-theoretic measures. Net-Ray [26] visualizes and mines adjacency matrices and scatter plots of a large graph, and discovers some interesting patterns.

In terms of anomaly detection in graphs, [25, 38] find communities and suspicious clusters in graph with spectral-subspace plots. SpokEN [38] considers the “eigenspokes” pattern on EE-plot produced by pairs of eigenvectors of graphs, and is later generalized for fraud detection. As more recent works, dense block detection (DBD) has been proposed to identify anomalous patterns and suspicious behaviors [11, 22, 30]. A representative work, Fraudar [22], proposed a densest subgraph-detection method that incorporates the suspiciousness of nodes and edges during optimization.

### Table 3: Comparison between algorithms.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>parameter free</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>non-spherical cluster</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>anomaly detection</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>summarization</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>linear in #nodes</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

A comparison between EagleMine and the majority of the above methods is summarized in Table 3. Our proposed method EagleMine is the only one that matches all specifications.

6 CONCLUSIONS

We propose a tree-based approach EagleMine to mine and summarize all node groups in a histogram plot of a large graph. The EagleMine algorithm finds optimal clusters based on water-level tree and statistical hypothesis test, describes them with a configurable model vocabulary, and detects some suspicious. EagleMine has desirable properties:

- **Automated summarization**: Our algorithm automatically summarizes a given histogram with vocabulary of distributions, inspired by human vision to find the graph node groups and outliers.
- **Effectiveness**: We compared EagleMine on real data with the baselines, the result shown that our detection is consistent with human vision, and also achieves better MDL in summarization.
- **Anomaly detection**: EagleMine can detect explainable anomalies on real data and achieve higher accuracy for finding suspicious users and bots micro-clusters.
- **Scalability**: EagleMine algorithm runs near linear in the number of graph nodes, and can handle the multi-dimensional correlated feature space.

To obtain better vision judgment and summarization for multi-dimensional features is still a challenge, our EagleMine-DM (with the multivariate Gaussian distribution as vocabularies) is a good choice in those cases for its comparable performance and better running times, and there are many theoretical well-defined hypothesis test approaches to be used to determine the optimal clusters.

ACKNOWLEDGMENTS

This work was partially funded by National Grand Fundamental Research 973 Program of China No. 2014CB340401, the Beijing NSF No. 4172059, the National NSF of China No. 61772498.

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation of China, or other funding parties.
EagleMine: Vision-Guided Mining in Large Graphs

Figure 6: EagleMine Alg. Performance. (a) MDL is compared on different real-world datasets. EagleMine achieves the shortest description code length, which means concise summarization, and outperforms all other baselines. (b), (c) EagleMine has the best AUC for detecting suspicious users and messages on Sina weibo. (d) The structure of anomalies identified in Sina weibo.

REFERENCES

[34] Julian John McAuley and Jure Leskovec. 2013. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In WWW. 897–908.