Untargeted Attack Against Federated Recommendation Systems via Poisonous Item Embeddings and the Defense

Yang Yu[1,2], Qi Liu[1,2*], Likang Wu[1,2], Runlong Yu[1,2], Sanshi Lei Yu[1,2], Zaixi Zhang[1,2]

[1]Anhui Province Key Laboratory of Big Data Analysis and Application,
School of Computer Science and Technology, University of Science and Technology of China

[2] State Key Laboratory of Cognitive Intelligence

Introduction

- Recommender systems are widely used to alleviate the information overload problem.
- Most existing recommender systems are trained on **centralized user data**.
  - Risk of data leakage.
  - Privacy concerns.
- Privacy regulations (e.g., GDPR, CCPA) make it more difficult to collect user data for centralized model training.

Introduction

- **Federated learning (FL)** enables multiple clients to collaboratively learn a global model without sharing their local data.
- Several studies have applied FL to train privacy-preserving **federated recommendation (FedRec)** systems.
- Unfortunately, FL is known to be vulnerable to **poisoning attacks**.
  - **Targeted Attack**
    - Increase the exposure rate of certain target items.
  - **Untargeted Attack**
    - Degrade the overall performance of the FedRec system.
    - Also known as the denial-of-service attack.
    - Continuously disrupt the user experience → Severe losses of customers and revenue.

Introduction

- **Challenges**
  - The attack method must be effective even with a small fraction of malicious clients.
  - The attacker can only access a small set of data stored on the malicious clients.
  - The attack needs to manipulate the model output on arbitrary inputs.
  - Many recommenders are naturally robust to malicious perturbation to a certain degree.

- In this work
  - **ClusterAttack**: an effective and covert untargeted model poisoning attack method.
  - **UNION**: a general uniformity-based defense mechanism.

Preliminaries

Federated Recommendation Systems

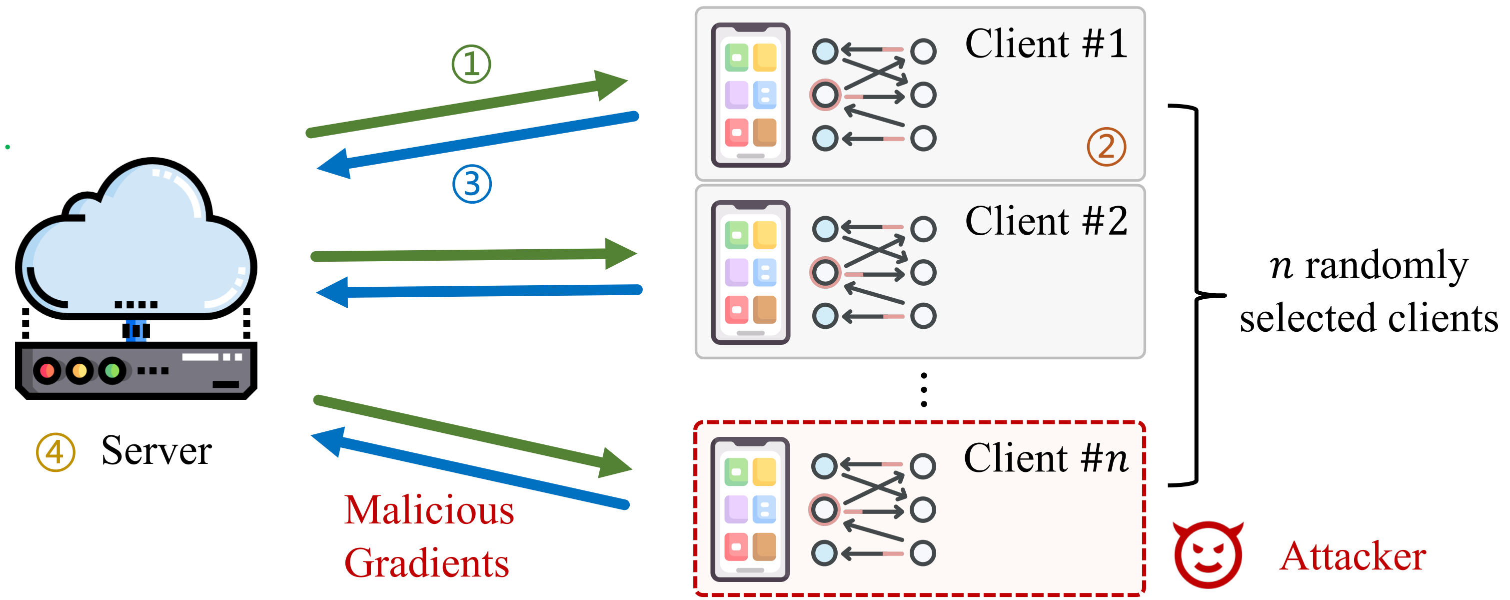
- The parameters of the recommendation model $\boldsymbol{\Theta} = [\boldsymbol{\Theta}_{\text{item}}; \boldsymbol{\Theta}_{\text{user}}; \boldsymbol{\Theta}_{\text{pred}}]$.
- Standard FL procedure

① Distribute global model $[\boldsymbol{\Theta}_{\text{item}}; \boldsymbol{\Theta}_{\text{pred}}]$.

② Compute local gradients $\mathbf{g} = [\mathbf{g}_{\text{item}}; \mathbf{g}_{\text{user}}; \mathbf{g}_{\text{pred}}]$.

③ Upload $[\mathbf{g}_{\text{item}}; \mathbf{g}_{\text{pred}}]$ and update local $\boldsymbol{\Theta}_{\text{user}}$ with $\mathbf{g}_{\text{user}}$.

④ Aggregate and update global model.



Threat Model

- $m\%$ (typically small, e.g., 1%) of clients are controlled by the attacker.
- The attacker knows the training code, local model, and user data of malicious clients.
- The attacker cannot access the data or gradients of other benign clients.

Methodology

- **ClusterAttack**
  - The recommendation model generally predicts the ranking score based on the user embedding and the item embedding.
  - Upload malicious gradients that converge item embeddings into several dense clusters.

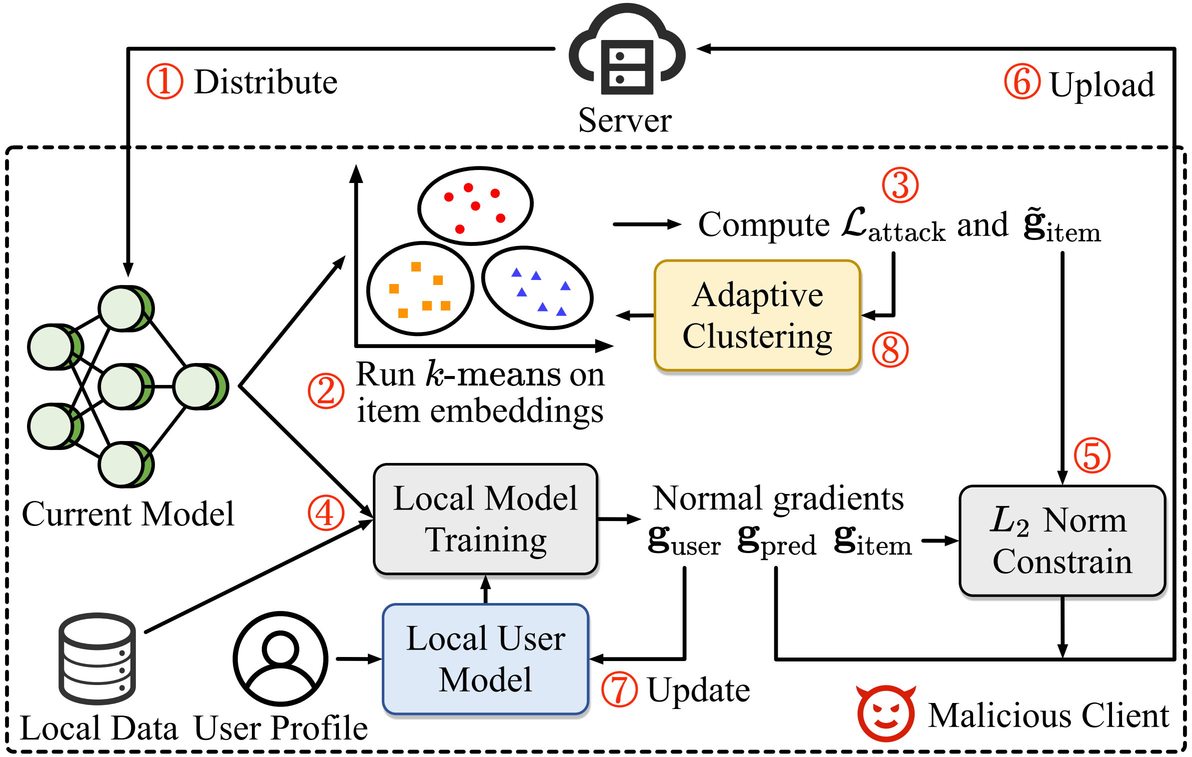


Figure 1: The procedure of ClusterAttack.

Methodology

ClusterAttack

- Apply k-means to split the item embeddings $\{\boldsymbol{v}_i\}_{i=1}^{M}$ into K clusters $\{C_i\}_{i=1}^{K}$ with centroids $\{\boldsymbol{c}_i\}_{i=1}^{K}$.
- Compute the within-cluster variance and the malicious gradient.

$$\mathcal{L}_{attack} = \sum_{i=1}^{K} \sum_{\boldsymbol{v}_j \in C_i} \left\| \boldsymbol{v}_j - \boldsymbol{c}_i \right\|_2^2 \qquad \tilde{\mathbf{g}}_{\boldsymbol{v}_i} = \frac{\partial \mathcal{L}_{attack}}{\partial \boldsymbol{v}_i}$$

- **Gradient clipping**
  - Compute the normal gradient of each malicious client.
  - Calculate the mean μ and standard deviation σ of the L_2 norms of all normal item embedding gradients.

$$b_i^{(j)} = \mu + \lambda_i^{(j)} \sigma, \ \lambda_i^{(j)} \in [0,3] \qquad \hat{\mathbf{g}}_{\boldsymbol{v}_i}^{(j)} = \frac{\tilde{\mathbf{g}}_{\boldsymbol{v}_i}}{\max(1, \left\|\tilde{\mathbf{g}}_{\boldsymbol{v}_i}\right\|_2 / b_i^{(j)})}$$

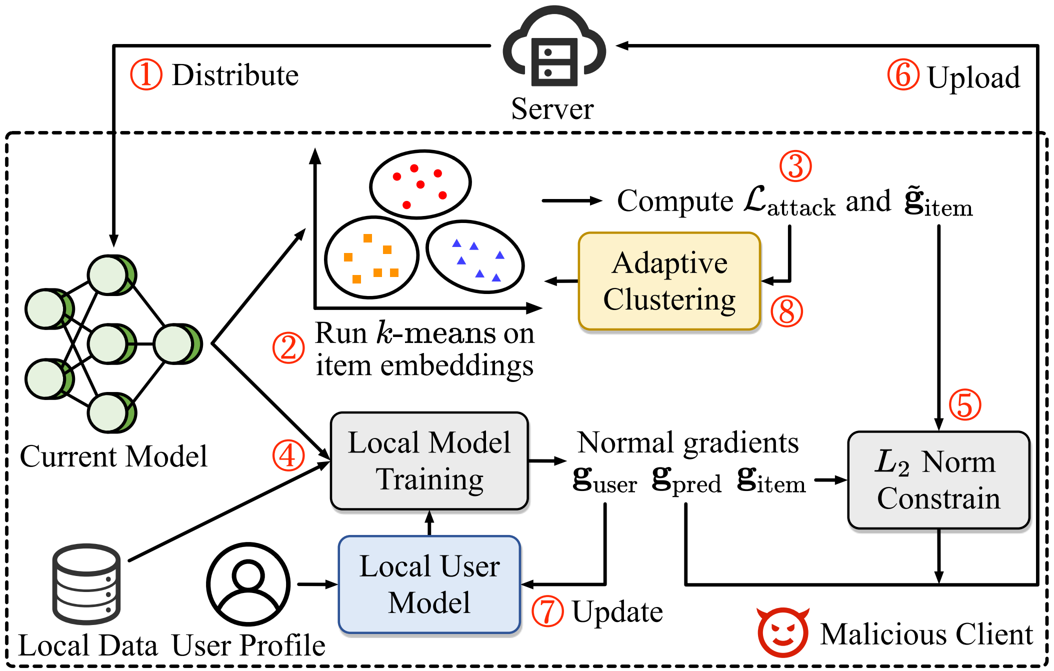


Figure 1: The procedure of ClusterAttack.

Methodology

ClusterAttack

Adaptive clustering

- Adjust the number of clusters K after each round of attack.
- Use the change of $\mathcal{L}_{attack}$ as the feedback.
- $\mathcal{L}_{attack}$ keeps increasing → $\mathcal{L}_{attack}$ cannot converge well.
- $\mathcal{L}_{attack}$ keeps decreasing → decrease K for stronger attack.

Algorithm 1: Adaptive Clustering

Input: Number of clusters K, range of number of clusters $[K_{\text{min}}, K_{\text{max}}]$, threshold R, and decay rate β.

Init: Set $\tilde{\mathcal{L}}^{(0)}_{\text{attack}}, n_{\text{inc}}, n_{\text{dec}}$ and t as 0.

// Repeat after each round of attack

$t \leftarrow t + 1$;

Calculate $\mathcal{L}^{(t)}_{\text{attack}}$ with Equation (2);

$\tilde{\mathcal{L}}^{(t)}_{\text{attack}} \leftarrow \beta \cdot \tilde{\mathcal{L}}^{(t-1)}_{\text{attack}} + (1-\beta) \cdot \mathcal{L}^{(t)}_{\text{attack}}$;

$\hat{\mathcal{L}}^{(t)}_{\text{attack}} \leftarrow \tilde{\mathcal{L}}^{(t)}_{\text{attack}} / \left(1-\beta^t\right)$;

if $\hat{\mathcal{L}}^{(t)}_{\text{attack}} > \hat{\mathcal{L}}^{(t-1)}_{\text{attack}}$ **then** $n_{\text{inc}} \leftarrow n_{\text{inc}} + 1$;

else $n_{dec} \leftarrow n_{\text{dec}} + 1$;

if $n_{\text{inc}} - n_{\text{dec}} \geqslant R$ **then**

  $K \leftarrow \min\left(\left\lfloor K + \sqrt{K_{\text{max}} - K}\right\rfloor, K_{\text{max}}\right)$;

  Reset $n_{\text{inc}}, n_{\text{dec}}$ and t as 0;

end if

if $n_{\text{dec}} - n_{\text{inc}} \geqslant R$ **then**

  $K \leftarrow \max\left(\left\lfloor K - \sqrt{K - K_{\text{min}}}\right\rfloor, K_{\text{min}}\right)$;

  Reset $n_{\text{inc}}, n_{\text{dec}}$ and t as 0;

end if

Methodology

UNION Mechanism

Client Side

- Train the local recommendation model with an **additional contrastive learning task**.
- Denote the item set interacted by the user as $\mathcal{V}_u = \{v_i\}_{i=1}^{L}$ and the entire item set as $\mathcal{V}$.
- For each $v_i \in \mathcal{V}_u$, randomly select another positive item $v_i^+ \in \mathcal{V}_u$ and P negative items $\{v_i^-\}_{i=1}^{P} \subseteq \mathcal{V} \backslash \mathcal{V}_u$.

$$\mathcal{L}_{\mathrm{cl}} = -\sum_{i=1}^{L} \log \frac{e^{f(v_i)^{\mathrm{T}} f(v_i^+)}}{e^{f(v_i)^{\mathrm{T}} f(v_i^+)} + \sum_{j=1}^{P} e^{f(v_i)^{\mathrm{T}} f(v_i^-)}} \qquad \mathcal{L} = \mathcal{L}_{\mathrm{rec}} + \alpha \mathcal{L}_{\mathrm{cl}}$$

- $\mathcal{L}_{\mathrm{cl}}$ can regularize the item embeddings toward a uniform distribution in the space [1] while training with the recommendation task (opposite to the goal of ClusterAttack).

[1] Wang et al. Understanding Contrastive Representation Learning through Alignment and Uniformity on the Hypersphere. In *ICML*. 2020.

Methodology

- UNION Mechanism
  - **Server Side**
    - Estimate the uniformity of updated item embeddings for each received gradient.

$$d_i = \mathbb{E}_{x,y \overset{\text{i.i.d}}{\sim} p_{\text{data}}} \|f(x) - f(y)\|_2^2$$

    - Use the **Gap Statistics algorithm** [2] to estimate the number of clusters in $\{d_i\}_{i=1}^n$.
    - If the algorithm estimates that there is more than one cluster, we apply k-means to split $\{d_i\}_{i=1}^n$ into two clusters and remove all the gradients belonging to the minor one.
  - **Note**
    - UNION is a general mechanism that aims to preserve the distribution of item embeddings.
    - It can be combined with existing Byzantine-robust FL methods (e.g., MultiKrum, NormBound) to provide more comprehensive protection for FedRec systems.

[2] Tibshirani et al. Estimating the Number of Clusters in a Data Set via the Gap Statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*. 2001.

Experiments

- **Datasets**
  - MovieLens-1M
  - Gowalla
- **Base Recommendation Model**
  - MF
  - SASRec
- **Metrics**
  - Hit Ratio (HR)
  - Normalized Discounted Cumulative Gain (NDCG)
  - Only calculated on benign clients using the all-ranking protocol.

Dataset	#Users	#Items	#Actions	Avg. length	Density
ML-1M	6,040	3,706	1,000,209	165.6	4.47%
Gowalla	29,858	40,981	1,585,043	53.1	0.13%

Table 2: Detailed statistics of the two datasets.

Experiments

Attack Performance with No Defense

Model	Attack Method	ML-1M		Gowalla	
		HR@5	NDCG@5	HR@5	NDCG@5
MF	No Attack	0.03549 (-)	0.02226(-)	0.02523 (-)	0.01697 (-)
	LabelFlip	0.03561 (-0.34%)	0.02238 (-0.54%)	0.02541 (-0.71%)	0.01711 (-0.82%)
	FedAttack	0.03358 (5.38%)	0.02118 (4.85%)	0.02371 (6.02%)	0.01585 (6.60%)
	Gaussian	0.03555 (-0.17%)	0.02224 (0.09%)	0.02528 (-0.20%)	0.01701 (-0.24%)
	LIE	0.03259 (8.17%)	0.02062 (7.37%)	0.02316 (8.20%)	0.01571 (7.42%)
	Fang	0.03038 (14.40%)	0.01897 (14.78%)	0.02131 (15.54%)	0.01448 (14.67%)
	ClusterAttack	**0.02451 (30.94%)**	**0.01545 (30.59%)**	**0.01664 (34.05%)**	**0.01117 (34.18%)**
SASRec	No Attack	0.10810 (-)	0.07053 (-)	0.03251 (-)	0.02217 (-)
	LabelFlip	0.10857 (-0.43%)	0.07071 (-0.26%)	0.03270 (-0.58%)	0.02222 (-0.23%)
	FedAttack	0.10013 (7.37%)	0.06572 (6.82%)	0.03054 (6.06%)	0.02087 (5.86%)
	Gaussian	0.10769 (0.38%)	0.07055 (-0.03%)	0.03226 (0.77%)	0.02222 (-0.23%)
	LIE	0.09677 (10.48%)	0.06281 (10.95%)	0.03008 (7.47%)	0.02021 (8.84%)
	Fang	0.08964 (17.08%)	0.05909 (16.22%)	0.02797 (13.96%)	0.01883 (15.07%)
	ClusterAttack	**0.06547 (39.44%)**	**0.04130 (41.44%)**	**0.02223 (31.62%)**	**0.01544 (30.36%)**

Table 1: Model performance under different untargeted attack methods with no defense. The percentages in parentheses indicate the relative performance degradation compared with the no-attack scenario.

Experiments

Attack Performance with No Defense

Model	Attack Method	ML-1M		Gowalla	
		HR@5	NDCG@5	HR@5	NDCG@5
MF	No Attack	0.03549 (-)	0.02226(-)	0.02523 (-)	0.01697 (-)
	LabelFlip	0.03561 (-0.34%)	0.02238 (-0.54%)	0.02541 (-0.71%)	0.01711 (-0.82%)
	FedAttack	0.03358 (5.38%)	0.02118 (4.85%)	0.02371 (6.02%)	0.01585 (6.60%)
	Gaussian	0.03555 (-0.17%)	0.02224 (0.09%)	0.02528 (-0.20%)	0.01701 (-0.24%)
	LIE	0.03259 (8.17%)	0.02062 (7.37%)	0.02316 (8.20%)	0.01571 (7.42%)
	Fang	0.03038 (14.40%)	0.01897 (14.78%)	0.02131 (15.54%)	0.01448 (14.67%)
	ClusterAttack	**0.02451 (30.94%)**	**0.01545 (30.59%)**	**0.01664 (34.05%)**	**0.01117 (34.18%)**
SASRec	No Attack	0.10810 (-)	0.07053 (-)	0.03251 (-)	0.02217 (-)
	LabelFlip	0.10857 (-0.43%)	0.07071 (-0.26%)	0.03270 (-0.58%)	0.02222 (-0.23%)
	FedAttack	0.10013 (7.37%)	0.06572 (6.82%)	0.03054 (6.06%)	0.02087 (5.86%)
	Gaussian	0.10769 (0.38%)	0.07055 (-0.03%)	0.03226 (0.77%)	0.02222 (-0.23%)
	LIE	0.09677 (10.48%)	0.06281 (10.95%)	0.03008 (7.47%)	0.02021 (8.84%)
	Fang	0.08964 (17.08%)	0.05909 (16.22%)	0.02797 (13.96%)	0.01883 (15.07%)
	ClusterAttack	**0.06547 (39.44%)**	**0.04130 (41.44%)**	**0.02223 (31.62%)**	**0.01544 (30.36%)**

Table 1: Model performance under different untargeted attack methods with no defense. The percentages in parentheses indicate the relative performance degradation compared with the no-attack scenario.

Experiments

Attack Performance with No Defense

Model	Attack Method	ML-1M		Gowalla	
		HR@5	NDCG@5	HR@5	NDCG@5
MF	No Attack	0.03549 (-)	0.02226(-)	0.02523 (-)	0.01697 (-)
	LabelFlip	0.03561 (-0.34%)	0.02238 (-0.54%)	0.02541 (-0.71%)	0.01711 (-0.82%)
	FedAttack	0.03358 (5.38%)	0.02118 (4.85%)	0.02371 (6.02%)	0.01585 (6.60%)
	Gaussian	0.03555 (-0.17%)	0.02224 (0.09%)	0.02528 (-0.20%)	0.01701 (-0.24%)
	LIE	0.03259 (8.17%)	0.02062 (7.37%)	0.02316 (8.20%)	0.01571 (7.42%)
	Fang	0.03038 (14.40%)	0.01897 (14.78%)	0.02131 (15.54%)	0.01448 (14.67%)
	ClusterAttack	**0.02451 (30.94%)**	**0.01545 (30.59%)**	**0.01664 (34.05%)**	**0.01117 (34.18%)**
SASRec	No Attack	0.10810 (-)	0.07053 (-)	0.03251 (-)	0.02217 (-)
	LabelFlip	0.10857 (-0.43%)	0.07071 (-0.26%)	0.03270 (-0.58%)	0.02222 (-0.23%)
	FedAttack	0.10013 (7.37%)	0.06572 (6.82%)	0.03054 (6.06%)	0.02087 (5.86%)
	Gaussian	0.10769 (0.38%)	0.07055 (-0.03%)	0.03226 (0.77%)	0.02222 (-0.23%)
	LIE	0.09677 (10.48%)	0.06281 (10.95%)	0.03008 (7.47%)	0.02021 (8.84%)
	Fang	0.08964 (17.08%)	0.05909 (16.22%)	0.02797 (13.96%)	0.01883 (15.07%)
	ClusterAttack	**0.06547 (39.44%)**	**0.04130 (41.44%)**	**0.02223 (31.62%)**	**0.01544 (30.36%)**

Table 1: Model performance under different untargeted attack methods with no defense. The percentages in parentheses indicate the relative performance degradation compared with the no-attack scenario.

Experiments

Attack Performance under Defense (Left five groups)

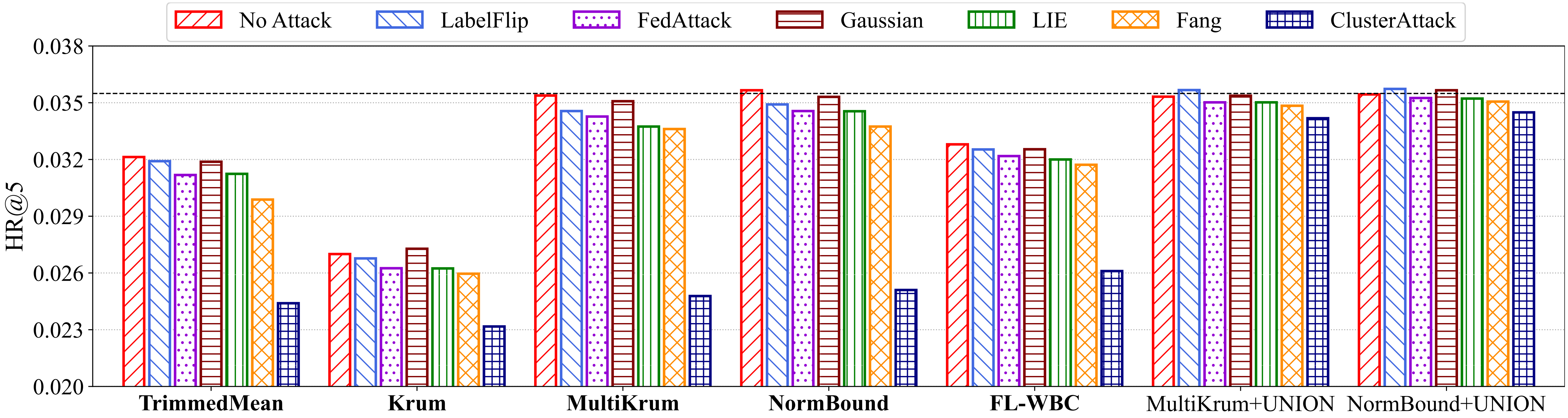


Figure 2: Model performance under different untargeted attack methods with different defense mechanisms. The black dashed line represents the model performance without any attack or defense.

Experiments

Attack Performance under Defense (Left five groups)

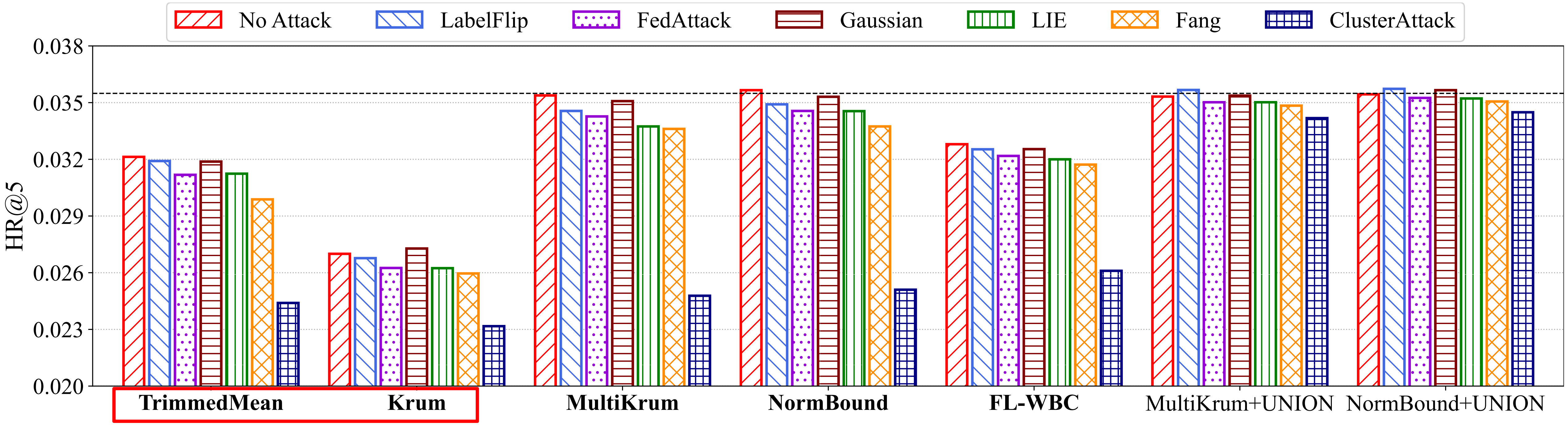


Figure 2: Model performance under different untargeted attack methods with different defense mechanisms. The black dashed line represents the model performance without any attack or defense.

Experiments

Attack Performance under Defense (Left five groups)

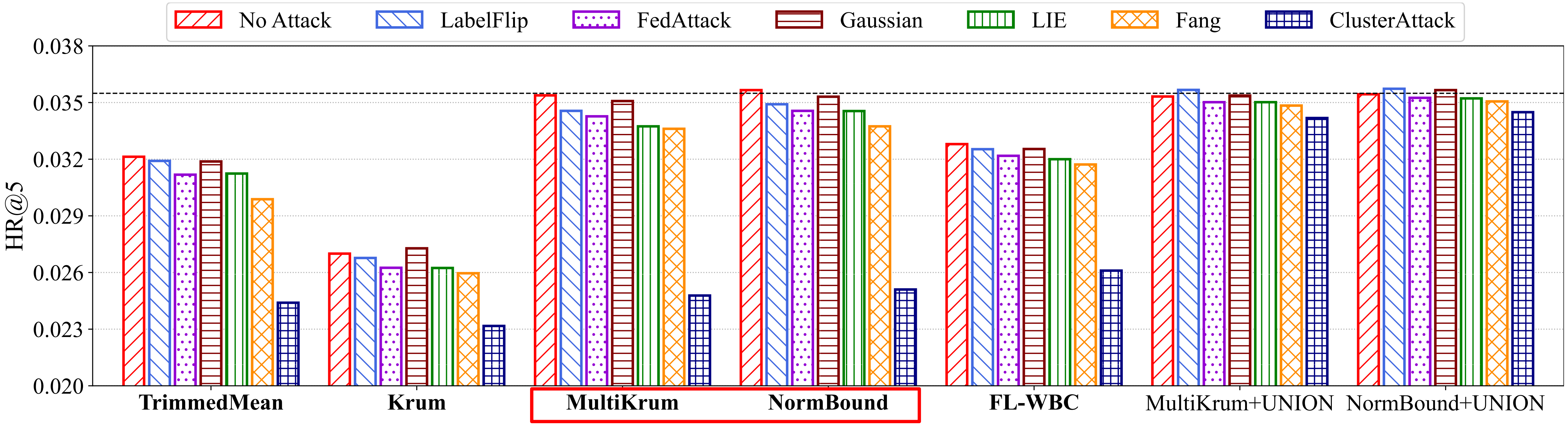


Figure 2: Model performance under different untargeted attack methods with different defense mechanisms. The black dashed line represents the model performance without any attack or defense.

Experiments

Defense Performance (Right two groups)

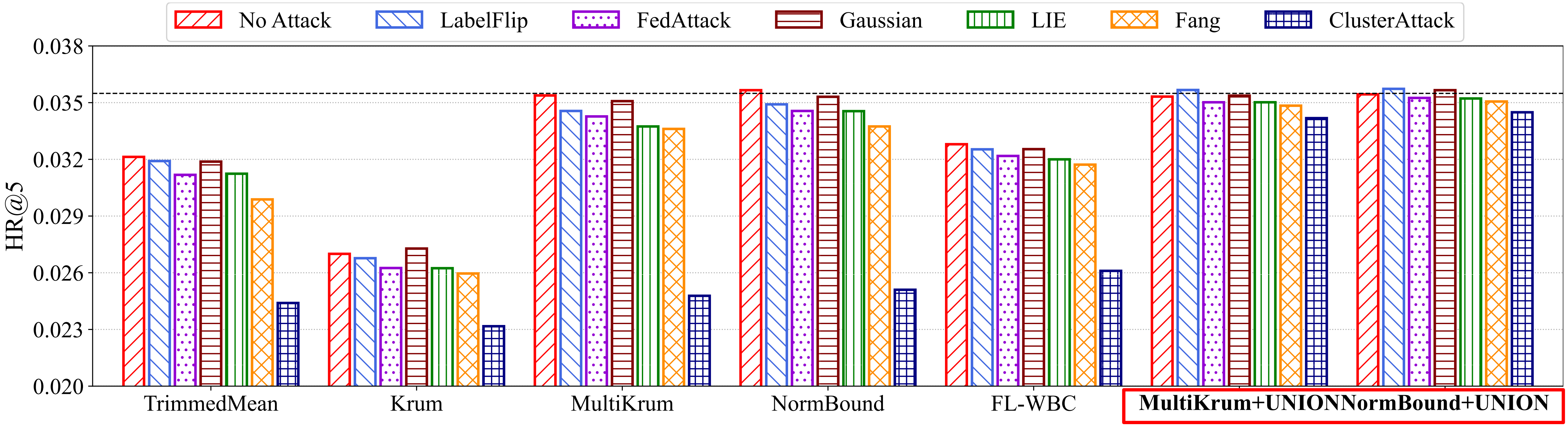


Figure 2: Model performance under different untargeted attack methods with different defense mechanisms. The black dashed line represents the model performance without any attack or defense.

Experiments

- **Can the attacker evade UNION?**
  - $\mathcal{L}'_{attack} = \mathcal{L}_{attack} + \alpha \cdot \mathcal{L}_{cl}$
  - The extra contrastive learning task weakens the attack effect of ClusterAttack.

Defense Method	Attack Method	HR@5
MultiKrum+UNION	ClusterAttack	0.03378 (4.82%)
	ClusterAttack+CL	0.03525 (0.68%)
NormBound+UNION	ClusterAttack	0.03449 (2.82%)
	ClusterAttack+CL	0.03566 (-0.48%)

Table 3: Attack performance of ClusterAttack+CL.

Experiments

Impact of Adaptive Clustering

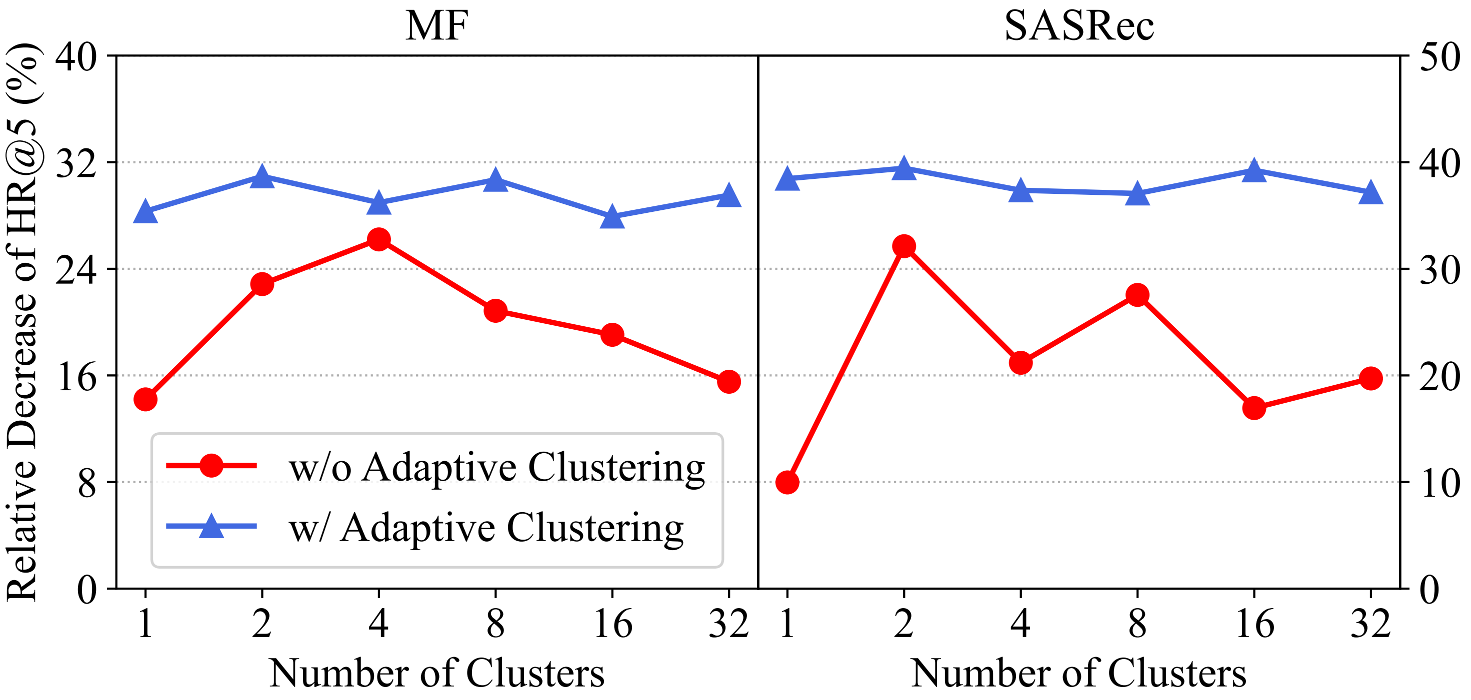


Figure 3: Impact of adaptive clustering.

Experiments

Gradients and Uniformity Analysis

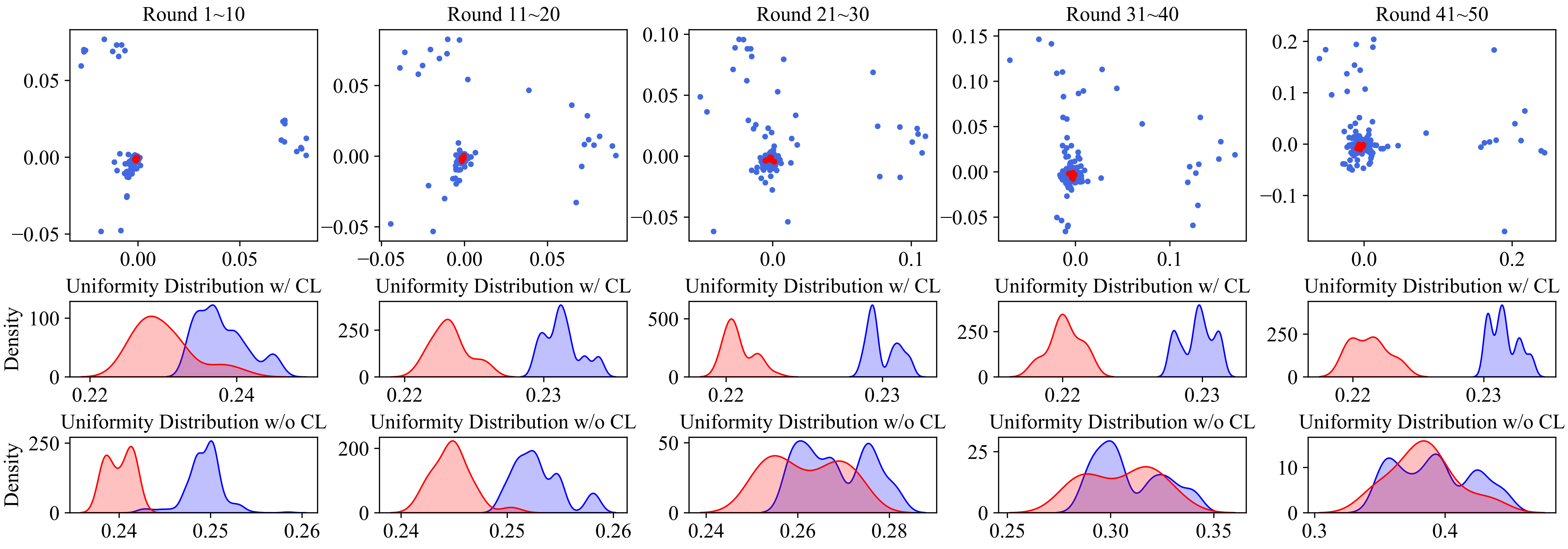


Figure 4: Visualization of the uploaded gradients and the uniformity distribution in different rounds of model training. The blue color and red color denote benign clients and malicious clients, respectively.

Conclusion

- ClusterAttack
  - Uploads malicious gradients that converge the item embeddings into dense clusters.
  - Reveals the security risk of FedRec systems even with existing defense methods.
- UNION
  - Preserves the distribution of item embeddings with an additional contrastive learning task.
  - Combines with existing Byzantine-robust FL methods to better protect the FedRec system from potential untargeted attacks in the real world.
- Extensive experiments validate the effectiveness of our attack and defense methods.

Paper

Code

Thanks For Your Attention