Finding Needles in Heterogeneous Haystacks

Bijaya Adhikari[1], **Liangyue Li**[2], **Nikhil Rao**[2], **Karthik Subbian**[2]
[1]Department of Computer Science, University of Iowa.
[2]Amazon.
Email: bijaya-adhikari@uiowa.edu, [liliangy, nikhilsr, ksubbian]@amazon.com

Abstract

Due to intense competition and lack of real estate on the front page of large e-commerce platforms, sellers are sometimes motivated to garner non-genuine signals (clicks, add-to-carts, purchases) on their products, to make them appear more appealing to customers. This hurts customers' trust on the platform, and also hurts genuine sellers who sell their items without looking to game the system. While it is important to find the sellers and the buyers who are colluding to garner these non-genuine signals, doing so is highly non-trivial. Firstly, the set of bad actors in the system is a very small fraction of all the buyers/sellers on the platform. Secondly, bad actors "hide" with the good ones, making them hard to detect. In this paper, we develop CONGCN, a context aware heterogeneous graph convolutional network to detect bad actors on a large heterogeneous graph. While our method is motivated by abuse detection in e-commerce, the method is applicable to other areas such as computational biology and finance, where large heterogeneous graphs are pervasive, and the amount of labeled data is very limited. We train CONGCN via novel sampling methods, and context aware message passing in a semi-supervised fashion to predict dishonest buyers and sellers in e-commerce. Extensive experiments show that our method is effective, beating several baselines; generalizable to an inductive setting and highly scalable.

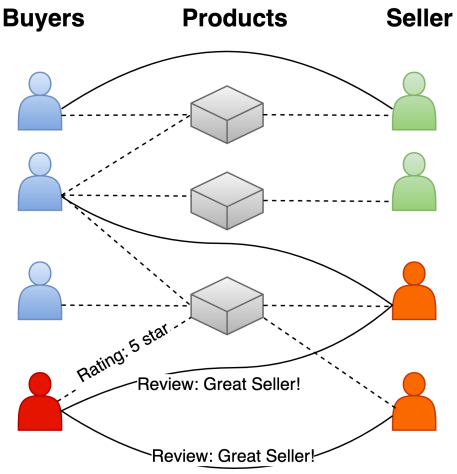


Figure 1: An example collusion occurring in e-commerce stores. Dishonest sellers (in orange) sell their products to a non-genuine buyer (in red), who in turn leaves fake 5-star rating for the product and great reviews for the sellers. The dashed edges represent purchase/sale relationship and the solid line represents buyers' feedback on sellers.

Introduction

Large online e-commerce stores such as Alibaba and Amazon allow merchants (sellers) to sell their products to a large numbers of customers (buyers). Buyers discover items they want to purchase via a search functionality on these platforms. These stores run large scale machine learning methods to rank items according to some notion of how relevant an item is for a customer query. A seller can maximize the sale of a product by ensuring the product ranks high in search results and garnering good reviews and ratings from the buyers. Higher search ranking adds product visibility, while good reviews and ratings help win customer trust, and eventually helps boost sales.

In rare cases, intense competition for real estate in the front page of e-commerce platforms motivates sellers to promote their products using unfair means. Some of these sellers try to promote their products by colluding with dishonest buyers to provide fake reviews and/or ratings and by generating non-genuine behavioral signals such as clicks, add-to-carts and purchases on their products. These fake signals get propagated to the search engine, which subsequently upranks items that are of poorer quality. Meanwhile, genuine products get demoted and honest sellers suffer. Moreover, this behavior also means that honest buyers are exposed to poor quality and sometimes irrelevant items in response to their queries. Products promoted via unfair means could erode customers' trust in the search system. Hence, it is critical to identify dishonest sellers, buyers (who collude with these sellers), and their activities in a timely manner. Failure to do so will result in a loss for honest merchants and promotion of irrelevant and low quality products. This ultimately leads to a bad customer experience for the e-commerce store. An example collusion between dishonest sellers and buyers is depicted in Figure 1. The e-commerce store should identify such accounts with alacrity, and furthermore be robust to potential actors looking to game the search engine.

There are several challenges that arise in detecting bad actors. Since dishonest buyers/sellers look to "blend in" with honest ones, simple features that can be gleaned from anonymized logs cannot be used to detect them. Specifically, standard outlier detection methods (Chandola, Banerjee, and Kumar 2009), be it classification based, nearest neighbor based or clustering based, might fail to identify such dishonest actors. Furthermore, even when labeled data is present, the fraction of known bad actors is several orders of magnitude lower than good actors in these settings, making it hard to apply classification based methods out of the box.

In most cases for fraud detection, one not only has features from buyer/seller logs but also additional structural information. In the context of detecting dishonest buyers and sellers in e-commerce stores, buyer/seller interactions form a large, heterogeneous graph. The nodes of the graph are the buyers, sellers and items. Edges correspond to various interactions between these entities, e.g., purchase, sales, review, etc. However, there typically is a lack of homophily in the graph: dishonest accounts associate more with honest buyers/sellers than the dishonest ones. This should not be too surprising, since as mentioned above, a dishonest seller is not only motivated to collude with buyers to get fake signals, but has to do this so as to escape detection from simple anomaly detection methods. In other words, a dishonest seller has to ensure that their actions go undetected so they can continue being dishonest without facing repercussions. The lack of homophily mentioned above also renders graph based methods such as belief propagation (Eswaran et al. 2017; Yedidia, Freeman, and Weiss 2001), label propagation (Zhu and Ghahramani 2002), and personalized pagerank (Lofgren et al. 2014) ineffective.

In this paper, we present a novel context-aware heterogeneous graph convolutional network, CONGCN, to address the above challenges for detecting dishonest actors in e-commerce. We make use of a large-scale attributed heterogeneous graph of buyers, products, and sellers from anonymized users' purchase, review, and feedback activities.

Our CONGCN bears a few distinct characteristics. First, in a *context-aware* manner, we learn separate representations for nodes in the graph from different contexts, and fuse them to have a final representation. Second, we design a *heterogeneous* graph convolutional network to perform convolutional operations respecting the heterogeneity of different edge types within each context. Third, we propose to sample the neighborhood of a node in a *label-aware* manner, employing a biased sampling model. This strategy allows us to always include labeled nodes that not only scales up the computation, but also amplifies the limited signal present due to the lack of labeled data. Extensive experiments on a large, real world dataset show that CONGCN outperforms several baseline methods that consider only features, only the heterogeneous graph, or even standard graph neural networks that take both into account.

Problem Setup

In this section, we formally set up the problem we want to solve. We will introduce notations that we will use for the rest of the paper, and also setup some preliminaries that will be useful in the sequel.

We assume we are given an attributed heterogeneous graph: $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R}, \mathcal{F})$, where $\mathcal{V}$ and $\mathcal{E}$ are the vertex and edge sets respectively. $\mathcal{T}$ and $\mathbf{R}$ refer to the set of node types and edge types in the graph respectively. Each node $v \in \mathcal{V}$ has a mapping $\phi : \mathcal{V} \rightarrow \mathcal{T}$ to its node type and each edge $e \in \mathcal{E}$ has a mapping $\tau : \mathcal{E} \rightarrow \mathcal{R}$ to its relation type. The attribute set $\mathcal{F} = \{\mathbf{F}_t | t \in \mathcal{T}\}$ consists of attribute matrix for nodes of type $t \in \mathcal{T}$. $\mathbf{F}_t$ is constructed by stacking attributes of all nodes v, such that its type, $\phi(v)$ is t. Note that the size of attributes (dimension of vector) could be of different size for different node types.

A small fraction of the nodes in the graph are labeled. Furthermore, any labeled nodes only belong to one class. In our setup, we know that a small set of sellers collude with buyers to obtain inappropriate signals to game the search platform. Our aim is to use this data to identify if there are other sellers/buyers in the graph that are also dishonest. Note that there are two challenges: a) a very small fraction of nodes will be dishonest, and a further smaller fraction of this is actually labeled, and b) we do not have any labels for known honest sellers/buyers. This makes it a Positive-Unlabeled (PU) learning problem (Li et al. 2014) with a very small labeled dataset. We let $\mathcal{L} \subset \mathcal{V}$ be the set of labeled nodes, and $|\mathcal{L}| \ll |\mathcal{V}|$. Throughout this paper, we let $y(v) = 1$ be the label corresponding to dishonest sellers/buyers in the graph. Our setup indicates that $y(v) = 1 \ \forall v \in \mathcal{L}$. Our problem can be stated formally below:

Problem 1 *Fraud detection in heterogeneous networks*

Given: *the attributed heterogeneous graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R}, \mathcal{F})$*; the labeled node sets* $\mathcal{L} = \{y(v) = 1 | v \in \mathcal{V}\}$, $|\mathcal{L}| \ll |\mathcal{V}|$

Predict: *the fraudulence score of remaining nodes in* $\mathcal{V} \backslash \mathcal{L}$

Data

Next, we briefly describe the dataset we use for our problem. We use anonymized buyer and seller logs from a popular e-commerce store over a random but continuous period of 6 months. We construct a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R}, \mathcal{F})$ between sellers, buyers and items from three contexts, namely, *purchase*, *review* and *feedback*. Table 2 details the various node and edge types in each of the contexts. In *purchase* context, buyers purchase a product from a seller that lists this particular product to be sold. In *review* context, buyers leave a review for a product, and in *feedback* context, buyers leave feedback directly for a seller. Note that the product reviews and seller feedbacks can be positive as well as negative, while the purchase and listing relationships are unsigned. Different edges/relations occur at different contexts and the edges between the same type of nodes in different context has entirely different semantics. For example, an edge between buyer and seller in the context of purchase and an edge between the same nodes in the context of feedback are different. In addition to the graph relationships, we also obtain buyer, seller, and product attributes from the logs based on order history. Buyers, sellers, and products each have attributes of different size.

Sampling

To ensure that we have a manageable and at the same time a representative dataset to train models on, we need to sample the data from the logs we collect. Note that a random sampling will not work here since that will not preserve neighborhood of known dishonest nodes. On the other hand, an extremely localized sampling strategy will not account for the global graph structure that we might want to capture. The structural information of the network encodes various higher level information. Hence, preserving it is critical.

To overcome the challenges mentioned above, we employed a variant of snowball sampling (Goodman 1961) to obtain a meaningful sampled network. Pseudocode for the sampling strategy we employed is detailed in Algorithm 1. We first perform a Breadth First Search (BFS) using the nodes $v \in \mathcal{L}$ as the seed nodes. Then, we uniformly sample a set of nodes $v \in \mathcal{V} \backslash \mathcal{L}$, and perform BFS on these as seed nodes. The union of the two sets are thus obtained, and the induced subgraph acts as the dataset we use.

Algorithm 1: Sampling algorithms for creating training data

Result: $\mathcal{G}(\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R}, \mathcal{F})$
$G = \emptyset$, $F =$ fraudulent seed nodes,
$H =$ honest seed nodes;
for *each of Purchase, Review, and Feedback logs* **do**
  $G' \leftarrow \emptyset$;
  for *each* $v \in F$ **do**
    $G' \leftarrow G' \cup$ BFS from v ;
  end
  for *each* $v \in H$ **do**
    $G' \leftarrow G' \cup$ BFS from v ;
  end
  $G \leftarrow G \cup G'$;
  Filter out nodes without attributes from G ;
end

The final dataset we have post-sampling has about 16MM nodes, of which almost 13MM are buyers, 2.7MM products and the rest are sellers. There are over 40MM edges in the network. We have summarized our dataset in Table 1.

Next, we tried to quantify the collusion evidence seen in the cases above. One way to measure this is by quantifying the rate at which dishonest buyers and sellers tend to connect with each other. To this end, we computed the fraction of dishonest and honest neighbors adjacent to each dishonest buyer in the network. Then, we computed the average of these fractions over all dishonest buyers. We then repeated the same process for the honest buyers, dishonest sellers, and honest sellers. Table 3 summarizes our result. We see from Table 3 that dishonest sellers connect with both dishonest and honest buyers. Dishonest buyers on the other hand do tend to connect with dishonest sellers, indicating collusion.

Proposed Method: CONGCN

We propose to solve our node labeling problem (Problem 1) by designing a deep neural network, and train the network

Table 1: Summary of our dataset used.

Data property	Value
Node types	{buyer, seller, product}
Edge types	See table 2
Total no. of nodes (post sampling)	16 M
Total no. of edges (post sampling)	40 M
No. of buyers	13 M
No. of products	2.7 M
No. of sellers	124 K
Buyer attribute length	54
Seller attribute length	154
Product attribute length	108

Table 2: The contexts and types of edges present in our dataset. Note that there are seven distinct types of relations.

Context	Edge Types
Purchase	buyer $\xrightarrow{\text{purchase from}}$ seller, product $\xrightarrow{\text{sold by}}$ seller, buyer $\xrightarrow{\text{purchase}}$ product
Review	buyer $\xrightarrow{\text{like}}$ product, buyer $\xrightarrow{\text{dislike}}$ product
Feedback	buyer $\xrightarrow{\text{like}}$ seller, buyer $\xrightarrow{\text{dislike}}$ seller

to predict the bad actors in the dataset. As mentioned earlier, we have a set $\mathcal{L} \subset \mathcal{V}$ of nodes which are known to be dishonest, with $|\mathcal{L}| \ll |\mathcal{V}|$. Following our earlier definition, we have $y(v) = 1$ for nodes in $\mathcal{L}$. Following earlier PU learning works, we assume a sample of nodes in $\mathcal{V} \backslash \mathcal{L}$ have $y(v) = 0$. Our goal is to use these node labels to train the deep network. Specifically, we develop a novel context-aware heterogeneous graph convolutional network to leverage node features and capture the heterogeneous nature of our network, differentiate various relations in our dataset, learn low-dimensional embeddings specific to different context and leverage these embeddings to classify whether a given node is dishonest or not.

One of challenges we face is (i) *modelling different types of relations*. As mentioned earlier, our network has several relations between three types of nodes (See Table 2). Naturally, the activities that lead to these relations are also very different: buying an item and leaving a review for a seller are related, but not the same. A standard GCN that does not take these differing contexts into account. The next challenge we face is to (ii) *ensure scalability in presence of skewed labels*. Graph convolution approaches often rely on subsampling the neighborhood to ensure scalability. However, a naive subsampling approach does not yield good solution in our setting. This is because the fraction of labeled nodes in our dataset is very low, and the fact that those labels are from a single class. Subsampling uniformly at random will mean that with high probability, we will miss the labeled nodes, and only aggregate the embeddings from unlabeled nodes.

Next we describe our novel approach CONGCN, which overcomes both of these challenges. (short for Context-aware Graph Convolutional Network). Our main idea in CONGCN is to define context aware graph convolution architecture, which learns context specific node embeddings, which are then aggregated before being fed into a classifier (See Figure 2). We detail CONGCN next.

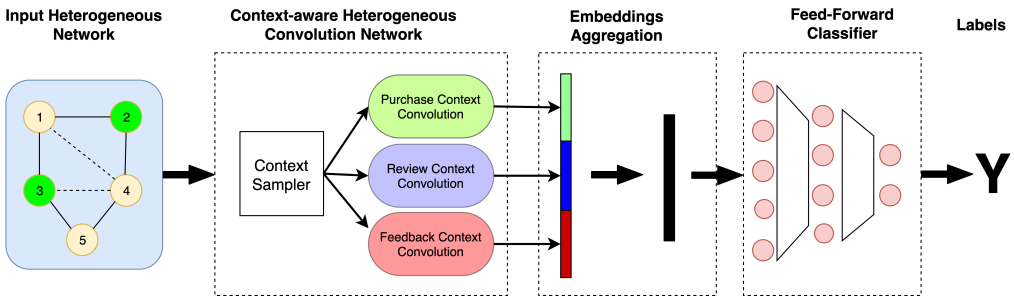
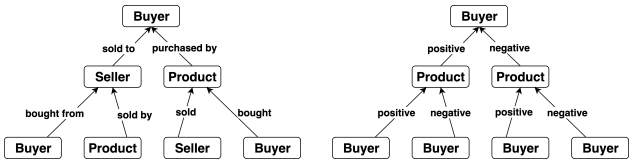


Figure 2: Overall Architecture of CONGCN. First context sample samples the heterogeneous subgraphs as per the predefined meta-trees, then the convolution operator on these sampled subgraphs generate context-aware embeddings. These embeddings are aggregated and fed into a classifier.

	D B	H B
D S	0.23%	99.77%
H S	$8.4e-5\%$	$\approx 100\%$

	D S	H S
D B	10.42%	89.58%
H B	3.63%	96.37%

Table 3: Collusion evidence in the network. FB, HB, FS, and HS stand for dishonest buyers, honest buyers, dishonest sellers, and honest sellers respectively. Each row shows the neighborhood distribution for a particular type of node.



(a) Purchase (b) Review

Figure 3: Meta-trees for different contexts. Meta-trees guide the sampling process as well as define our convolution operation.

Differentiating Contexts via Meta-Trees

Distinguishing the context from which different relations stem is critical in designing a deep neural network to capture heterogeneous networks in their true essence. In CONGCN, we do so by separating out the different contexts. This is implemented in two steps. First, we sample context specific heterogeneous subgraphs from the entire graph $\mathcal{G}$ and second, we separate convolution layers for each context (See Figure 2).

We represent contexts by defining meta-trees. A meta-tree for a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R}, \mathcal{F})$ is a tree where nodes are a subset of the node-types and edges are subset of the relationship types. Note that the meta-tree can also be heterogeneous in nature. Formally, we define the concept of meta-trees as follows:

Definition 1 *Given a heterogeneous graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R}, \mathcal{F})$, a meta-tree τ is a directed tree $\mathcal{G}' = (\mathcal{V}', \mathcal{E}')$, where $\mathcal{V}' \subset \mathcal{T}$ and $\mathcal{E}' \subset \mathcal{R}$. Each $e \in \mathcal{E}'$ is directed from child to parent node.*

A meta-tree is designed to capture a specific context. We show three different types of meta-trees for classifying buyers in Figure 3. Consider a meta-tree for *Purchase* context rooted on buyer nodes in Figure 3 (a). The *Purchase* context has three node-types, namely, Buyer, Seller, and Products. The root node indicates that the convolution defined by the meta-tree learns features for buyers. In the top layer, the Buyer node in the meta-tree has two children, Seller and Product. Seller to Buyer edge has a *sold-to* relation and Product to Buyer edge has a *purchased-by* relation. Similarly, in the second layer, Seller and Product nodes have two children each as per the purchase relationship. In Figure 3 (b), we show meta-tree for buyers in the *Review* context. Here, the root node Buyer has two children, each of type Product. However, the relation type to the children differ. The left child has $positive$ relation with the Buyer node, whereas the right child has $negative$ relation, corresponding to the type of review left by the buyer to the product (see Table 2). In turn, at the second layer, each Product node has two Buyer children each.

Due to lack of space we do not show other meta-trees including the ones rooted on Seller nodes. However, they can be constructed in a similar fashion to those for buyers.

Heterogeneous Convolution on Meta-Trees

Note that a meta-tree also defines computational subgraph for a heterogeneous convolutional layer. The subgraphs instantiated by a context specific meta-tree in the original heterogeneous network $\mathcal{G}$ define how features are computed and aggregated at different convolution layers. To describe the convolution operations due to a meta tree, we introduce some additional notations.

A neighborhood $\mathcal{N}(v)$ of a node v, with respect to a meta tree τ at layer l exists only if the node-type of v, $\phi(v)$ is present in τ at layer l. In such a case, the neighborhood of node b with respect to tree τ, the original graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{T}, \mathcal{R}, \mathcal{F})$, layer l and branch $e' \in E'$ is given by the following:

$$\mathcal{N}(v|\tau, G, l, e') = \{u \mid (v,u) \in E \text{ and } e' = (\phi(u), \phi(v))\} \quad (1)$$

The equation above states that the neighborhood of node v with respect to layer l and branch e' consists of nodes which are its neighbors in the original network $\mathcal{G}$ and whose type matches the type indicated by the child node in the branch

e' of the meta tree τ. For example, the neighbors of a buyer node v_b in the second layer of meta-tree for *Purchase* context as per the sold-to branch (See Figure 3 (a)) consists only of sellers which are connected to v_b in the original network with the sold-to relation. Next, we define our convolution layer as follows:

$$H_\tau^{(l+1)}[v] = \underset{e \in \text{branch}}{Agg} \left[f\left(W_e^l \sum_{u \in \mathcal{N}(v \mid \tau, \mathrm{G}, l, e)} H_\tau^l[u] \right) \right] \quad (2)$$

where $H_\tau^l[v]$ is the node v's feature in the layer l of convolution, W_e^l is the learnable weight parameter corresponding to layer l and branch e, $f(\cdot)$ is the non-linear activation function rectified linear unit (Nair and Hinton 2010), and $Agg(\cdot)$ is the aggregation function, which could be *mean*, *sum*, *max* and *concatenation*. The weight matrices W_e^l make sure the representations from different branches have the same length for *mean*, *sum* and *max* aggregations.

To illustrate the convolution operation, let's take an example to see how to update a particular buyer node's representation due to the meta-trees in the *Purchase* context . Let $\mathbf{F}_B$, $\mathbf{F}_S$, $\mathbf{F}_P$ be matrix of buyer, seller, and product features respectively, where a^{th} row of $\mathbf{F}_B$ is the feature of the buyer a. Now, the feature for node a at second layer with respect to Purchase meta-tree with *sum* aggregation operator is as follows:

$$H_\tau^{(2)}[a] = f\left(W_1^2 \sum_{b \in \mathcal{N}_1(a)} H_\tau^1[b] \right) + f\left(W_2^2 \sum_{c \in \mathcal{N}_2(a)} H_\tau^1[c] \right) \quad (3)$$

where $\mathcal{N}_1(a)$ and $\mathcal{N}_2(a)$ are the sets of neighbors of node a who are sellers connected via 'sold to' relations and neighbors who are products connected via 'purchased by' relations respectively. $H_\tau^1[b]$ for nodes in $\mathcal{N}_1(a)$ is computed in a similar fashion by aggregating features ($\mathbf{F}_B$ and $\mathbf{F}_P$) of neighbors of node a.

Note that we maintain different weight matrices for each edge in the meta-tree. It enables us to have different feature length for different node types at each layer and also allows us to easily handle the issue of each node type having different feature length.

Scalability via Biased Sampling

The runtime of convolutions depend linearly on the size of the neighborhood of each node. For real-world graphs with power law degree distribution, the hub nodes will have high degree, making the operation cumbersome. To alleviate this, GCNs subsample the neighborhoods for nodes that are heavily connected. We set a threshold k such that if a node has more than k neighbors, we randomly sample k nodes to perform the aggregation in Eq. (2).

While the sampling strategy described above makes GCNs scalable, it does not address a second concern: a very small fraction of the nodes in the graph is actually labeled. In the course of sampling the neighborhood uniformly at random, we could end up not sampling labeled nodes and lose valuable information in the process. To alleviate this, we perform a weighted random sampling, where the probability of sampling a positively labeled $y(v) = 1$ node is up-weighted by a constant $\alpha > 1$. This allows us to not end up with a sampled neighborhood of only unlabeled $y(v) = 0$ nodes.

Overall Architecture and Training Algorithm

The overall architecture of CONGCN is shown in Figure 2. For each node to be classified, a context sampler samples a subgraph from $\mathcal{G}$ for each context as per the meta-tree for the context. Then, we apply context aware convolution layers for each context which results in context-aware embeddings for each node. Specifically, for each node v, we obtain $\mathbf{F}_{pur}[v]$, $\mathbf{F}_{rev}[v]$, $\mathbf{F}_{fed}[v]$, which are embeddings of node v for purchase, review, and feedback respectively.

To obtain the final probability of a node being dishonest, we aggregate the embedding and pass it to a feed forward classification network. We experiment with various aggregation methods: *sum*, *mean*, element-wise *max* and *concatenation* We use Adam to train the model, and tune the learning rate, embedding size, neighborhood sample size, and experiment with different aggregation operators on a validation dataset. We train the model to minimize the binary cross-entropy loss.

Experiments

We design several experiments to evaluate the performance of CONGCN against several state-of-the art methods.We performed all of our experiments in an AWS p3.8x instance. We implemented CONGCN using the open-source DGL library (Wang et al. 2019a)[1].

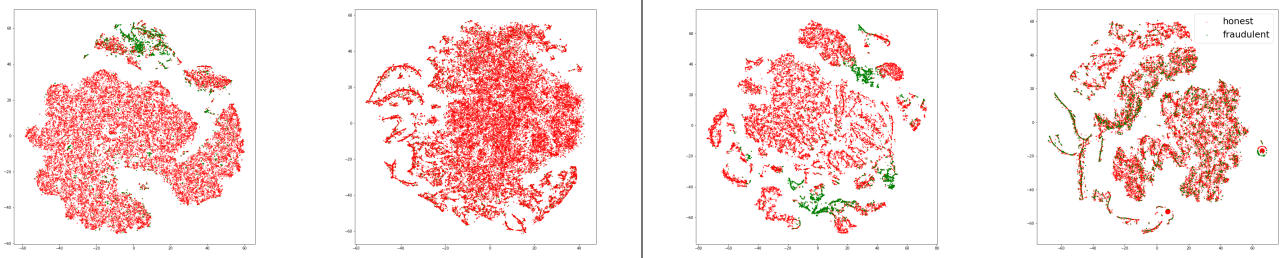
Training and Test Data: From our data (Table 1), we sample 30000 unlabeled and 3000 labeled dishonest buyers uniformly at random to create a test set. Here we are assuming that a random sample of unlabeled nodes are honest, which is a fair assumption. Similarly, we select 30000 unlabeled and 3000 labeled dishonest sellers uniformly at random as a test set. The rest of the nodes were used for training.

Baselines: We compare CONGCN against several baselines include Personalized PageRank (Lofgren et al. 2014), state-of-the-art graph based fraud detection approach ZooBP (Eswaran et al. 2017), and neural approaches including Feed-Forward Network, Graph Convolutional Network (GCN) (Kipf and Welling 2017), and GraphSage (Hamilton, Ying, and Leskovec 2017). For GCN and Graphsage we use the model implemented in the DGL library, and tune the relevant hyperparameters to obtain the best performance on our validation dataset.

Performance

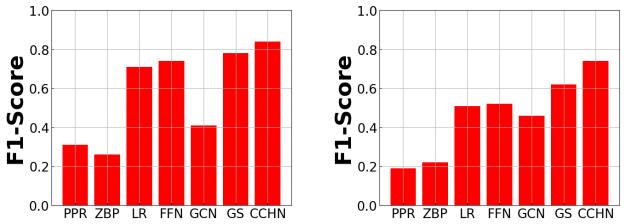
First, we measure performance of CONGCN and compare it against all the baselines for classifying both buyer and sellers. In this experiment, all the methods were trained leveraging the training data and the entire graph. We compute the F1-score for each method at the threshold where the it is maximized. We display the results in Figure 5.

[1] https://www.dgl.ai/



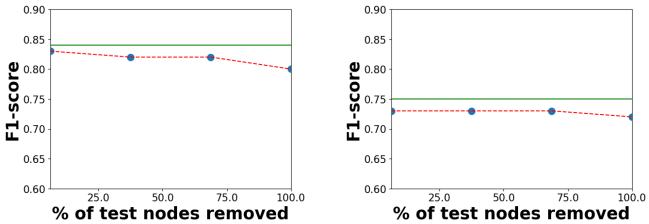
(a) Buyers (b) Sellers (c) Buyers (d) Sellers

Figure 4: Visualization of 2-d projection of buyers and sellers representations. The original features are on the left ((a), (b)) and the final representations learned by our approach are on the right ((c), (d)). In the original feature space, there is some separation between honest and dishonest buyers but there is no separation between honest and dishonest sellers. Our approach has improved the separation for both buyers and sellers.



(a) Buyer (b) Seller

Figure 5: Performance in classifying fraudulent buyers and sellers. CONGCN outperforms nearest baseline by 4% for buyers and by an impressive 19.3% for sellers.



(a) Buyer (b) Seller

Figure 6: CONGCN performs well in inductive settings. The green solid line and red dashed line are the performance in transductive and the inductive settings respectively.

We see from Figure 5 that CONGCN outperforms all the baselines for both buyers and sellers. The absolute gain of CONGCN over nearest competitor GraphSage is 4% for buyers and an impressive 19.3% for sellers. Several key observations can be made from the figure. The first is that all the methods consistently perform better in classifying buyers than in classifying sellers, implying that identifying dishonest sellers is a more challenging problem than identifying fraudulent buyers. This is explained by the observation that the collusion evidence for dishonest buyers are stronger than that for the dishonest sellers (Table 3).

The second observation we make is that the methods which rely only on graph structure perform worse than the methods which rely only on features, and the methods which use both graph structure and features (GCN, Graphsage and CONGCN) tend to perform the best generally. The reason behind relatively poor performance of only graph-based methods on our data are (1) class imbalance and (2) lack of clear homophily in the data. Due to heavy class imbalance, most of the graph is dominated by neighborhoods of honest nodes. Furthermore, we find that both dishonest buyers and sellers have more honest neighbors than the dishonest neighbors, showing a lack of homophily (See Table 3). Hence, graph based methods which assume that the nodes of the same class tend to connect more with each other fail. Moreover, we also observe that differences between the features of dishonest and honest buyers is more pronounced than that for the sellers (See Figure 4). We hypothesize this is because dishonest buyers show behavior that is vastly different from normal buyers, in that they display abnormal activity in the user logs. Dishonest sellers tend to blend in with other honest sellers.

Performance in an Inductive Setting

In the previous experiment, we measured the performance of CONGCN in a transductive setting, where we assume that the entire graph is known beforehand. However, in deployed systems, new buyer and seller accounts are constantly created. Hence, we would like CONGCN to be able to accurately classify new nodes in this inductive setting.

We simulate an inductive setting, where we do not observe the nodes in the test set as well as the edges incident upon them during training. We do so by removing various percentage of test nodes from the graph for the training. Note that removing 100% of test nodes and their edges from the graph is a complete inductive scenario. The results are shown in Figure 6. The green solid line represents the performance in the transductive setting. The y-axis represents F1-score and x-axis represents the percentage of test nodes removed from the graph. As seen, CONGCN maintains F1-score well

for both buyers and sellers. Even in the complete inductive setting (where 100% of the test nodes and their edges are removed from the graph) only a slight drop of 4.8% and 2.7% is observed for buyers and sellers respectively.

Ablation Study

Here we measure the effect of removing/altering different components of CONGCN. The first component we look at is the classifier module. Here, we remove feed-forward network and replace it with a single layer classifier. The second component we look at is our weighted sampling approach, were we replace our weighted sampling with the uniform sampling approach (See Section 4 in the main paper). The result is summarized in Table 4.

Table 4: Effect of altering components of CONGCN.

Component	Buyers			Sellers		
Methods	P	R	F1	P	R	F1
Classifier	0.79	0.80	0.80	0.80	0.66	0.72
Sampling	0.83	0.83	0.83	0.76	0.73	0.74
CONGCN	0.85	0.83	0.84	0.77	0.73	0.75

As shown in Table 4, the reduction in classifier significantly effects the performance of CONGCN for both buyer and seller. Similarly, the change in weighted to uniform neighbor sampling also has derogatory effect on the performance, implying that both the components are critical for the performance of CONGCN.

Effect of Aggregation Function

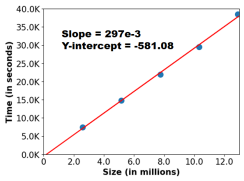
As mentioned earlier, various approaches can be leveraged to aggregate features generated by different contexts. Here, we quantify the effect of various aggregation strategies, namely mean, max, concatenation, and sum. Table 5 summarizes the results.

Table 5: Effect of different functions in aggregating the features generated by convolution on different contexts.

Component	Buyers			Sellers		
Methods	P	R	F1	P	R	F1
Mean	0.84	0.84	0.84	0.77	0.73	0.75
Max	0.80	0.79	0.79	0.80	0.70	0.75
Concatenation	0.87	0.78	0.82	0.74	0.71	0.72
Sum	0.85	0.83	0.84	0.78	0.71	0.74

We observe that the mean and sum perform consistently well for both buyers and sellers. The similarity between the two is due to the fact that mean is just a scaled version of the sum in this context as the number of features to aggregate are constant. On the other hand, we observe that max performs well for sellers whereas it performs slightly worse for buyers. Similarly, we see that concatenation degrades performance of CONGCN slightly for both buyers and sellers. Overall, we observe that CONGCN is robust to the choice of the aggregation function.

Scalability



Finally, we measure the scalability of CONGCN. We randomly sample subgraphs of our data of varying size and train CONGCN on these sampled subgraphs. We measure the time to train **100 epochs** of CONGCN on each subgraph. We keep other factors like batch size and hyper parameters constant. The result is presented in the figure on right. As we can see, the training time of CONGCN is linear w.r.t. the number of target nodes in the training set. For the complete network, it roughly takes 10.5 hours to train CONGCN for 100 epochs.

Related Work

Graph Convolution Networks There has been a surge in research in the area of graph convolution networks (Henaff, Bruna, and LeCun 2015; Kipf and Welling 2017). These methods build upon the idea of aggregating information from neighbors of a graph (Rao et al. 2015) to higher order neighborhoods (Shah, Rao, and Ding 2017; Wu et al. 2019) in a nonlinear fashion. Some popular recent graph convolutional models include Graph Attention Networks (Veličković et al. 2017), GraphSage (Hamilton, Ying, and Leskovec 2017), Graph LSTM (Tai, Socher, and Manning 2015), Patchy-SAN (Niepert, Ahmed, and Kutzkov 2016), Geniepath (Liu et al. 2019) which modify the way in which information is aggregated from (multi-hop) neighbors of a node.

For heterogenous graphs, Zhang et al. proposed GraphInception, which defines convolution on a set of homogeneous subgraphs sampled from a heterogeneous graph. Wang et al. proposed Heterogeneous Graph Attention networks (Wang et al. 2019b), which defined semantic level attention. However, the approach is not scalable to graphs with millions of nodes. Similarly, Schlichtkrul et al. proposed r-GCN (Schlichtkrull et al. 2018) for knowledge graphs with multi-relational data. Recently, Liu et al. proposed GEM for malicious account detection (Liu et al. 2018).

Graph Mining for E-commerce: A closely related field is graph mining for E-commerce. Graphs have been used for various tasks in E-commerce such as mining query relations (Adhikari et al. 2018; Beeferman and Berger 2000), item recommendation (Huang, Chung, and Chen 2004), intrusion detection (Foo et al. 2005) and data visualization (Hao et al. 2001).

Fraud Detection: Several approaches have been proposed for fraud detection. Recently, Eswaran et al. proposed belief prgopagation based approach for fraud detection in E-commerce (Eswaran et al. 2017). There have been works on fraudulent payments (Quah and Sriganesh 2008; Raj and Portia 2011) as well as fraudulent accounts (Liu et al. 2018). There also has been interest in detecting spam in E-commerce platforms (Lin et al. 2014; Lu et al. 2013; Heydari, Tavakoli, and Salim 2016).

Conclusions and Discussions

In this paper we present ConGCN, a scalable context-aware heterogeneous graph convolutional network. ConGCN can label nodes in a graph even when the number of known labeled nodes is very small, and all from a single class. To make our method account for multiple node types and relation types in the graph, our approach samples subgraphs instantiated from meta-trees on the graph. Experiments on a large dataset collected from an e-commerce store show that ConGCN outperforms multiple baselines that rely on only features, graph information and even neural network methods that combine both.

Deploying this model involves setting up the engineering pipelines to create the input data graph, and the subsampling routines. The resulting graph can then be fed into the ConGCN framework. Since the method is inductive, we can use this trained model to predict the probability of the nodes being bad. These predictions can be used directly or vended to other downstream models.

References

Adhikari, B.; Sondhi, P.; Zhang, W.; Sharma, M.; and Prakash, B. A. 2018. Mining E-Commerce Query Relations using Customer Interaction Networks. In *The World Wide Web Conference*, 1805–1814. International World Wide Web Conferences Steering Committee.

Beeferman, D.; and Berger, A. 2000. Agglomerative clustering of a search engine query log. In *KDD*, volume 2000, 407–416.

Chandola, V.; Banerjee, A.; and Kumar, V. 2009. Anomaly detection: A survey. *ACM computing surveys (CSUR)* 41(3): 15.

Eswaran, D.; Günnemann, S.; Faloutsos, C.; Makhija, D.; and Kumar, M. 2017. Zoobp: Belief propagation for heterogeneous networks. *VLDB* 10(5): 625–636.

Foo, B.; Wu, Y.-S.; Mao, Y.-C.; Bagchi, S.; and Spafford, E. 2005. ADEPTS: Adaptive intrusion response using attack graphs in an e-commerce environment. In *2005 International Conference on Dependable Systems and Networks (DSN'05)*, 508–517. IEEE.

Goodman, L. A. 1961. Snowball sampling. *The annals of mathematical statistics* 148–170.

Hamilton, W.; Ying, Z.; and Leskovec, J. 2017. Inductive representation learning on large graphs. In *NIPS*, 1024–1034.

Hao, M. C.; Dayal, U.; Hsu, M.; Sprenger, T.; and Gross, M. H. 2001. Visualization of directed associations in e-commerce transaction data. In *Data Visualization 2001*, 185–192. Springer.

Henaff, M.; Bruna, J.; and LeCun, Y. 2015. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163* .

Heydari, A.; Tavakoli, M.; and Salim, N. 2016. Detection of fake opinions using time series. *Expert Systems with Applications* 58: 83–92.

Huang, Z.; Chung, W.; and Chen, H. 2004. A graph model for E-commerce recommender systems. *Journal of the American Society for information science and technology* 55(3): 259–274.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. *ICLR 2017* .

Li, H.; Chen, Z.; Liu, B.; Wei, X.; and Shao, J. 2014. Spotting fake reviews via collective positive-unlabeled learning. In *2014 ICDM*, 899–904. IEEE.

Lin, Y.; Zhu, T.; Wu, H.; Zhang, J.; Wang, X.; and Zhou, A. 2014. Towards online anti-opinion spam: Spotting fake reviews from the review sequence. In *ASONAM 2014*, 261–264. IEEE.

Liu, Z.; Chen, C.; Li, L.; Zhou, J.; Li, X.; Song, L.; and Qi, Y. 2019. Geniepath: Graph neural networks with adaptive receptive paths. In *AAAI*, volume 33, 4424–4431.

Liu, Z.; Chen, C.; Yang, X.; Zhou, J.; Li, X.; and Song, L. 2018. Heterogeneous graph neural networks for malicious account detection. In *CIKM*, 2077–2085. ACM.

Lofgren, P. A.; Banerjee, S.; Goel, A.; and Seshadhri, C. 2014. FAST-PPR: scaling personalized pagerank estimation for large graphs. In *SIGKDD*, 1436–1445. ACM.

Lu, Y.; Zhang, L.; Xiao, Y.; and Li, Y. 2013. Simultaneously detecting fake reviews and review spammers using factor graph model. In *Proceedings of the 5th annual ACM web science conference*, 225–233. ACM.

Nair, V.; and Hinton, G. E. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*, 807–814.

Niepert, M.; Ahmed, M.; and Kutzkov, K. 2016. Learning convolutional neural networks for graphs. In *ICML*, 2014–2023.

Quah, J. T.; and Sriganesh, M. 2008. Real-time credit card fraud detection using computational intelligence. *Expert systems with applications* 35(4): 1721–1732.

Raj, S. B. E.; and Portia, A. A. 2011. Analysis on credit card fraud detection methods. In *2011 International Conference on Computer, Communication and Electrical Technology (ICCCET)*, 152–156. IEEE.

Rao, N.; Yu, H.-F.; Ravikumar, P. K.; and Dhillon, I. S. 2015. Collaborative filtering with graph information: Consistency and scalable methods. In *Advances in neural information processing systems*, 2107–2115.

Schlichtkrull, M.; Kipf, T. N.; Bloem, P.; Van Den Berg, R.; Titov, I.; and Welling, M. 2018. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, 593–607. Springer.

Shah, V.; Rao, N.; and Ding, W. 2017. Matrix Completion via Factorizing Polynomials. *arXiv preprint arXiv:1705.02047* .

Tai, K. S.; Socher, R.; and Manning, C. D. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* .

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph on networks. *arXiv preprint arXiv:1710.10903* .

Wang, M.; Yu, L.; Zheng, D.; Gan, Q.; Gai, Y.; Ye, Z.; Li, M.; Zhou, J.; Huang, Q.; Ma, C.; et al. 2019a. Deep graph library: Towards efficient and scalable deep learning on graphs. *arXiv preprint arXiv:1909.01315* .

Wang, X.; Ji, H.; Shi, C.; Wang, B.; Ye, Y.; Cui, P.; and Yu, P. S. 2019b. Heterogeneous Graph Attention Network. In *WWW*, 2022–2032. ACM.

Wu, F.; Zhang, T.; Souza Jr, A. H. d.; Fifty, C.; Yu, T.; and Weinberger, K. Q. 2019. Simplifying graph convolutional networks. *arXiv preprint arXiv:1902.07153* .

Yedidia, J. S.; Freeman, W. T.; and Weiss, Y. 2001. Generalized belief propagation. In *Advances in neural information processing systems*, 689–695.

Zhu, X.; and Ghahramani, Z. 2002. Learning from labeled and unlabeled data with label propagation. In *CMU CALD tech report*.