SYBILS IN FL

MITIGATING SYBILS IN FEDERATED LEARNING

REFERENCE

- Fung, Clement, Chris JM Yoon, and Ivan Beschastnikh. "Mitigating sybils in federated learning poisoning." arXiv preprint arXiv:1808.04866 (2018).

FEDERATED LEARNING

FEDERATED LEARNING

- 본 논문에서는 표준 연합 학습 환경을 상정
- 데이터
  - 여러 소유자에게 분산되어 있음
  - 공유되지 않음

FEDERATED LEARNING

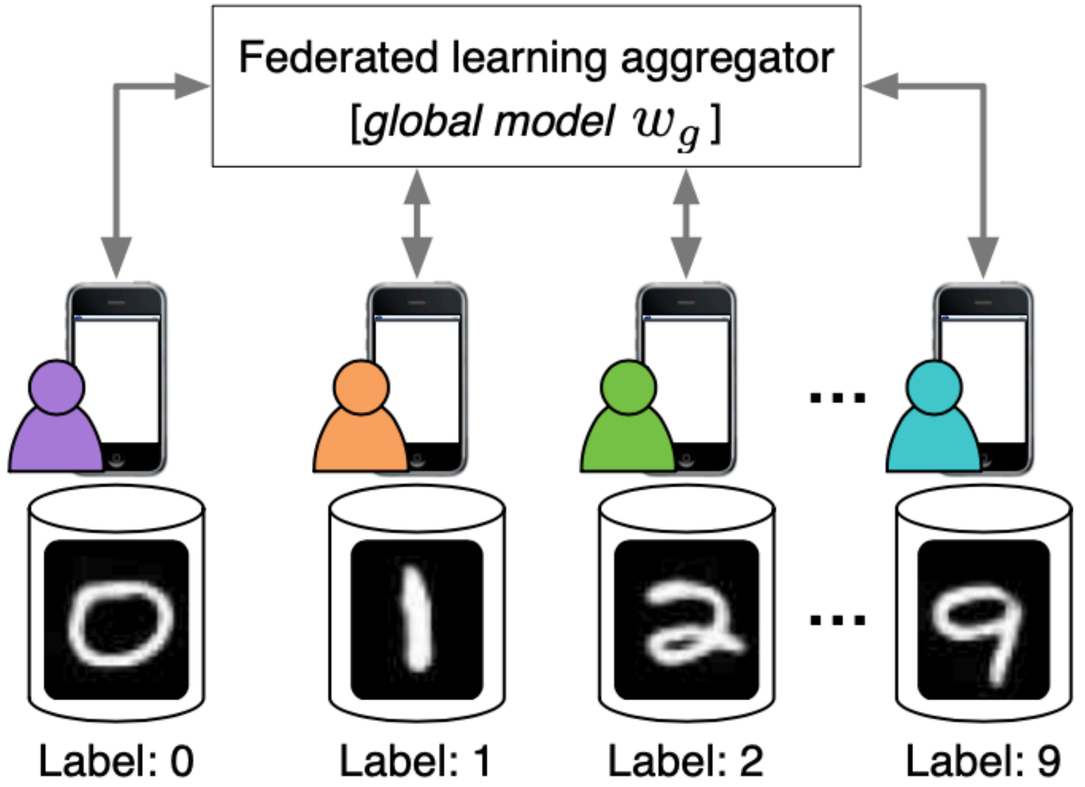
- 본 논문에서는 표준 연합 학습 환경을 상정
- 분산 학습 절차

$$w_{g,t+1} = w_{g,t} + \sum_k \frac{n_k}{n} \Delta_{k,t}$$

  - 클라이언트의 집합에 걸친 동기적(Synchronous) 업데이트 라운드
  - k 클라이언트 업데이트의 가중 평균으로
    - 학습 데이터 사이즈에 대한 가중치

FEDERATED LEARNING

- 데이터가 non-IID (not independent and identically) 해도
  - 여전히 수렴



FEDERATED LEARNING

- FL에는 두 방법이 있음:
- FEDSGD
  - 각 클라이언트가 모든 SGD 업데이트를 서버에게 전송
- FEDAVG
  - 클라이언트가 지역적으로 여러 SGD의 이터레이션을 수행 후, 서버에게 전송
  - 통신 효율적

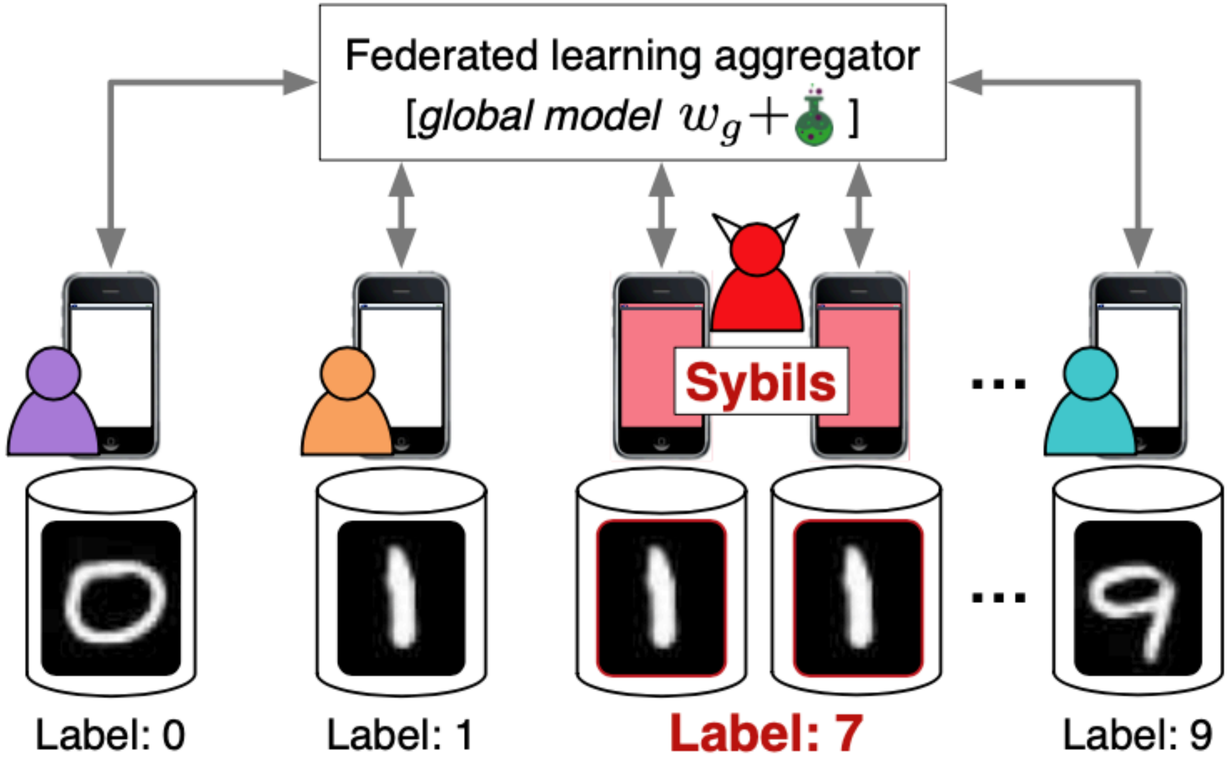
POISONING ATTACK & SYBIL ATTACK

TARGETED POISONING ATTACKS ON ML

- 공격자가 세심하게 학습 데이터를 오염시킴
  - 이를 공격할 모델의 학습 데이터셋에 포함
- 학습된 모델이 목표한 예제를 목표한 클래스로 분류할 확률을
  - 높게 또는 낮게 조정할 수 있음
  - Fraud detection
  - 스팸 메일 필터링 회피 등

TARGETED POISONING ATTACKS ON ML

- 학습된 모델이 목표한 예제를 목표한 클래스로 분류할 확률을
  - 높게 또는 낮게 조정

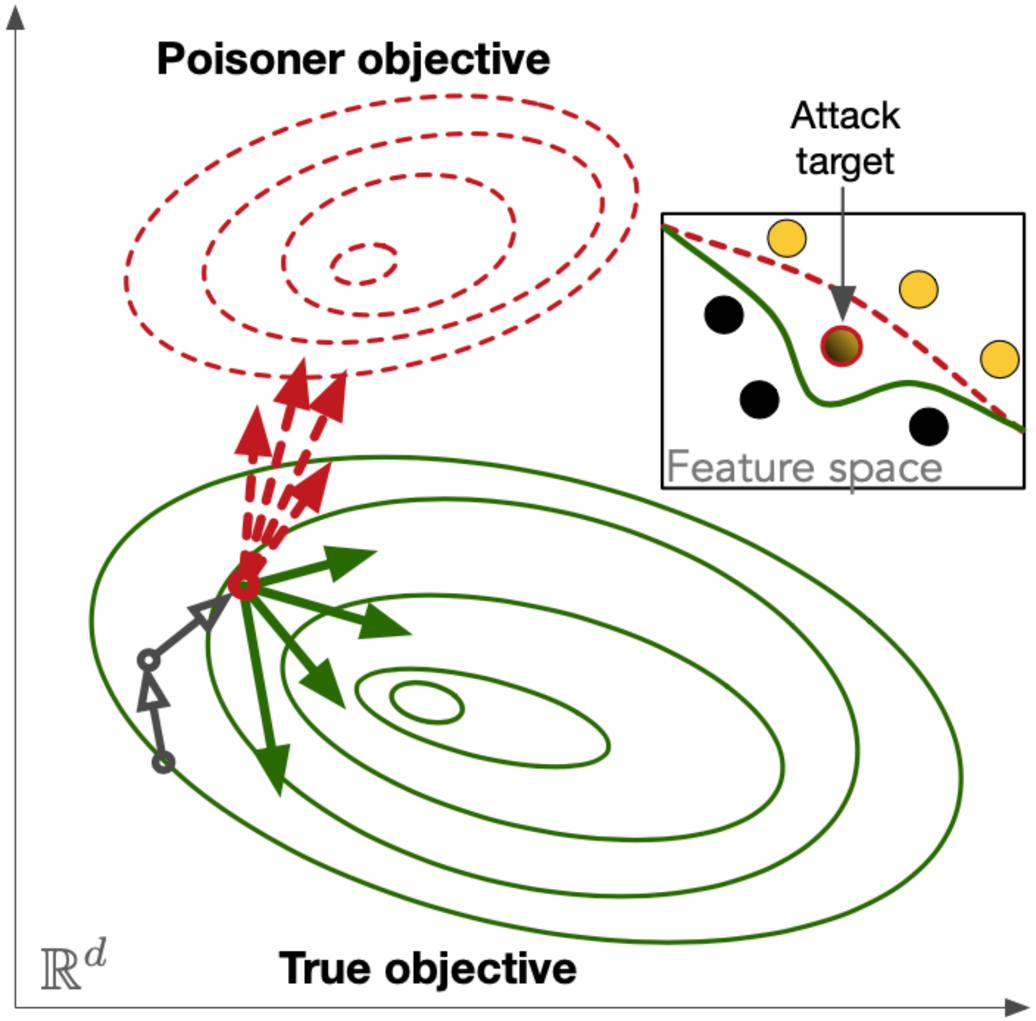


TARGETED POISONING ATTACKS ON ML

- 본 논문에서는 두 종류의 targeted attack을 고려:
- Label-flipping 공격
  - 한 클래스의 정직한 학습 예제의 레이블이 다른 클래스로 바뀜
- 백도어(Backdoor) 공격
  - 데이터에서 하나의 특징(feature) 또는 작은 부분이
  - 비밀 패턴으로 덧씌워지거나 재-레이블 됨
  - 비밀 패턴이 목표 클래스로 가는 트리거가 됨

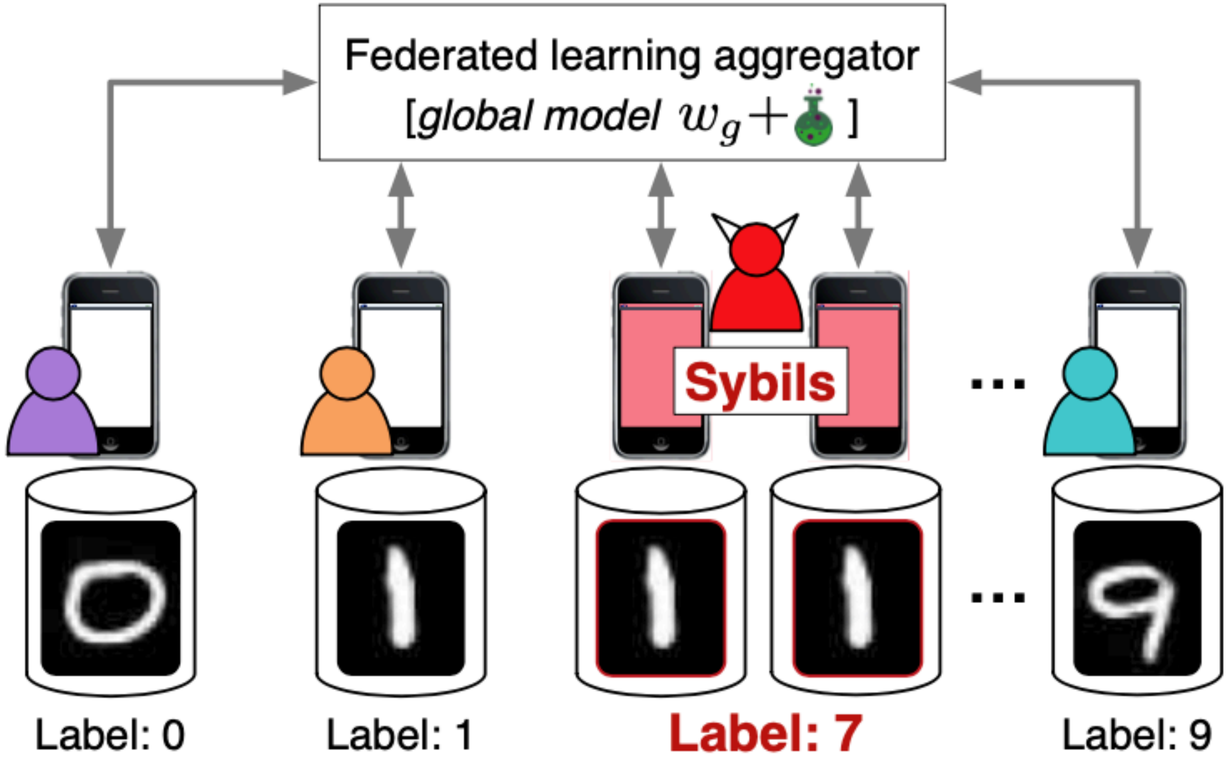
TARGETED POISONING ATTACKS ON FL

- FL에서 수집하는 측(서버)은
  학습 데이터를 보지 못하므로
  - 중독 공격(Poisoning attack)을
    모델 업데이트만으로 판단해야 함



SYBIL ATTACK IN FL

- 클라이언트들의 참여 및 떠남을 허가하는 시스템은
  - 시빌 공격에 취약
- 시빌 공격
  - 공격자가 여러 신원을 통해 영향력을 끼칠 수 있는 공격



FOOLSGOLD

FOOLSGOLD

- 오직 클라이언트의 SGD 결과에만 접근 가능함
- “시스템상 정직한 클라이언트의 비율” 가정과 상관 없는 방법 제안
- **학습 프로세스**로부터의 상태만 이용
  - 클라이언트에 대한 학습률을 조정하기 위해

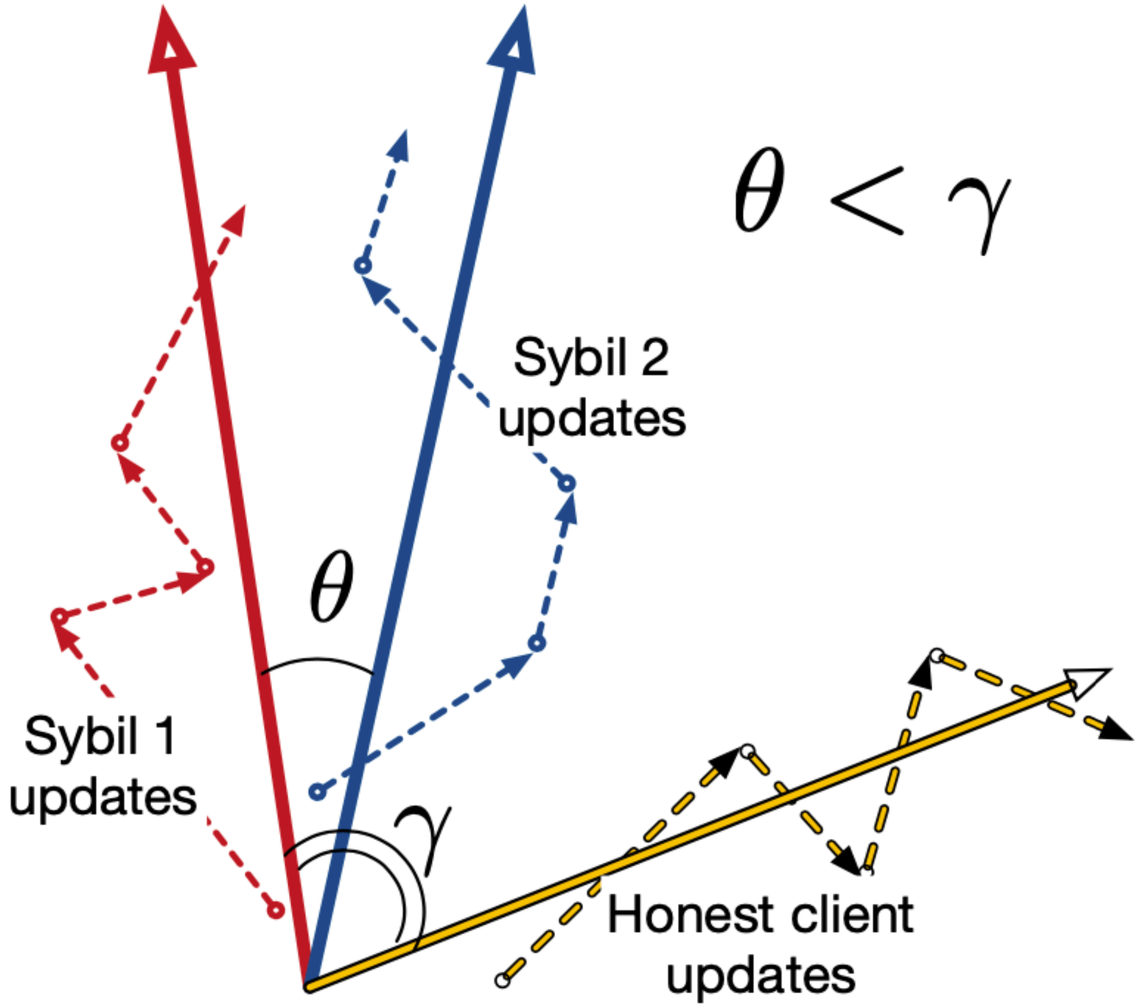
FOOLSGOLD

- 인사이트
  - 정직한 클라이언트와 시빌 간 구분이 가능
  - 그레디언트 업데이트의 양상을 보면 알 수 있음

FOOLSGOLD

- 연합 학습에서
  - 각 클라이언트의 학습 데이터는
    유니크한 분포를 가짐/공유되지 않음
  - 시빌은 공통의 목적을 가지고
    지속적으로 유사한 양상으로 업데이트함
- 코사인 유사도가 유사도를 대변함

$$cs_{ij} = cosine\_similarity(\sum_{t=1}^{T} \Delta_{i,t}, \sum_{t=1}^{T} \Delta_{j,t})$$



FOOLSGOLD

- 이러한 인사이트를 가지고
  - 각 클라이언트의 학습률을 각 반복마다 수정
  - 지속적으로 유사해 보이는 그레디언트 업데이트를 제안하는 클라이언트의 학습률을 줄임

FOOLSGOLD – FIVE GOALS

- 1. 시스템이 공격받지 않을 때, FL의 성능을 보존할 수 있어야 함
- 2. 유사한 방향을 향하는 클라이언트의 기여도를 평가 절하할 수 있어야 함
- 3. 중독 공격에 있는 시빌의 수가 증가하더라도 강인해야 함
- 4. 정직한 업데이트와 시빌의 업데이트를 구분할 수 있어야 함
  - 정직한 업데이트: SGD의 변칙성 때문에 등장하는 우연한 악의적 행로와
  - 시빌의 업데이트: 공격적 목적을 위해 행동하는
- 5. 클라이언트 또는 시빌의 수와 같은 외부 가정에 의존하지 않아야 함

ALGORITHM

ALGORITHM

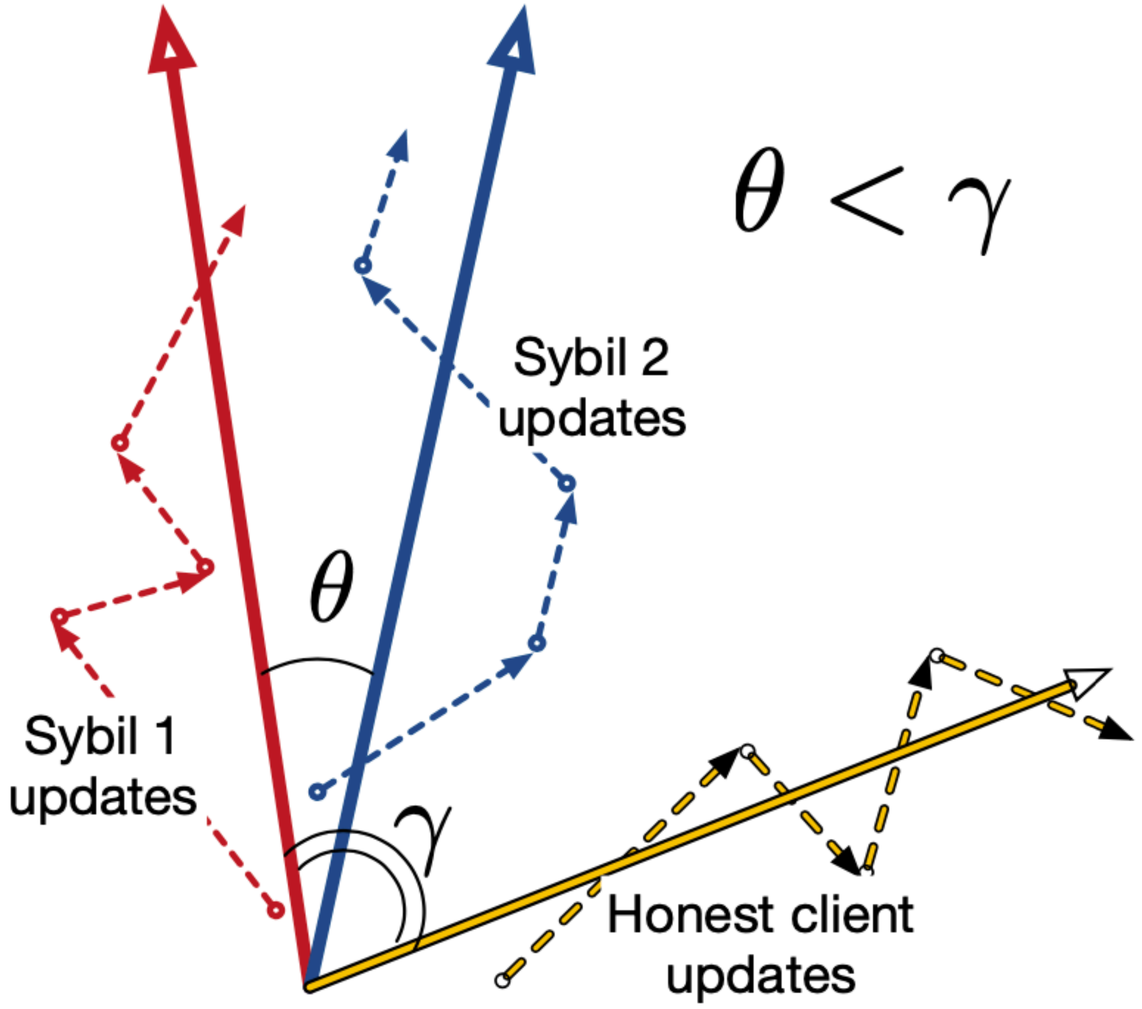
- FL 프로토콜에서
  - 그레디언트 업데이트는 라운드에 따른 동기적인 “모으기 -> 통합”의 연속
- FoolsGold에서
  - 클라이언트별 학습률 α_i를 도입
    - 어떠한 반복에서
    - 대표(indicative) 특징에서의 업데이트 유사도와
    - 과거 반복에서의 히스토리를 이용해 학습률을 조정

ALGORITHM

- 대표 특징
  - 글로벌 모델 출력 레이어의 모델 파라미터 크기로부터 찾을 수 있음
  - 출력 레이어의 파라미터는 예측 확률에 직접적으로 상관하므로
- 대표 특징 공간에서만 (코사인) 유사도를 살펴볼 것

ALGORITHM

- 히스토리 업데이트
  - 각 클라이언트의 업데이트 히스토리를 가짐
  - 과거에서부터 현재까지 통합된 형태로 저장
- 전체적인 기여의 유사도를 추정하는데 도움
  - 각 단계의 업데이트 방향은 다르지만
  - 전체적으로는 유사한 경우가 있음:



ALGORITHM

- 코사인 유사도로부터 정직한 클라이언트와 시빌을 구분할 수 있음은
  - 약한 가정
  - 정직한 클라이언트가 이러한 스킴으로부터 피해를 받을 수 있음
  - 관용(Pardoning)이 필요함

ALGORITHM

- 관용
  - 어떤 클라이언트 j와 (코사인) 유사했는데
  - j는 나보다 더 높은 유사도를 가진 클라이언트가 있으면
  - 나의 j에 대한 코사인 유사도를 비율에 맞게 조정해 줌

ALGORITHM

- 관용
  - 유사하지 않을 수록 학습률 α가 큼: 1에서 빼야 함
  - 이후, 정직한 노드는 그 기여에 패널티를 받지 않도록 조정

$$\alpha_i = 1 - \max_j(cs_i)$$

$$\alpha_i = \frac{\alpha_i}{\max_i(\alpha)}$$

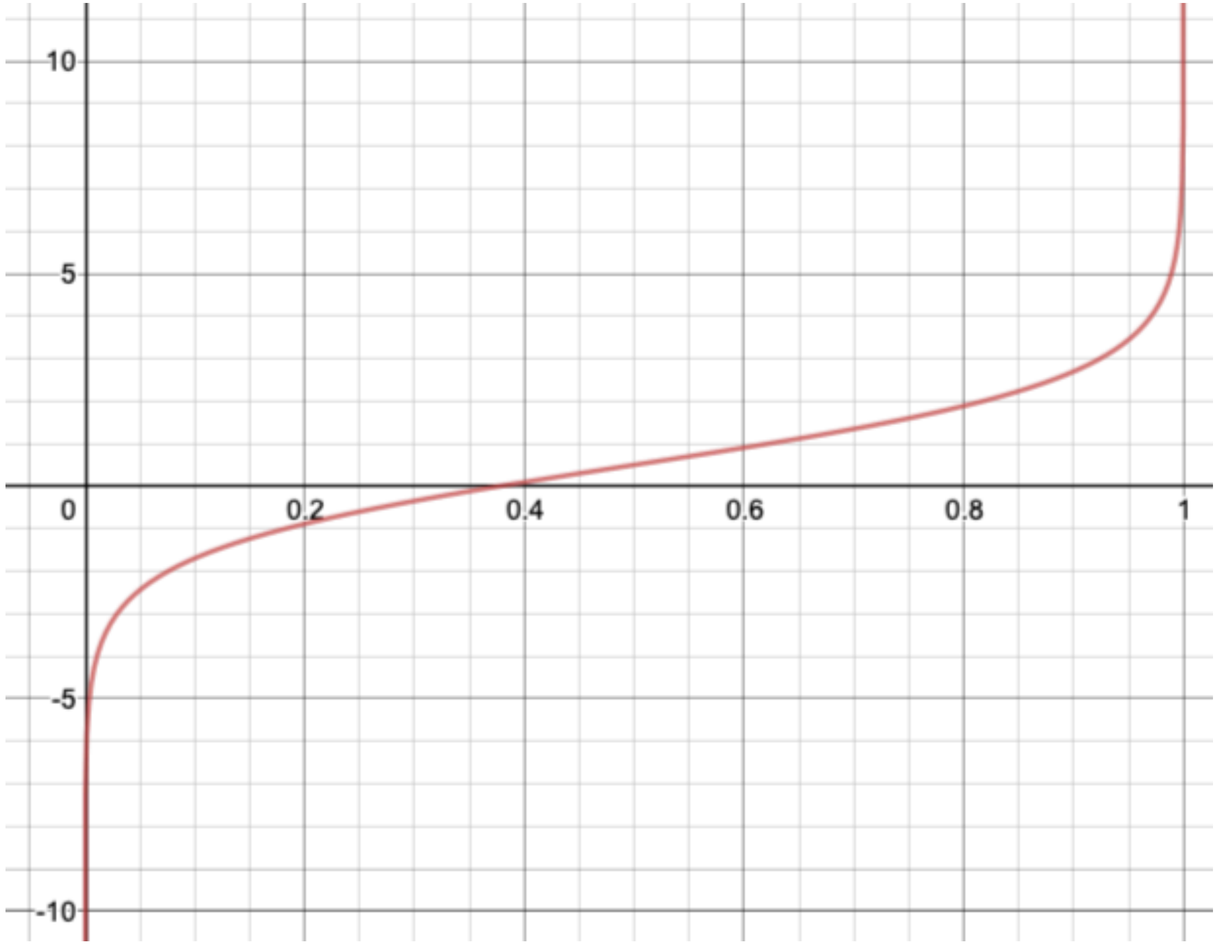
ALGORITHM

- 로그 변환 (Logit)
  - 코사인 유사도는 커봐야 1의 값을 가짐
  - 공격자가 영향력을 유지하면서 시빌의 수를 늘릴 수 있음
- 더 높은 값의 다양성을 주고 싶으며
  - 특히 양 극단은 좀 더 가파른 양상을 보이고 싶음

ALGORITHM

- 로그 변환 (Logit)
  - 시그모이드의 역함수
  - 의 중심을 0.5로 조정한 함수 사용:

$$\alpha_i = \kappa(\ln[\frac{\alpha_i}{1-\alpha_i}] + 0.5)$$



ALGORITHM

- 로그 변환 (Logit)
  - 이후 0-1 범위로 자른(clip) 다음
- 비로소 전체 그레디언트를 업데이트:

$$w_{t+1} = w_t + \sum_i \alpha_i \Delta_{i,t}$$

ALGORITHM

- 코사인 유사도 계산
  - 히스토리 간, 대표 특징 공간에서의
- 클라이언트별 최대 유사도 산출 (v_i)
- 관용
- 학습률 계산
  - 스케일링
  - 로그 변환 (Logit)
- 그레디언트 업데이트

Data: Global Model w^t and SGD update $\Delta_{i,t}$ from each client i at iteration t. Confidence parameter κ

for *Iteration* t **do**
  for *All clients* i **do**
    // Updates history
    Let H_i be the aggregate historical vector $\sum_{t=1}^{T} \Delta_{i,t}$
    // Feature importance
    Let S_t be the weight of indicative features at iteration t
    for *All other clients* j **do**
      Let cs_{ij} be the S_t-weighted cosine similarity between H_i and H_j
    end
    Let $v_i = \max_j(cs_i)$
  end
  for *All clients* i **do**
    for *All clients* j **do**
      // Pardoning
      if $v_j > v_i$ **then**
        $cs_{ij} *= v_i/v_j$
      end
    end
    // Row-size maximums
    Let $\alpha_i = 1 - \max_j(cs_i)$
  end
  // Logit function
  $\alpha_i = \alpha_i / \max_i(\alpha)$
  $\alpha_i = \kappa(\ln[(\alpha_i)/(1-\alpha_i)] + 0.5)$
  // Federated SGD iteration
  $w_{t+1} = w_t + \sum_i \alpha_i \Delta_{i,t}$
end

EVALUATION

EVALUATION

- 평가에 사용한 데이터셋:
  - 숫자 손글씨, 얼굴, 네트워크 침입 패턴, 그리고 제품 리뷰 텍스트

Dataset	Train Set Size	Classes	Features	Model Used
MNIST	60,000	10	784	1-layer softmax
VGGFace2	7,380	10	150,528	SqueezeNet, VGGNet11
KDDCup	494,020	23	41	1-layer softmax
Amazon	1,500	50	10,000	1-layer softmax

- 학습 데이터를 겹치지 않는 non-IID로 구분

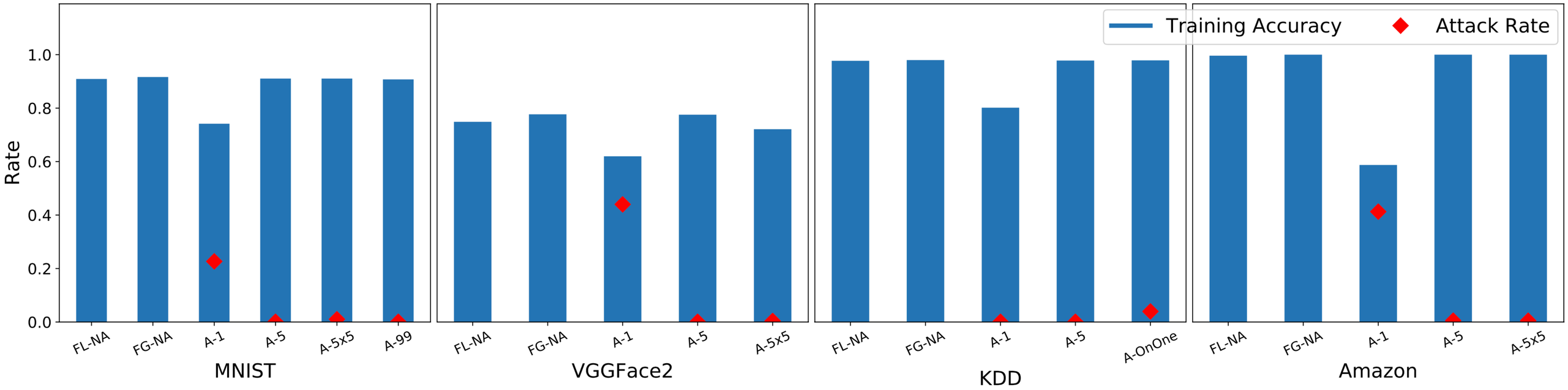
EVALUATION

- 공격 시나리오:

Attack	Description	Dataset
A-1	Single client attack.	All
A-5	5 clients attack.	All
A-5x5	5 sets of 5 clients, concurrent attacks.	MNIST, Amazon, VGGFaces2
A-OnOne	5 clients executing 5 attacks on the same target class.	KDDCup99
A-99	99% sybils, performing the same attack.	MNIST

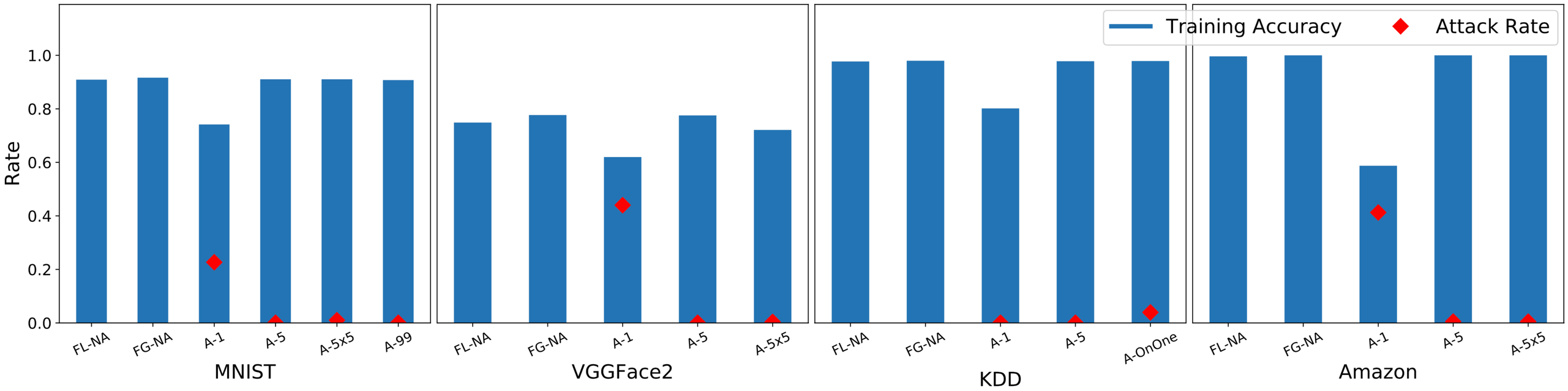
EVALUATION – RESULT

- 공격자가 없는 환경도 함께 비교
  - FL-NA: 오리지널 FL
  - FG-NA: PoolsGold



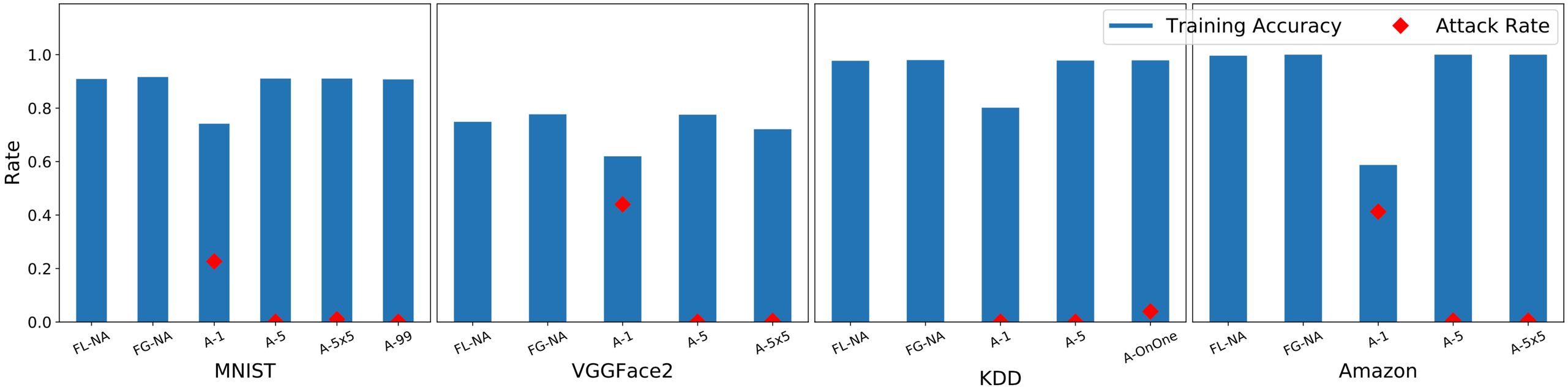
EVALUATION – RESULT

- 높은 정확도를 유지하며
  - Attack Rate: 공격 target이 부정확하게 분류된 비율 / Accuracy: 잘 분류한 비율
- 효과적으로 공격을 방어



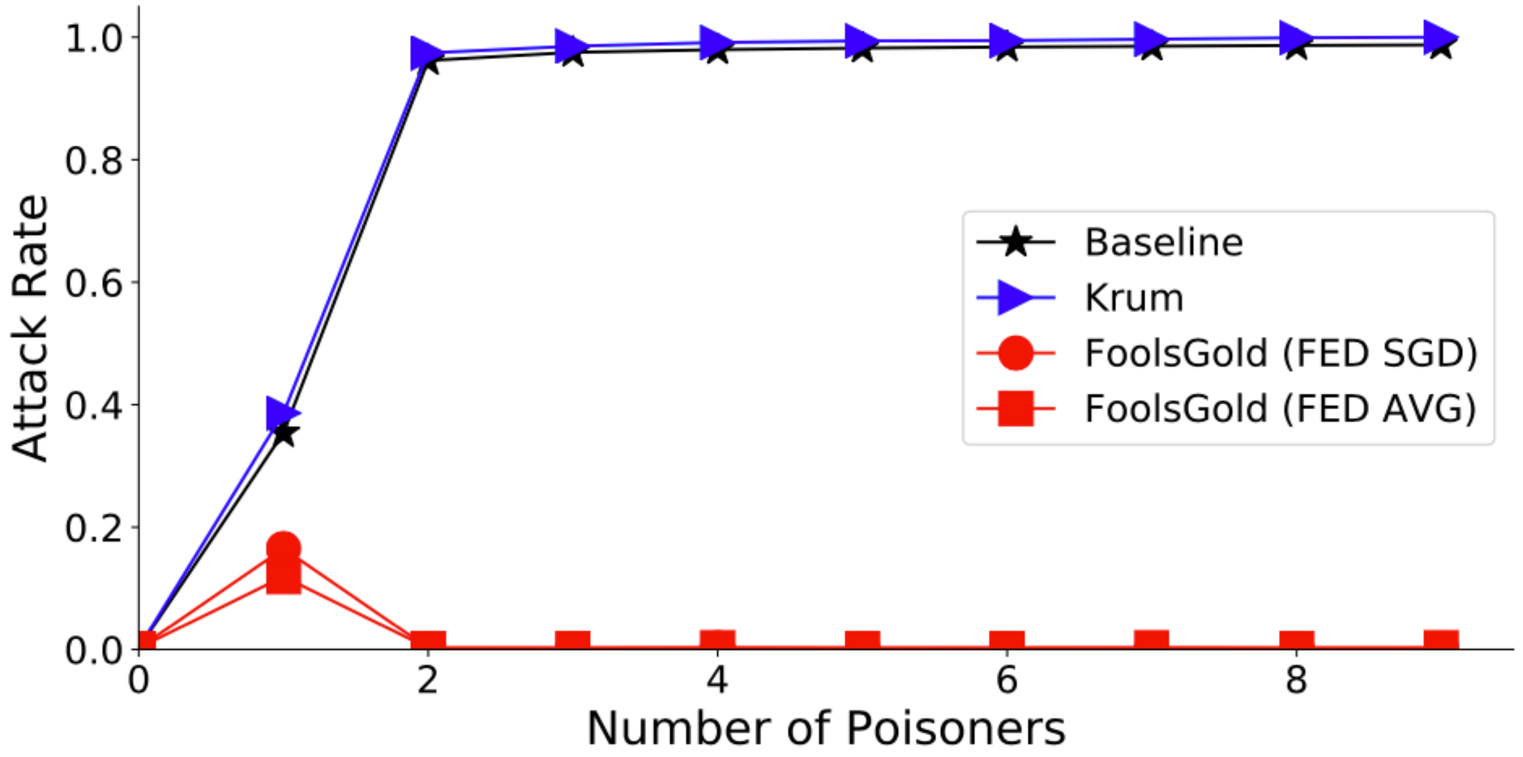
EVALUATION – RESULT

- A-1 공격에서 가장 좋지 못한 성능
  - 단 하나의 공격자: 시빌이라 볼 수 있는가?
  - 정직한 클라이언트와 구별이 불가능



EVALUATION

- 사전 연구들과의 비교:



WHAT IF
HE KNOWS FOOLSGOLD

WHAT IF THE ATTACKER KNOWS FOOLSGOLD?

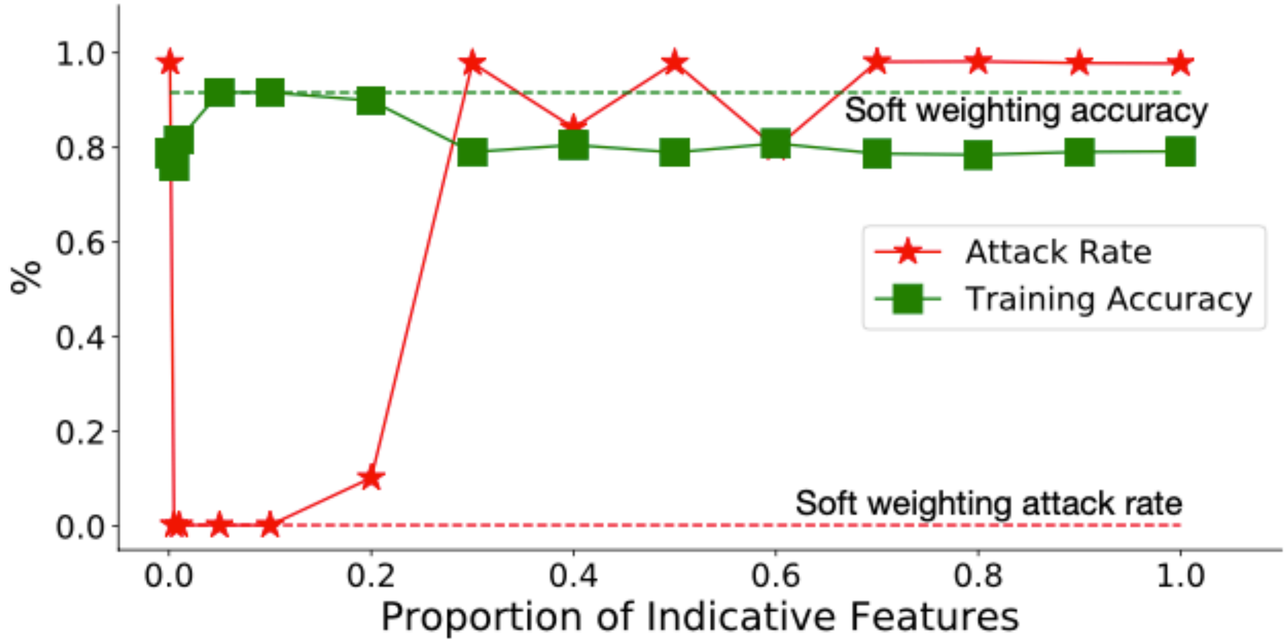
- 만약 공격자가 FoolsGold 알고리즘에 대해 이미 알고 있다면?
- 트레이드-오프가 존재
  - 공격자가 서로 덜 유사하게 업데이트 하면
    - 검출의 가능성을 줄이기 위해
  - 공격에 대한 집중도가 떨어짐
    - 공격 효용이 낮아짐

WHAT IF THE ATTACKER KNOWS FOOLSGOLD?

- 저자들은 FoolsGold를 전복시키기 위한 네 가지 시도를 고안하고 평가해 봄
- 1. 악의적인 데이터와 정상 데이터를 섞음
- 2. 시빌의 학습 배치 사이즈를 변경
- 3. 업데이트에 노이즈를 더해 혼란스럽게 함
- 4. 비주기적이고 적응적이게 오염된 업데이트를 전송
- 결론적으로
  - 잘 동작하며, 이전 연구들의 성능을 압도

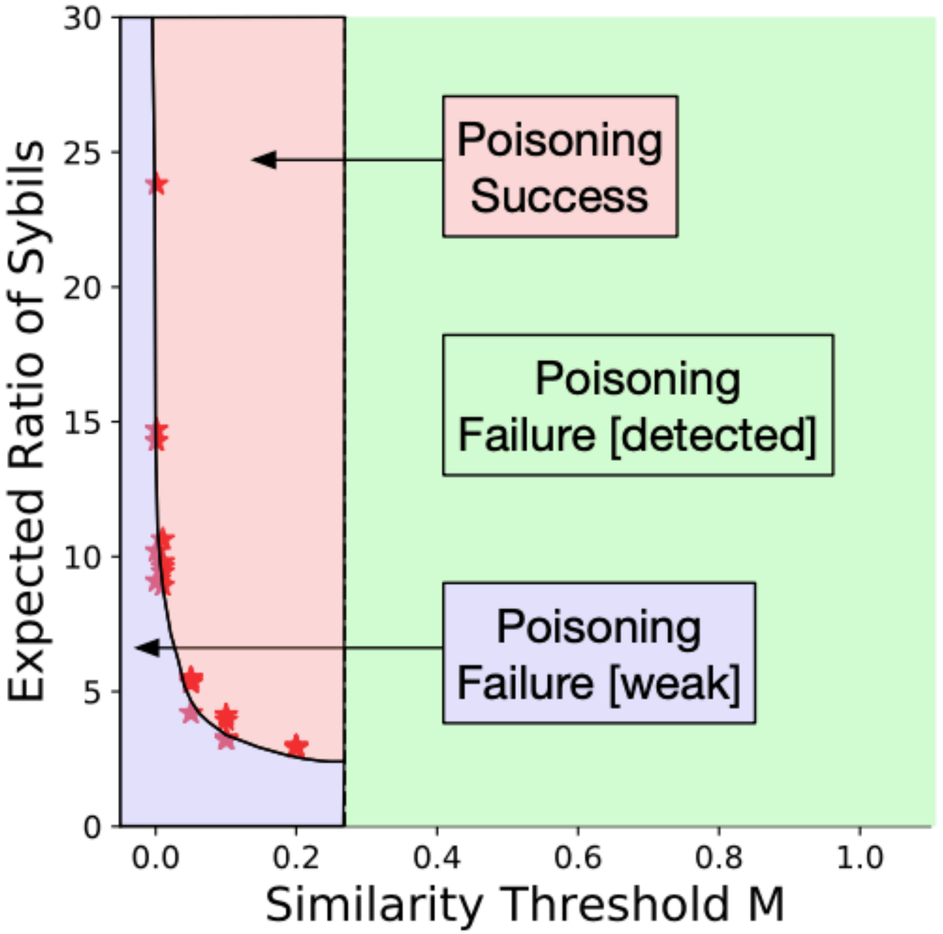
WHAT IF THE ATTACKER KNOWS FOOLSGOLD?

- 3. 업데이트에 노이즈를 더해 혼란스럽게 함
  - 유사도 계산에 모든 feature들을 사용하면 중독 공격에 취약
  - 선출한 대표 특징이 전체의 0.1 (10%) 보다 낮으면 효과적
  - 너무 낮으면 (0.01) 훈련 정확도에 영향을 끼침



WHAT IF THE ATTACKER KNOWS FOOLSGOLD?

- 4. 비주기적이고 적응적이게 오염된 업데이트를 전송
  - 시빌 간의 코사인 유사도와 (x축)
  - 정직한 클라이언트 대비 시빌의 비 (y축)
  - 에 따른 공격 성공/실패
- M > 0.27 부터는 검출됨
- 많은 수의 공격자가 필요
  - 그러나 정직한 노드 수 등의 정보는 공격자가 모름



CONCLUSION

CONCLUSION

- Machine Learning의 탈중앙화
  - 프라이버시의 강화
  - 확장성 문제 해결책
- Federate Learning
  - State of the Art한 제안

CONCLUSION

- 탈중앙화는 공격자들에게 문을 열어준 격
  - 시빌로부터의 중독 공격
- FoolsGold
  - 클라이언트 기여도 유사도를 이용해
  - 그러한 공격을 비효율적/비효용적이게 만듦
- FoolsGold는 다양한 종류의 공격을 완화하고
  - 심지어 시빌이 정직한 클라이언트들보다 많아도 잘 작동

SYBILS IN FL

MITIGATING SYBILS IN FEDERATED LEARNING