ECAI 2024
U. Endriss et al. (Eds.)

doi:10.3233/FAIA240788

Adaptive Fraud Detection on e-Commerce Platforms

Shaowen Tang[a,*] **and Raymond K. Wong**[a]

[a]University of New South Wales, Sydney, Australia.

Abstract. Fraud detection has attracted massive research interests due to its broad applications, especially on e-commerce websites, where fraudulent users interact with products unfairly to mislead potential customers for monetary incentives. Previous studies have made significant research progress on fraud detection, while only a few of them pay attention to adaptability and scalability of their methods. As a result, most of prior fraud detectors cannot perform consistently well across different scenarios due to issues such as sparsity, pattern inconsistency and camouflage. To address these issues, this paper proposes an adaptive algorithm called AdaptFD to effectively detect fraudulent users on e-commerce platforms. In particular, we first tackle sparsity issue with finer granularity by inferring relevant coefficients on entity level during runtime. Next, variable regularization terms are designed for pattern inconsistency tolerance. Lastly, we analyze the behavior patterns of users to reveal camouflage of fraudsters. Together, AdaptFD can adapt to different scenarios and handle with large-scale data efficiently. We also provide theoretical guarantees for our algorithm in linear time complexity and training convergence. Extensive experiments on four real-world datasets demonstrate that AdaptFD outperforms other state-of-the-art baselines. It's worth noticing that our algorithm achieves up to 560x speedup on the real-world dataset, compared with the most related baseline.

1 Introduction

Fraud detection has attracted lots of interests from both academic and industrial community due to its extensive and practical applications in many areas such as finance [4, 32] and review management [5, 23, 29]. Numerous approaches have been developed for fraud detection on e-commerce platforms like Amazon, eBay and TripAdvisor. Graph-based methods [14, 27, 31] have shown promising results, which utilize the interactions between users and items to construct a bipartite graph. Then, graph-based models detect suspicious nodes or edges by exploiting the topological structures of graphs, based on the assumption that fraudsters have different interaction patterns from normal users in the same graph.

Although the problem of fraud detection on e-commerce websites has been well-investigated by existing graph-based approaches, they still cannot perform consistently well across different graphs due to some existing challenges. Firstly, existing graph-based detectors suffer from class imbalance issue on mainstream online trading websites, where fraudsters only account for a small proportion [17]. In particular, only 0.09% - 0.66% of users are fraudulent as observed from four public real-world datasets collected from Amazon.

* Corresponding Author. Email: shaowen.tang@unsw.edu.

Moreover, the performance of graph-based methods rely on massive interactions among entities in the graph, providing sufficient information to reflect the behavior patterns of users. However, the sparsity issue is ubiquitous in real-world scenarios, as many entities only have a few interactions with others, which limits the performance of graph-based detectors [24]. For instance, a newly registered user might be classified as fraudulent by existing fraud detection systems because of the first rating, which might be opposite to the mainstream. However, it still could be a normal user with an unique opinion on a specific product.

Besides, fraudsters often camouflage their malicious behaviors to avoid being detected by anti-fraud systems [6, 10, 25]. In general, spammers intentionally interact with popular items to establish strong connections with normal users for alleviating suspiciousness. Lastly, benign users do not always share the same behavior pattern with other normal users due to different preferences. Thus, effective fraud detectors are supposed to be inconsistency-tolerated [18].

To address the above issues, in this paper, we propose a novel HITS-based Adaptive Fraud Detection Scoring Algorithm (AdaptFD for short) to effectively identify fraudsters on e-commerce platforms. Specifically, AdaptFD also formulates user-item interaction networks as bipartite graphs and analyze the topological structures of graphs as other graph-based methods did. However, AdaptFD offers finer granularity by dynamically tuning related coefficients for each entity and edge in the graph during the inferring process instead of handling with them equally. As a result, AdaptFD provides not only better predictions on one specific graph, but also better adaptability to deal with different graphs regardless of size, the extent of sparsity and class imbalance. In this way, AdaptFD achieves up to 560x speedup on real-world datasets used for evaluation, compared with the most related baseline [14]. Besides, flexible regularization terms are integrated into AdaptFD, which reflect the overall behavior pattern of each user, controlling the sensitivity of AdaptFD on pattern inconsistency. We also analyze the distribution of targets under different kinds of interactions from both normal and fraudulent users to reveal the camouflage behaviors from fraudsters, which guides AdaptFD to shift focus on discernible behavior patterns for identifying fraudsters under camouflage.

We highlight the contributions of our work as follows:

- We propose a novel algorithm called AdaptFD to detector frauds on e-commerce platforms, which provides adaptability to scenarios in different settings by addressing sparsity, pattern inconsistency and camouflage issues with finer granularity.
- We provide theoretical guarantees for AdaptFD in terms of linear scalability and training convergence, which indicates AdaptFD is scalable for large-scale data in practice.
- We conduct extensive experiments on four real-world datasets

with different properties to show the effectiveness of AdaptFD, which outperforms all other baselines.

Our code and appendix material is available at https://www.cse.unsw.edu.au/~wong/ECAI2024/.

2 Related Work

The following summarizes the related work in fraud detection and, in particular, graph-based fraud detection.

2.1 Generic Fraud Detection

To detect frauds in user-product review platforms, [11] is the first work to extract handcraft features corresponding to users, products and reviews such as review length and average rating given by user, and learn suitable weights for features using a logistic regression classifier. Inspired by this work, Behavior [16] expands feature set to detect target-based spamming. Co-training [30] is a semi-supervised method, training two independent models simultaneously with different feature sets regarding to reviews and users respectively. In this way, two models boost the performance of each other iteratively by mutually annotating unseen data as new training samples. In addition, Deceptive [22] and [24] focus on textual content analysis to find out frauds. [33] leverages temporal pattern discovery to tackle singleton review issue. Birdnest [9] and SpEagle [23] distinguish normal and suspicious users by examining the difference between their behavior patterns and the global patterns.

However, the above approaches lack scalability due to their massive feature generation and collection process. Besides, they disregarded topological information between users and items, which limits the performance. Different from them, we leverage topological structures to build a more effective and scalable fraud detector.

2.2 Graph-based Fraud Detection

Since PageRank [3] and HITS [13] were introduced, many researches in different fields started to explore structural relations among nodes in graph-like data rather than contextual information, such as opinion spam detection [6, 29], recommender systems [34] and financial fraud detection [35]. For fraud detection, BAP [20] iteratively computes the bias and prestige of each node, based on the idea that opinion should weigh significantly even from biased users if it is in unusual pattern. Similarly, Troll-Trust [31] normalizes measurements with sigmoid-like functions and introduces variable Laplacian smoothing terms for addressing cold start issue, while it fails to handle with discrete values, i.e., most of ratings across e-commerce websites. Eagle [1] uses loopy belief propagation to update the entire graph iteratively with prior knowledge. Based on HITS algorithm [13], Trustiness [27, 28] and Rev2 [14] formulate graphs with more designed metrics to mathematically examine each entity and edge in the graph. More recently, Graph Neural Networks (GNNs) like GCN [12] and GAT [26] have been extensively investigated by many studies [6, 7, 17, 25] to learn low-dimensional vector representation for nodes in the graph by aggregating information from neighbors.

However, the above graph-based methods either did not consider sparsity or they only addressed the issue on graph level. Hence, they are not adaptive to different graphs with different properties. On the contrast, this paper addresses sparsity on entity level so that the proposed AdaptFD can perform consistently well across different graphs without any need for prior knowledge.

3 Problem Formulation

In this section, we present preliminaries needed to formally define the problem of graph-based fraud detection on e-commerce platforms. Then, we present three essential metrics for tackling the problem.

3.1 Preliminaries

We define a user-item rating network as a bipartite graph $\mathcal{G} = (\mathcal{U}, \mathcal{P}, \mathcal{R})$, consisting of a set of user nodes $\mathcal{U} = \{u_1, u_2, ..., u_m\}$, a set of item nodes $\mathcal{P} = \{p_1, p_2, ..., p_n\}$, where m is the total number of unique user entities and n denotes the amount of unique items. $\mathcal{R}$ denotes the group of interactions from user nodes to item nodes, where each edge $r_{i,j} \in \mathcal{R}$ represents the interaction from u_i to p_j. For simplicity, we use $\mathcal{R}_{i,*}$ to denote all ratings given by user u_i and $\mathcal{R}_{*,j}$ to denote all ratings received by item p_j. The weight of edge $r_{i,j}$ is denoted as $w(r_{i,j})$. For generalization, we scale $w(\cdot)$ to the range between -1 and $+1$, i.e., $w(r) \in [-1, 1],\ \forall r \in \mathcal{R}$. With above preliminaries, we define the problem of graph-based fraud detection as following:

Definition 1 (Graph-based Fraud Detection Problem)**.** *Given a bipartite rating graph $\mathcal{G} = (\mathcal{U}, \mathcal{P}, \mathcal{R})$, what is the probability $P(u_i)$ of user node u_i to be a fraudster, whose rating behavior is maliciously deviated from normal users, where $1 \leq i \leq m$?*

3.2 Metrics

To address this problem, we introduce three necessary metrics when modeling user-item rating networks as bipartite graphs: (1) quality of item; (2) trustworthiness of rating; (3) fairness of user. The definitions of these three metrics will be explained as below:

Quality measures the inherent goodness of items, which also can be regarded as the deserved rating from general users. We denote it as $Q(p) \in [-1, +1],\ \forall p \in \mathcal{P}$. Intuitively, high quality $Q(p)$ (close to $+1$) refers to high expected rating from fair users on item p, and vice versa.

Trustworthiness measures how trustworthy is a rating $r_{i,j}$ from user u_i on item p_j, denoted as $T(r)$ where $r \in \mathcal{R}$. The value of $T(\cdot)$ ranges from 0 to 1, as 0 and 1 stand for absolute unreliability and complete authenticity, respectively.

Fairness quantifies the reliability of users within the graph, denoted as $F(u) \in [0, 1],\ \forall u \in \mathcal{U}$. Intuitively, higher fairness score $F(u)$ indicates more honest user u, and vice versa.

4 Methodology

In this section, we first introduce the basic equations to compute the metrics presented in Section 3.2. Then, we extend the basic equations to address the issues described in Section 1. Finally, we introduce the proposed algorithm AdaptFD with all equations combined.

4.1 Quality

According to [14, 27], the quality of item is decided by the values and trustworthiness scores of received ratings. Specifically, more trustworthy positive ratings lead to a higher quality score. Thus, the quality of item is calculated as:

$$Q(p_j) = \frac{\sum_{r \in \mathcal{R}_{*,j}} T(r) \cdot w(r)}{\sum_{r \in \mathcal{R}_{*,j}} T(r)} \tag{1}$$

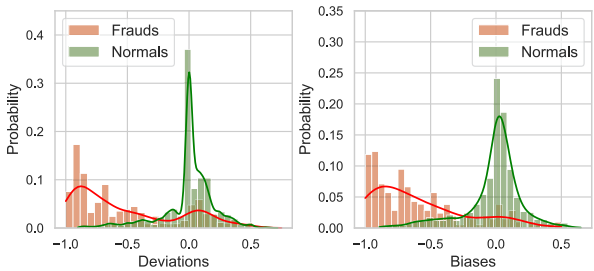

Figure 1. Behavior analysis of users: deviations and biases.

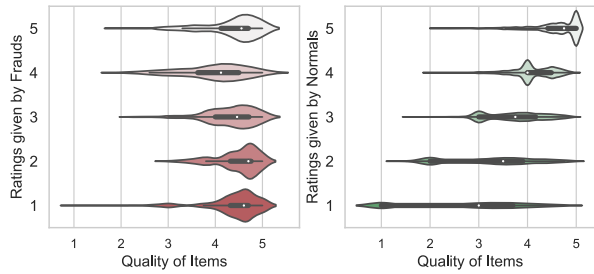

Figure 2. Distributions of targets under different ratings from users.

4.2 Trustworthiness

The key factor to decide the trustworthiness of a rating is the fairness of the corresponding user. In general, fair users are more likely to give reliable ratings, while fraudsters prefer to give untrustworthy ratings. The trustworthiness of a rating is also dependent on how it deviates from the intrinsic quality of its target. Formally, we use $D(r)$ to denote the deviation of rating r, computed as:

$$D(r_{i,j}) = \frac{w(r_{i,j}) - Q(p_j)}{z} \tag{2}$$

where z is a normalization coefficient to ensure $D(r) \in [-1, 1]$, $\forall r \in \mathcal{R}$. As shown in Fig. 1, in a real-world rating network from Amazon, normal users tend to rate items close to what targets deserve, while fraudsters most likely rate items with unexpectedly low values. Hence, extreme deviations intuitively imply untrustworthy ratings. Then, we infer the trustworthiness of rating as:

$$T(r_{i,j}) = \frac{\gamma_1 \cdot F(u_i) + \gamma_2 \cdot \tilde{D}(r_{i,j})}{\gamma_1 + \gamma_2} \tag{3}$$

where $\tilde{D}(\cdot) = 1 - \|D(\cdot)\|$ and γ_1, γ_2 are non-negative coefficients to control the importance of two factors in the calculation of trustworthiness respectively.

4.3 Fairness

The fairness of a user will depend on all its given ratings with respect to the trustworthiness scores. Intuitively, fair users are more likely to give trustworthy ratings than fraudsters. Thus, we develop the equation for computing the fairness of user as:

$$F(u_i) = \sum_{r \in \mathcal{R}_{i,*}} T(r) \;/\; |\mathcal{R}_{i,*}| \tag{4}$$

where $|\cdot|$ is element count operator. Finally, we infer the probability $P(u)$ of a user u being fraudulent based on corresponding fairness:

$$P(u_i) = 1 - F(u_i) \tag{5}$$

4.4 Alleviate Inconsistency

The inconsistency problem is caused by active users who occasionally give unusual ratings due to specific reasons such as unique personal preference [18]. Therefore, we include the overall behavior pattern of users to smooth the impacts of few unusual interactions when computing the trustworthiness for ratings. We use $B(u)$ to denote the bias of user u, reflecting the pattern of giving higher or lower ratings than what items deserve, inferred as:

$$B(u_i) = \frac{\sum_{r_{i,j} \in \mathcal{R}_{i,*}} w(r_{i,j}) - Q(p_j)}{|\mathcal{R}_{i,*}| \;\cdot\; z} \tag{6}$$

where $B(\cdot)$ is normalized into the same interval $[-1, 1]$ as $D(\cdot)$ using the same normalization coefficient z. As we can see from Fig. 1, normal users are more neutral than anomalies in terms of bias score, which is consistent with our analysis. Then, we revise Eq. (3) as:

$$T(r_{i,j}) = \frac{\gamma_1 \cdot F(u_i) + \gamma_2 \cdot \tilde{D}(r_{i,j}) + \gamma_3 \cdot \tilde{B}(u_i)}{\gamma_1 + \gamma_2 + \gamma_3}, \; r_{i,j} \in \mathcal{R} \tag{7}$$

where $\tilde{B}(\cdot) = 1 - \|B(\cdot)\|$ and γ_3 is also a non-negative coefficient to decide the regularization intensity of overall behavior pattern on the calculation for trustworthiness.

4.5 Uncover Camouflage

To avoid being identified by anti-fraud systems in real world, smart fraudsters camouflage themselves by intentionally giving reasonable ratings to popular items and building up strong connections with benign users to reduce their overall suspiciousness [6]. We explore the correlations between the distributions of ratings and the targeted items to uncover camouflage behaviors. Observed from Fig. 2, we have several insightful discoveries: (1) most of positive ratings from spammers are given to high-quality items, the same as normal users, which could be the evidence of camouflage; (2) most of negative ratings from fraudsters target at high-quality items, which is quite different from normal users, who prefer to give low ratings on items with medium quality or below; (3) the lower the ratings, the more significant rating deviations can be perceived from fraudsters than benign users. Therefore, we adjust the significance of each rating according to its value when computing the fairness of user in Eq. (4) to alleviate the effects of camouflage behaviors:

$$F(u_i) = \sum_{r \in \mathcal{R}_{i,*}} T(r) \cdot S(r) \;/\; \sum_{r \in \mathcal{R}_{i,*}} S(r) \tag{8}$$

where $S(r)$ indicates the significance of each rating r on deciding the fairness of its corresponding user u. In this case, we calculate it as $S(r) = (3 - w(r)) \;/\; 2$, so that the lowest ratings have just twice the significance of the highest ratings.

4.6 Address Sparsity

Our equations for inferring true values of essential metrics work properly when every entity has adequate interactions in the graph. While on real-world platforms, the performance of fraud detection systems suffer from sparsity problem [21, 33]. For example, it is hard for existing detectors to classify users with only few ratings, which could either be inactive normal users or malicious users for executing planned bursty spamming [2]. Similarly for item nodes, it is difficult to infer the real quality of items when they only receive few or unreliable ratings. Thus, effective fraud detectors in real world

are supposed to address the sparsity issue for improving customer experience and increasing revenue [15].

In Rev2 [14], they attempt to address this issue on graph level. Specifically, for a single rating graph, they train multiple independent models with different settings. The final prediction is based on the outputs of all model. However, it is not fine-grained to assign the same parameters on the graph in individual model, since different entities might be in different contextual situations even within the same graph. Although this ensemble strategy improves the stability and capability of managing different graphs to some extents, it actually sacrifices the scalability to large datasets due to the significantly increasing computational cost.

In this work, we aim to address the sparsity issue with entity-level granularity, which is more adaptive and efficient when handling with different scenarios. For the quality of items, in case they don't receive sufficient trustworthy ratings, a variable Laplacian smoothing term is added to Eq. (1) as:

$$Q(p_j) = \frac{\sum_{r \in \mathcal{R}_{*,j}} T(r) \cdot w(r) + \alpha \cdot q}{|\mathcal{R}_{*,j}|} \tag{9}$$

where q denotes the default quality value, which could be the mean of all quality values in the graph. Term α is calculated during runtime as $\alpha = \sum_{r \in \mathcal{R}_{*,j}} (1 - T(r))$, dynamically controlling the proportion of default. For each item in the graph, the more trustworthy ratings received, the less dependency on the default. Likewise, when inferring the trustworthiness of ratings, we dynamically tune coefficients $\gamma_1, \gamma_2, \gamma_3$ in training process with respect to the interaction intensity level of the corresponding user as following:

$$\gamma_1 = \min(1.0,\ |\mathcal{R}_{i,*}| \ / \ \sigma),\ 1 \le i \le m$$
$$\gamma_2 = 1.0$$
$$\gamma_3 = \gamma_1 \cdot \lambda$$

where parameter σ is a positive integer as the threshold value of intensive interactions for users and parameter λ is a positive number to control the regularization intensity of inconsistency-tolerance term. As a result, users with more than σ interactions are considered as active users and will be treated equally. Besides, the coefficient γ_3 of bias term is mutually decided by both parameters σ and λ, which only takes perceptible effect when the corresponding user has enough interactions to reflect the overall behavior pattern.

Moreover, rating deviation as the key factor to infer the trustworthiness of ratings, could be inaccurate when the corresponding item doesn't receive adequate reliable ratings and the quality of it is mainly decided by the default. Hence, we further revise Eq. (7) as:

$$T(r_{i,j}) = (1 - \beta) \cdot \frac{\gamma_1 \cdot F(u_i) + \gamma_2 \cdot \tilde{D}(r_{i,j}) + \gamma_3 \cdot \tilde{B}(u_i)}{\gamma_1 + \gamma_2 + \gamma_3} + \beta \cdot t \tag{10}$$

where t refers to the default trustworthiness value in the graph and β is also a flexible smoothing term to control the importance of default based on all ratings received by the corresponding item, inferred as:

$$\beta = \sum_{r \in \mathcal{R}_{*,j}} (1 - T(r)) \ / \ |\mathcal{R}_{*,j}| \tag{11}$$

Lastly, we add a fixed smoothing term in Eq. (8) to cope with sparsity issue as following:

$$F(u_i) = \Big(\sum_{r \in \mathcal{R}_{i,*}} T(r) \cdot S(r) + f \Big) \ / \ \Big(\sum_{r \in \mathcal{R}_{i,*}} S(r) + 1 \Big) \tag{12}$$

where f represents the default fairness value in the graph. All default values can be updated during training process, so it does not matter which values should be selected as initialization.

Algorithm 1 The AdaptFD Algorithm

Input Bipartite rating network $\mathcal{G} = (\mathcal{U}, \mathcal{P}, \mathcal{R})$
Input: Parameters σ and λ
Input: Training epochs $epochs$, error threshold ϵ
Output: Probability $P(u)$ for each user entity being fraudulent

Randomly initialization default values for each metric i.e., q, t, f
Let $i = 0$
repeat
  $i := i + 1$
  Compute $Q^i(p),\ \forall p \in \mathcal{P}$ using Eq. (9) and update q
  Compute $T^i(r),\ \forall r \in \mathcal{R}$ using Eq. (10) and update t
  Compute $F^i(u),\ \forall u \in \mathcal{U}$ using Eq. (12) and Update f
  $error := \sum \|F^i(u) - F^{i-1}(u)\| \ / \ |\mathcal{U}|,\ \forall u \in \mathcal{U}$
until $i == epochs$ Or $error < \epsilon$
return $P(u),\ \forall u \in \mathcal{U}$, computed by Eq. (5)

4.7 The AdaptFD Algorithm

By combining all the equations above (Section 4.1 to Section 4.6), the proposed algorithm AdaptFD is presented in this section. Specifically, AdaptFD iteratively computes values of metrics for all entities and edges within the bipartite rating graph as shown in Algorithm 1, where superscript i and $i-1$ indicate the current and last iteration of the training process respectively.

4.7.1 Time Complexity Analysis

In each iteration of training, the proposed AdaptFD goes through all user nodes twice, all items nodes twice and all ratings four times for updating values of metrics. Hence the time complexity of each iteration is denoted as $\mathcal{O}(2m + 2n + 4l)$, where m, n, l represent the total number of users, items and ratings respectively. Let, k be the number of training iterations required for AdaptFD to converge. Then, the overall time complexity for AdaptFD is $\mathcal{O}(k * (2m + 2n + 4l))$, where k is a constant. In conclusion, AdaptFD has a linear time complexity, which is scalable to large-scale datasets in practice.

4.7.2 Error Bound Analysis

The error between the calculated value for the metric at any iteration and the true value is bounded, as shown by the theorem below. The proofs of the below lemmas and theorem can be found in the Appendix. Let, $Q^\infty(p)$ and $F^\infty(u)$ represent the true quality value of item p and true fairness value of user u, respectively.

Lemma 1. *In any two iterations i and j, the difference between the computed quality values for any item p is at most 1, i.e. $\|Q^i(p) - Q^j(p)\| \le 1$.*

Lemma 2. *Similarly, in any two iterations i and j, the difference of the computed trustworthiness values for any rating r is at most $\frac{1+z}{2z}$; the difference of the computed fairness values for any user u is at most $\frac{1+z}{2z}$.*

Theorem 3. *The difference between the true fairness of user u and calculated value at any iteration i is bounded, i.e. $\|F^\infty(u) - F^i(u)\| \le (\frac{1+z}{2z})^i$. Similarly, the difference between the true quality of item p and calculated value at any iteration i is bounded, i.e. $\|Q^\infty(p) - Q^i(p)\| \le (\frac{1+z}{2z})^{i-1}$; and the difference between the true trustworthiness of rating r and calculated value at any iteration i is bounded, i.e. $\|T^\infty(r) - T^i(r)\| \le (\frac{1+z}{2z})^i$.*

Table 1. Statistics for the four real-world datasets.

Datasets	# Users (% Fraud)	# Items	# Ratings	Density
FineFoods	256,059 (0.09%)	74,258	568,454	2.22
Instruments	339,231 (0.12%)	83,046	500,176	1.47
Electronics	192,403 (0.27%)	63,001	1,689,188	8.78
Beauty	22,363 (0.66%)	12,101	198,502	8.88

4.7.3 Training Convergence Analysis

After analyzing error bound for metrics within the graph during training process, we now analyze the convergence behavior of the proposed AdaptFD. Given an error threshold ϵ, the number of training iterations needed for convergence k should satisfy $(\frac{1+z}{2z})^{k-1} < \epsilon$. Theoretically, the maximum number of training iterations k required for AdaptFD to converge is:

$$k = \lceil \log_{(\frac{1+z}{2z})}(\epsilon) + 1 \rceil$$

5 Experiments

In this section, we conduct extensive experiments on real-world datasets to show the effectiveness of the proposed AdaptFD. In particular, we aim to answer the following research questions:

- **RQ1.** Does AdaptFD effectively identify frauds across different rating networks compared with the state-of-the-art baselines?
- **RQ2.** How do different designed components of AdaptFD contribute to the overall performance?
- **RQ3.** How does the performance of AdaptFD change under different parameter settings?
- **RQ4.** Is AdaptFD scalable to large datasets in real world?

5.1 Experiment Setup

5.1.1 Datasets

To validate the effectiveness of AdaptFD, we conduct experiments on four real-world datasets [8, 19] **FineFoods**, **Instruments**, **Electronics** and **Beauty**, collected from Amazon from different categories with different properties in terms of size, extent of sparsity and class imbalance. The detailed information of datasets is shown in Table 1, where the column "Density" indicates the average number of ratings given by users. As we can observe from the statistics, class imbalance is prevalent in all datasets. Besides, we include both sparse and dense datasets to examine the adaptability of AdaptFD and other baselines. These original datasets do not provide ground truth for fraud detection. Hence following by [14], we use helpfulness votes to generate labels for test.

5.1.2 Baselines

We compare AdaptFD with five state-of-the-art baselines to show the effectiveness of the proposed algorithm for fraud detection on e-commerce platforms. For fair comparison, we convert all baselines to calculate the probability of each user being fraudulent. **Birdnest** [9] only considers distributions and timestamps of ratings, utilizing Bayesian inference to examine the deviation between the estimated distribution and posterior distribution. **BAP** [20] measures the bias of users and the prestige of items, based on the idea that ratings even from biased users should be weighed significantly if the ratings show unusual pattern. **Eagle** [1] regards fraud detection as a a node classification problem in signed graphs with prior knowledge, and uses loopy belief propagation to update the entire graph iteratively. **Behavior** [16] extracts multiple features relevant to rating behaviors and combine them together to calculate the anomaly scores for users. **Rev2** [14] is based on HITS [13] algorithm and iteratively computes three intrinsic metrics in the bipartite rating graph until convergence.

5.1.3 Evaluation Metrics

As shown in Table 1, all datasets are extremely imbalanced in terms of the ratio of fraudulent users to normal users. Thus, we select *Area Under the ROC Curve (AUC)* as one of evaluation metrics, which is widely used in imbalanced data classification problems. We also choose three widely-used evaluation metrics in ranking systems *Average Precision (AP)*, *Normalized Discounted Cumulative Gain (NDCG)* and *Precision@K* to evaluate the capabilities of different models on separating malicious users from massive normal users.

5.1.4 Implementation Details

For BAP and Rev2, we follow the instructions from their original papers to implement algorithms with reported parameters if available. For Birdnest, we use the source codes provided by authors to run experiments. For Eagle and Behavior, we carry out available open-source implementations online. As for the proposed AdaptFD, parameters are selected and tuned on validation set, as σ is chosen from $\{ 5, 8, 11, 14, 17 \}$ and λ is chosen from $\{ 0.1, 0.3, 0.5, 0.7, 0.9 \}$. We set training epochs as 100 and the raining error threshold $\epsilon = 10^{-4}$ as many other works did. All models are running on Python3.9.16, 16GB RAM, 12th Gen Intel Core i7-12700 CPU.

5.2 Performance Comparison (RQ1)

5.2.1 Overall Evaluation

To answer the RQ1, we evaluate the performance of the proposed AdaptFD and all the compared methods on identifying frauds on across four real-world datasets as shown in Table 2. We can obtain the following observations from the experiment results.

Firstly, AdaptFD outperforms all baselines in terms of all evaluation metrics on four datasets. Even though class imbalance and sparsity issues exist in all datasets to different extents, AdaptFD still boosts performance significantly, especially in *AP* value by 4.80%, 2.99%, 4.76% and 10.14% respectively. This implies that AdaptFD is not only capable of effectively identifying fraudsters in one specific graph, but also adaptive to different scenarios with only topological structure provided.

Secondly, Birdnest has poor results in all metrics across all datasets, mainly because it only focuses on rating patterns of users and ignore interactions between users and items. Although Behavior achieves promising results by relying on behavior- and text-based features extraction and analysis, AdaptFD still shows better performance, especially on dense datasets Electronics and Beauty, by 2.71% and 4.09% in AUC value, respectively. The experiment results validate the superiority of analyzing structural information for fraud detection on rating networks, especially when interconnections in the graph are relatively dense.

Thirdly, graph-based fraud detectors, like BAP, Eagle and Rev2, also attempt to exploit transmitted information upon interactions as what AdaptFD does. However, BAP and Eagle fail to deal with graph sparsity problem, which leads to worse performance than Rev2 and AdaptFD, especially across sparse datasets. The reason of Eagle achieving higher *AUC* values than other baselines on dense graph

Table 2. Overall performance comparison of the proposed AdaptFD with five state-of-the-art baselines on four real-world datasets. The best of each evaluation metric is highlighted in bold. The second best result of each metric is underlined.

Dataset	FineFoods			Instruments			Electronics			Beauty		
Metric	AP	AUC	NDCG	AP	AUC	NDCG	AP	AUC	NDCG	AP	AUC	NDCG
Birdnest	0.1158	0.5568	0.6479	0.1098	0.5253	0.6134	0.0684	0.4806	0.6287	0.0497	0.4526	0.5336
BAP	0.4309	0.8348	0.8290	0.6399	0.8846	0.9251	0.2971	0.7144	0.8171	0.2105	0.6484	0.7344
Eagle	0.3828	0.8275	0.8345	0.4865	0.8986	0.8639	0.2772	0.8246	0.7926	0.2185	0.7582	0.7216
Behavior	0.4466	0.8566	0.8452	0.6621	0.9139	0.9327	0.3568	0.8033	0.8433	0.2565	0.7462	0.7617
Rev2	0.4623	0.8478	0.8558	0.6584	0.9060	0.9311	0.3456	0.7985	0.8393	0.2406	0.7067	0.7525
AdaptFD$_{\setminus S}$	0.4407	0.8452	0.8405	0.6528	0.9023	0.9294	0.3601	0.8042	0.8445	0.2544	0.7101	0.7589
AdaptFD$_{\setminus I}$	0.4833	0.8683	0.8558	0.6789	0.9257	0.9371	0.3715	0.8126	0.8487	0.2743	0.7672	0.7703
AdaptFD$_{\setminus C}$	0.4830	0.8704	0.8547	0.6824	0.9286	0.9383	0.3588	0.8183	0.8447	0.2781	0.7768	0.7763
AdaptFD	**0.4845**	**0.8704**	**0.8560**	**0.6819**	**0.9288**	**0.9381**	**0.3738**	**0.8251**	**0.8502**	**0.2825**	**0.7767**	**0.7762**

Electronics and Beauty is that it sacrifices true positive rate in exchange of low false positive rate, which is not preferred in practice. Although Rev2 tries to alleviate impacts of sparsity issue on graph level by running different parameter combinations on the same rating network, it actually becomes very computationally expensive to train many similar independent models. This leads to the lack of scalability compared with the proposed AdaptFD, which addresses graph sparsity on entity level. As a result, AdaptFD achieves remarkable improvements even on sparse datasets FineFoods and Instruments, by 4.80% and 3.57% in AP value, respectively.

5.2.2 *Ranking Evaluation*

In practice, effective fraud detection systems are supposed to separate fraudsters from massive normal users. The motivation is to restrict suspicious activities from potential fraudsters in time and reduce the chance to bother normal users for identification verification [15]. Therefore, we further evaluate the AdaptFD against all baselines in terms of capability of distinguishing fraudulent and normal users on two representative datasets, as shown in Table 3.

As we can observe from the experiment results, AdaptFD identifies 64 and 65 real fraudsters among predicted top 100 suspicious users on two datasets respectively, which both are the best compared with all baselines. By addressing graph sparsity issue on entity level, AdaptFD improves the performance under both 20% and 80% recalls. Notably, AdaptFD outperforms the second best model in *AP@20%Recall* by 6.94% on FineFoods, indicating the better ability to filter out obvious fraudsters. Likewise, AdaptFD has the best *Precision@80%Recall* result led by 10.80% on FineFoods, which shows the effectiveness of the proposed algorithm on separating fraudulent users from massive normal users. Although the graph sparsity issue is not prevalent in dense rating networks, AdaptFD still outperforms other baselines due to the inconsistency-tolerance and anti-camouflage modules, which help to improve ranking performance by 2.77% and 4.18% on Electronics in *AP* value at 20% and 80% recalls, respectively.

In summary, by analyzing the experiment results, AdaptFD outperforms the state-of-the-art baselines with respect to all our evaluations on different real-world datasets, which validates the effectiveness and adaptability of AdaptFD.

5.3 *Ablation Study (RQ2)*

To answer the RQ2, we identify three key components of AdaptFD for addressing graph sparsity (S), tolerating inconsistency (I) and alleviating camouflage (C). Then, we construct three ablation models

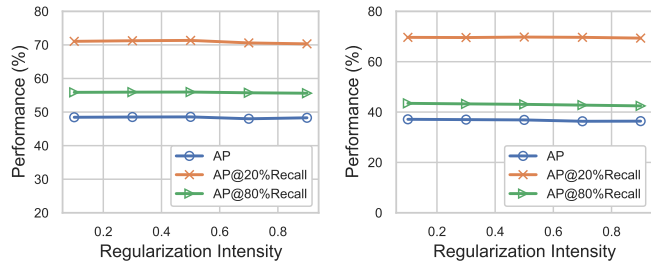

Figure 3. Sensitivity analysis on parameter λ (FineFoods & Electronics).

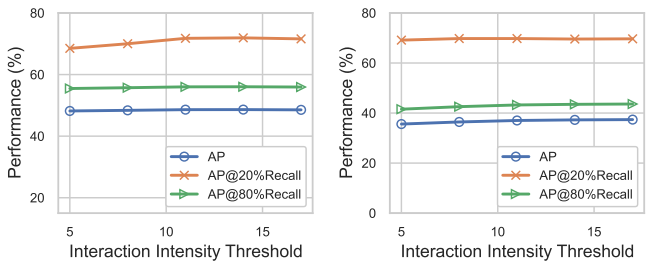

Figure 4. Sensitivity analysis on parameter σ (FineFoods & Electronics).

by removing each of component respectively to validate each of their contributions to the overall performance of AdaptFD. The experiment results are shown in Table 2.

As we can see from the results, by removing the module for addressing graph sparsity (AdaptFD$_{\setminus S}$), the performance decreases on all datasets (more obviously in AP). This result suggests that the sparsity issue exists in all rating networks. In particular, AdaptFD$_{\setminus S}$ is consistently outperformed by other two ablation models on all datasets, demonstrating the importance and necessary of component (S) in AdaptFD to tackle sparsity issue.

By removing the inconsistency-tolerance module (AdaptFD$_{\setminus I}$), the performance of AdaptFD drops marginally on sparse datasets than dense datasets. This is because inactive nodes cannot provide sufficient interactions to reflect their behavior patterns with AdaptFD for avoiding overfitting. On the contrast, the performance drops more on dense datasets by removing the anti-camouflage component (AdaptFD$_{\setminus C}$) than on sparse datasets.

In summary, experiment results show the module (S) is critical to AdaptFD's performance, since the sparsity issue is prevalent in all rating networks. The (I) and (C) modules still occasionally improve the overall performance.

Table 3. Ranking performance evaluation (%) with baselines on two representative real-world datasets FineFoods (sparse) and Electronics(dense), where each model ranks users based on computed probabilities of being fraudsters. The best of each metric is in bold.

Dataset	Metric	Baselines					Ours
		Birdnest	BAP	Eagle	Behavior	Rev2	AdaptFD
FineFoods	Precision@100	12	63	46	62	62	**64**
	Precision@20%Recall	13.37	64.00	46.15	64.86	68.57	**70.59**
	AP@20%Recall	14.70	62.45	54.48	63.49	66.82	**71.46**
	Precision@80%Recall	9.65	26.30	29.27	27.83	27.23	**32.43**
	AP@80%Recall	12.11	50.17	44.39	51.57	53.78	**55.88**
Electronics	Precision@100	1	64	44	62	61	**65**
	Precision@20%Recall	5.70	51.98	37.63	50.48	48.39	**52.50**
	AP@20%Recall	5.06	67.79	41.71	67.50	66.14	**69.67**
	Precision@80%Recall	7.80	10.04	17.31	15.32	14.95	**17.40**
	AP@80%Recall	6.54	35.01	31.21	41.86	40.47	**43.61**

5.4 *Parameter Sensitivity (RQ3)*

To answer the RQ3, we explore the sensitivity of AdaptFD with respect to the parameters λ and σ, which control the regularization intensity and the threshold for intensive interactions. When varying the value of each parameter, we keep another constant, then record the variations of performance on two representative datasets as shown in Fig. 3 and Fig. 4. We have several observations from the results.

Firstly, from Fig. 3, as the parameter λ increases from 0.1 to 0.9 with step size 0.2, the performance of AdaptFD on sparse dataset FineFoods remains stable. This suggests that AdaptFD is insensitive to parameter λ on sparse rating networks due to the marginal effects of inconsistency-tolerance module when the entities are inactive. While on dense dataset Electronics, the performance of AdaptFD reaches its best when $\lambda = 0.1$ and starts to decline as λ increases.

Likewise from Fig. 4, as the value of parameter σ grows, AdaptFD is actually more sensitive to the interaction intensity. As a result, the performance on both datasets tend to have stable gains as we gradually increase σ until reaching the peak. This is more obvious on sparse dataset FineFoods, where the *AP@20%Recall* value is improved by around 3.08% when $\sigma = 14$ compared with $\sigma = 5$.

In summary, the performance of AdaptFD is generally stable when changing parameters λ and σ. We also observe that AdaptFD is more sensitive to parameter λ on dense graphs and to parameter σ on sparse graphs.

5.5 *Scalability (RQ4)*

To answer the RQ4, we generate training samples in different magnitudes of amount from 10^3 to 10^7 each time, then we record the training time of AdaptFD with generated samples, shown as blue points in Fig. 5. As we see from the figure, the time cost is increasing linearly with respect to the growing number of samples. Additionally, we record the training time cost of AdaptFD on the real-world datasets, marked as red circles in Fig. 5. The results basically comply with our analysis about linear scalability, despite of some fluctuations due to different extents of graph sparsity.

Moreover, we compare the training time cost of AdaptFD on dataset FineFoods with other baselines as shown in Fig. 5. From the result, we can see AdaptFD is slightly slower than Birdnest, BAP and Behavior when handling with real-world dataset. However, the time cost of AdaptFD is still on the same order of magnitude with the fastest method, even though AdaptFD incorporates more functional modules to achieve better predictions. Rev2 is the only one in baselines that considers graph sparsity problem and tries to alleviate it on graph level by exhausting different parameter combinations to obtain better adaptability. Nevertheless, Rev2 become more computationally expensive and less scalable. Compare with Rev2, AdaptFD achieves a 560x acceleration by adapting variable coefficients for each node and edge during the runtime.

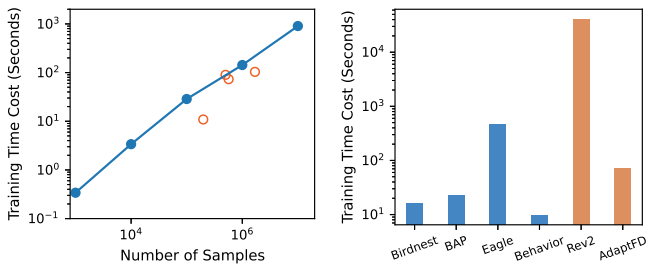

Figure 5. (Left) The training time cost of AdaptFD under different number of samples. (Right) The training time of all methods on the same dataset FineFoods. Blue bars represent models ignoring the sparsity issue, while brown bars take that into consideration.

In summary, experiment results show AdaptFD achieves linear scalability and is able to cope with different rating networks well.

6 Conclusion

We have proposed AdaptFD for detecting frauds on e-commerce platforms, which focuses on tackling three existing challenges in real world; i.e., graph sparsity, behavior inconsistency and fraud camouflage. Extensive experiments on four public real-world datasets have shown that AdaptFD consistently outperforms other state-of-the-art baselines in multiple evaluations. In particular, by considering sparsity issue on entity level, AdaptFD is more effective and adaptive when handling different graphs. Additionally, AdaptFD achieves two degrees of magnitude faster (up to 560x) on real-world datasets compared with the baseline that uses ensemble strategy, indicating the better scalability of AdaptFD on large-scale data in practice.

The proposed AdaptFD doesn't include all important elements of interactions on e-commerce platforms such as textual and temporal information, which might limit the performance of AdaptFD in real-world scenarios. Specifically, many irrelevant advertisement-type interactions could be detected by analyzing special characters or keywords. Besides, planned spams are usually bursty and time-related, which can be identified by temporal pattern analysis. However, it remains challenging to maintain the tradeoff between effectiveness and efficiency of fraud detectors when considering more elements. Our future work is to include other elements such as textual content (e.g., user reviews) in AdaptFD.

References

[1] L. Akoglu, R. Chandy, and C. Faloutsos. Opinion fraud detection in online reviews by network effects. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 7, pages 2–11, 2013.

[2] A. Breuer, R. Eilat, and U. Weinsberg. Friend or faux: graph-based early detection of fake accounts on social networks. In *Proceedings of The Web Conference 2020*, pages 1287–1297, 2020.

[3] S. Brin and L. Page. The anatomy of a large-scale hypertextual web search engine. *Computer networks and ISDN systems*, 30(1-7):107–117, 1998.

[4] D. Cheng, S. Xiang, C. Shang, Y. Zhang, F. Yang, and L. Zhang. Spatio-temporal attention-based neural network for credit card fraud detection. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 362–369, 2020.

[5] S. Dhawan, S. C. R. Gangireddy, S. Kumar, and T. Chakraborty. Spotting collective behaviour of online frauds in customer reviews. *arXiv preprint arXiv:1905.13649*, 2019.

[6] Y. Dou, Z. Liu, L. Sun, Y. Deng, H. Peng, and P. S. Yu. Enhancing graph neural network-based fraud detectors against camouflaged fraudsters. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 315–324, 2020.

[7] Y. Gao, X. Wang, X. He, Z. Liu, H. Feng, and Y. Zhang. Alleviating structural distribution shift in graph anomaly detection. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 357–365, 2023.

[8] R. He and J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517, 2016.

[9] B. Hooi, N. Shah, A. Beutel, S. Günnemann, L. Akoglu, M. Kumar, D. Makhija, and C. Faloutsos. Birdnest: Bayesian inference for ratings-fraud detection. In *Proceedings of the 2016 SIAM International Conference on Data Mining*, pages 495–503. SIAM, 2016.

[10] B. Hooi, H. A. Song, A. Beutel, N. Shah, K. Shin, and C. Faloutsos. Fraudar: Bounding graph fraud in the face of camouflage. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 895–904, 2016.

[11] N. Jindal and B. Liu. Opinion spam and analysis. In *Proceedings of the 2008 international conference on web search and data mining*, pages 219–230, 2008.

[12] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[13] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM (JACM)*, 46(5):604–632, 1999.

[14] S. Kumar, B. Hooi, D. Makhija, M. Kumar, C. Faloutsos, and V. Subrahmanian. Rev2: Fraudulent user prediction in rating platforms. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 333–341, 2018.

[15] A. Li, Z. Qin, R. Liu, Y. Yang, and D. Li. Spam review detection with graph convolutional networks. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2703–2711, 2019.

[16] E.-P. Lim, V.-A. Nguyen, N. Jindal, B. Liu, and H. W. Lauw. Detecting product review spammers using rating behaviors. In *Proceedings of the 19th ACM international conference on Information and knowledge management*, pages 939–948, 2010.

[17] Y. Liu, X. Ao, Z. Qin, J. Chi, J. Feng, H. Yang, and Q. He. Pick and choose: a gnn-based imbalanced learning approach for fraud detection. In *Proceedings of the web conference 2021*, pages 3168–3177, 2021.

[18] Z. Liu, Y. Dou, P. S. Yu, Y. Deng, and H. Peng. Alleviating the inconsistency problem of applying graph neural network to fraud detection. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 1569–1572, 2020.

[19] J. J. McAuley and J. Leskovec. From amateurs to connoisseurs: modeling the evolution of user expertise through online reviews. In *Proceedings of the 22nd international conference on World Wide Web*, pages 897–908, 2013.

[20] A. Mishra and A. Bhattacharya. Finding the bias and prestige of nodes in networks based on trust scores. In *Proceedings of the 20th international conference on World wide web*, pages 567–576, 2011.

[21] A. Mukherjee, A. Kumar, B. Liu, J. Wang, M. Hsu, M. Castellanos, and R. Ghosh. Spotting opinion spammers using behavioral footprints. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 632–640, 2013.

[22] M. Ott, Y. Choi, C. Cardie, and J. T. Hancock. Finding deceptive opinion spam by any stretch of the imagination. *arXiv preprint arXiv:1107.4557*, 2011.

[23] S. Rayana and L. Akoglu. Collective opinion spam detection: Bridging review networks and metadata. In *Proceedings of the 21th acm sigkdd international conference on knowledge discovery and data mining*, pages 985–994, 2015.

[24] V. Sandulescu and M. Ester. Detecting singleton review spammers using semantic similarity. In *Proceedings of the 24th international conference on World Wide Web*, pages 971–976, 2015.

[25] F. Shi, Y. Cao, Y. Shang, Y. Zhou, C. Zhou, and J. Wu. H2-fdetector: A gnn-based fraud detector with homophilic and heterophilic connections. In *Proceedings of the ACM Web Conference 2022*, pages 1486–1494, 2022.

[26] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[27] G. Wang, S. Xie, B. Liu, and S. Y. Philip. Review graph based online store review spammer detection. In *2011 IEEE 11th international conference on data mining*, pages 1242–1247. IEEE, 2011.

[28] G. Wang, S. Xie, B. Liu, and P. S. Yu. Identify online store review spammers via social review graph. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 3(4):1–21, 2012.

[29] J. Wang, R. Wen, C. Wu, Y. Huang, and J. Xiong. Fdgars: Fraudster detection via graph convolutional networks in online app review system. In *Companion proceedings of the 2019 World Wide Web conference*, pages 310–316, 2019.

[30] J. Wu, L. Li, and W. Y. Wang. Reinforced co-training. *arXiv preprint arXiv:1804.06035*, 2018.

[31] Z. Wu, C. C. Aggarwal, and J. Sun. The troll-trust model for ranking in signed networks. In *Proceedings of the Ninth ACM international conference on Web Search and Data Mining*, pages 447–456, 2016.

[32] S. Xiang, M. Zhu, D. Cheng, E. Li, R. Zhao, Y. Ouyang, L. Chen, and Y. Zheng. Semi-supervised credit card fraud detection via attribute-driven graph representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 14557–14565, 2023.

[33] S. Xie, G. Wang, S. Lin, and P. S. Yu. Review spam detection via temporal pattern discovery. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 823–831, 2012.

[34] D. Zhang, Y. Zhu, Y. Dong, Y. Wang, W. Feng, E. Kharlamov, and J. Tang. Apegnn: Node-wise adaptive aggregation in gnns for recommendation. In *Proceedings of the ACM Web Conference 2023*, pages 759–769, 2023.

[35] Q. Zhong, Y. Liu, X. Ao, B. Hu, J. Feng, J. Tang, and Q. He. Financial defaulter detection on online credit payment via multi-view attributed heterogeneous information network. In *Proceedings of The Web Conference 2020*, pages 785–795, 2020.