ADAPTIVE FL

ADAPTIVE FEDERATED OPTIMIZATION

REFERENCE

- Reddi, Sashank, et al.
  "Adaptive Federated Optimization."
  arXiv preprint arXiv:2003.00295 (2020).

ABSTRACT

ABSTRACT

- 연합 학습(Federated Learning, FL)은 분산 머신 러닝 패러다임
  - 많은 수의 클라이언트들이 중앙 서버에게
  - 자신의 훈련 데이터를 공유하지 않고 학습에 협력

ABSTRACT

- 클라이언트 데이터셋들의 이종적인 특성 때문에
  - 연합 평균(Federated Averaging, FEDAVG)과 같은
  - 표준 연합 최적화 방법은 조율이 힘든 경우에 종종 처함
  - 수렴 양상이 바람직못한 경우들

ABSTRACT

- 비-연합 환경에서는
  - 적응형(adaptive) 최적화 기법이 이러한 이슈를 잘 다룸

ABSTRACT

- 저자들은 적응형 최적화기의 연합 버전을 제안
  - ADAGRAD
  - ADAM
  - YOGI

ABSTRACT

- 이종적인 환경에서의 수렴성 분석
  - 클라이언트의 이종성과 통신 효율 사이의 상호 영향을 조명
- 적응형 최적화기들이 연합 학습의 성능을 매우 향상

INTRODUCTION

INTRODUCTION

- FL은
  - 중앙 서버의 조율 하에서
  - 많은 수의 클라이언트가 모델 학습에 협력하는
    - 엣지(edge) 디바이스
    - 여러 기관 등
  - 머신 러닝 패러다임

INTRODUCTION

- FL의 핵심은
  - 원(raw) 클라이언트의 데이터가
  - 절대로 서버나 다른 클라이언트에게
  - 공유되지 않는다는 것

INTRODUCTION

- 기존의 분산 최적화와는 달리
  - FL은 이종적인 데이터를 다룬다는 점이
  - 문제이자 쟁점

INTRODUCTION

- 미니배치(mini-batch) SGD를 사용하는
  - 표준적인 접근법은
  - 높은 통신 비용을 초래하므로
  - 많은 FL 환경에 적합하지 않음

INTRODUCTION

- 이에 FL을 위한 많은 최적화 기법은
  - 로컬(local) 클라이언트 업데이트를 활용
  - 클라이언트가 통신 전에 로컬 모델을 수 차례 업데이트
- FL에 널리 쓰이는 로컬 최적화 기법 중 하나인
  - FEDAVG

INTRODUCTION

- FEDAVG의 각 라운드에서
  - 클라이언트의 일부는 병렬적으로
  - SGD의 몇 에폭을 수행
- 클라이언트는 모델 업데이트 정보를 서버와 통신
  - 서버는 평균을 통해 새 글로벌 모델을 계산

INTRODUCTION

- FEDAVG가 큰 성공을 거뒀음에도
  - 최근의 연구들로부터 단점이 조명됨
- 본 논문에서는 두 문제를 다룸
  - 클라이언트 드리프트(client drift)
  - 적응적 학습률(adaptive learning rate)의 부재

INTRODUCTION

- 클라이언트 드리프트(client drift)
  - 이종적 환경에서 여러 로컬 SGD 에폭의 수행은
  - 글로벌한 최적 모델로부터 멀리 떨어지게(drift) 만듦

INTRODUCTION

- 적응적 학습률(adaptive learning rate)의 부재
  - 학습 과정 중의 Heavy-tail 노이즈 분포 등
    - Zhang, Jingzhao, et al.
      "Why ADAM beats SGD for attention models."
      arXiv preprint arXiv:1912.03194 (2019).
  - 필연적으로 적응적 학습률이 필요한 환경들이 있음

INTRODUCTION

- 적응적 학습률(adaptive learning rate)의 부재
  - 필연적으로 적응적 학습률이 필요한 환경들이 있음
  - 이는 과거의 반복(iteration)들로부터 정보에 기반한
  - 그래디언트 기반의 최적화를 수행하도록 함

INTRODUCTION

- 그러나 정보 공유를 하지 않는 FL의 기본 속성에 따르면
  - 적응형 학습률은 도전적인 문제임

INTRODUCTION

- 클라이언트들이 모델을 업데이트
  - 여러 에폭을
  - 임의의 클라이언트 최적화기를 이용해
  - 로컬 데이터의 손실(loss)을 줄이기 위해

INTRODUCTION

- 서버는 글로벌 모델을 업데이트
  - 모델 업데이트의 평균으로부터
  - 그라디언트 기반
  - 서버 최적화기를 이용
  - 클라이언트들에 걸친 손실을 최소화하기 위해

INTRODUCTION

- 클라이언트와 서버의 학습률을 분리하는 자연스러운 방법을 제시
  - FEDAVG는 클라이언트와 서버가 SGD를 쓰고, 서버의 학습률이 1인 경우
  - 즉, FEDAVG의 일반화
- 클라이언트와 서버가 다른 학습률을 활용할 수 있도록 함
  - 클라이언트 드리프트를 해결할 방법을 제공

INTRODUCTION

- 적응형 방법들인
  - ADAGRAD, ADAM, YOGI
  - 등은 머신 러닝 커뮤니티에서 유명한 기법들
  - 학습률을 튜닝할 수고를 덜고
  - Heavy-tail 노이즈 분포 등을 잘 다룸

INTRODUCTION

- 이들 프레임워크 위에서
  - FL을 위한 적응형 최적화 기술들을 개발

CONTRIBUTIONS

- 일반화된 FL 최적화 프레임워크
  - 서버와 클라이언트 최적화기 양 쪽에 대한
  - FEDAVG를 포함한 FL의 주 최적화 기법들을 캡슐화/일반화

CONTRIBUTIONS

- 이 프레임워크에 기반해 FL을 위한 적응형 최적화 기법들을 제안
  - 또한, 수렴성 분석
  - “최초”의 FL을 위한 적응형 최적화 기법

CONTRIBUTIONS

- 6개의 실험을 통해
  - 모멘텀(momentum)의 중요성과
  - 클라이언트의 학습률 감소 스케쥴의
  - 중요성을 보임
- 제안하는 방법을 통해
  - 기존 FEDAVG 대비
  - 4개의 실험에서 극적인 향상을 보임

FL&FEDAVG

FEDERATED LEARNING

- 주로 다음 문제를 풀고자 함:

$$\arg\min_{x \in \mathbb{R}^d} f(x) = \frac{1}{m} \sum_{i=1}^{m} F_i(x)$$

  - 파라미터 x
  - $F_i(x) = \mathbb{E}_{z \sim D_i}[f_i(x, z)]$: i번째 클라이언트의 손실 함수
    - Non-convex
    - $f(x)$ 역시 non-convex
  - D_i : i번째 클라이언트의 데이터 분포

FEDERATED LEARNING

- 가정들
  - 1. 립시츠 그라디언트(Lipschitz Gradient)
  - 2. 유계 분산(Bounded Variance)
  - 3. 유계 그라디언트(Bounded Gradients)
- 1과 3은 non-convex 최적화 얘기에서는 흔한 가정
- 서로 다른 클라이언트 데이터셋에 대한 추가 가정은 논의하지 않음
  - 가정 2는 목적 함수의 분산에 관한 내용
  - 여러 FL 연구에서 두는 가정

FEDERATED LEARNING

- 1. 립시츠 그라디언트(Lipschitz Gradient)
- 함수 F_i 는 L-smooth
- 모든 x, y에 대해 $\|\nabla F_i(x) - \nabla F_i(y)\| \leq L\|x - y\|$

FEDERATED LEARNING

- 2. 유계 분산(Bounded Variance)
- 함수 F_i 는 σ_l-bounded 로컬 분산을 가짐
  - $\mathbb{E}[\|\nabla[f_i(x,z)]_j - [\nabla F_i(x)]_j\|^2] = \sigma_{l,j}^2$
- 글로벌 분산 역시 유계
  - $\frac{1}{m}\sum_{i=1}^{m}\|\nabla[F_i(x)]_j - [\nabla f(x)]_j\|^2 \leq \sigma_{g,j}^2$

FEDERATED LEARNING

- 2. 유계 분산(Bounded Variance)
- 만일 $\sigma_g^2 = \Sigma\sigma_{g,j}^2$ 에서 $\sigma_g = 0$ 이라면 i.i.d.한 상황
  - 독립항등분포(independent and identically distributed)

FEDERATED LEARNING

- 3. 유계 그라디언트(Bounded Gradients)
- 함수 $f_i(x, z)$ 는 G-bounded 그라디언트를 가짐
- 어떠한 i, x, z에 대해, 모든 j에 대해 $|[\nabla f_i(x, z)]_j| \leq G$

FEDERATED LEARNING

- 주로 다음 문제를 풀고자 함:

$$\arg\min_{x \in \mathbb{R}^d} f(x) = \frac{1}{m} \sum_{i=1}^{m} F_i(x)$$

- 가장 유명한 접근법은 FEDAVG

FEDERATED LEARNING

- FEDAVG 기법
  - 각 라운드에서 클라이언트들의 서브셋이 선택되고(주로 무작위로)
  - 서버가 그들에게 글로벌 모델을 브로드캐스트
  - 병렬적으로 클라이언트들이 고유의 손실 함수를 통해 SGD를 수행
  - 서버에게 모델을 전송
  - 서버는 로컬 모델들의 평균을 구해 글로벌 모델을 업데이트

FEDERATED LEARNING

- FEDAVG 기법

$$x_i^t = \mathrm{SGD}_K(x_t, \eta_l, F_i)$$

  - 그라디언트 $\nabla f_i(x, z)$에 대해
  - 로컬 학습률 η_l 을 통해
  - x_t 에서 시작해
  - K 스텝만큼 SGD

FEDERATED LEARNING

- FEDAVG 기법

Algorithm 1 Simplified FEDAVG

Initialization: x_0
for $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    $x_i^t = \mathrm{SGD}_K(x_t, \eta_l, F_i)$
  $x_{t+1} = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} x_i^t$

FEDERATED LEARNING

- FEDAVG 기법의 재작성:

$$x_{t+1} = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} x_{i,K}^t = x_t - \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \left(x_t - x_{i,K}^t \right)$$

FEDERATED LEARNING

- $\Delta_i^t := x_{i,K}^t - x_t$
- $\Delta^t = \frac{1}{|S|}\Sigma\Delta_i^t$
- 서버는
  - 평균을 통한 수도-그라디언트(pseudo-gradient)를
  - 학습률 1로 SGD함과 동일

FEDERATED LEARNING

- 수도-그라디언트 관점에서 일반화

Algorithm 2 Generalized FedAvg

Initialization: x_0
for $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = \text{ClientOpt}(x_{i,k}^t, g_{i,k}^t, \eta_l, t)$
    $\Delta_i^t = x_{i,K}^t - x_t$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $x_{t+1} = \text{ServerOpt}(x_t, -\Delta_t, \eta, t)$

FEDERATED LEARNING

- 수도-그라디언트 관점에서 일반화
- CLIENTOPT는 로컬 데이터에 기반한 목적을 최적화함에 초점
- SERVEROPT는 글로벌 관점

ADAPTIVE FEDERATED OPTIMIZATION

ADAPTIVE FEDERATED OPTIMIZATION

- 일반화한 알고리즘 2에 기반
  - 다음과 같이 세팅
  - SERVEROPT = {ADAGRAD, ADAM, YOGI}
  - CLIENTOPT = SGD
- 다음 알고리즘 3과 4는
  - 각각 FEDADAGRAD, FEDYOGI와 FEDADAM에 해당:

ADAPTIVE FEDERATED OPTIMIZATION

Algorithm 3 FEDADAGRAD

Initialization: x_0, $\tau > 0$ and $v_{-1} \geq \tau^2$
for $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = x_{i,k}^t - \eta_l g_{i,k}^t$
    $\Delta_i^t = x_{i,K}^t - x_t$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $v_t = v_{t-1} + \Delta_t^2$
  $x_{t+1} = x_t + \eta \frac{\Delta_t}{\sqrt{v_t} + \tau}$

ADAPTIVE FEDERATED OPTIMIZATION

Algorithm 3 FEDADAGRAD

Initialization: x_0, $\tau > 0$ and $v_{-1} \geq \tau^2$
for $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = x_{i,k}^t - \eta_l g_{i,k}^t$ CLIENTOPT = SGD
    $\Delta_i^t = x_{i,K}^t - x_t$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $v_t = v_{t-1} + \Delta_t^2$
  $x_{t+1} = x_t + \eta \frac{\Delta_t}{\sqrt{v_t} + \tau}$

ADAPTIVE FEDERATED OPTIMIZATION

Algorithm 3 FEDADAGRAD

Initialization: x_0, $\tau > 0$ and $v_{-1} \geq \tau^2$
for $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = x_{i,k}^t - \eta_l g_{i,k}^t$
  $\Delta_i^t = x_{i,K}^t - x_t$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $v_t = v_{t-1} + \Delta_t^2$
  $x_{t+1} = x_t + \eta \frac{\Delta_t}{\sqrt{v_t} + \tau}$

통합 후 평균

ADAPTIVE FEDERATED OPTIMIZATION

Algorithm 3 FEDADAGRAD

Initialization: x_0, $\tau > 0$ and $v_{-1} \geq \tau^2$
for $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = x_{i,k}^t - \eta_l g_{i,k}^t$
    $\Delta_i^t = x_{i,K}^t - x_t$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $v_t = v_{t-1} + \Delta_t^2$
  $x_{t+1} = x_t + \eta \frac{\Delta_t}{\sqrt{v_t} + \tau}$

ADAGRAD

ADAPTIVE FEDERATED OPTIMIZATION

Algorithm 4 **FEDYOGI** (and **FEDADAM**)

Initialization: $x_0, v_{-1} \geq \tau^2$, decay $\beta_2 \in (0, 1)$
for $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = x_{i,k}^t - \eta_l g_{i,k}^t$
    $\Delta_i^t = x_{i,K}^t - x_{t-1}$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $v_t = v_{t-1} - (1-\beta_2)\Delta_t^2 \operatorname{sign}(v_{t-1} - \Delta_t^2)$ **(FEDYOGI)**
  $v_t = \beta_2 v_{t-1} + (1-\beta_2)\Delta_t^2$ **(FEDADAM)**
  $x_{t+1} = x_t + \eta \frac{\Delta_t}{\sqrt{v_t} + \tau}$

ADAPTIVE FEDERATED OPTIMIZATION

Algorithm 4 **FEDYOGI** (and **FEDADAM**)

Initialization: $x_0, v_{-1} \geq \tau^2$, decay $\beta_2 \in (0, 1)$
for $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = x_{i,k}^t - \eta_l g_{i,k}^t$
    $\Delta_i^t = x_{i,K}^t - x_{t-1}$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $v_t = v_{t-1} - (1-\beta_2)\Delta_t^2 \operatorname{sign}(v_{t-1} - \Delta_t^2)$ **(FEDYOGI)**
  $v_t = \beta_2 v_{t-1} + (1-\beta_2)\Delta_t^2$ **(FEDADAM)**
  $x_{t+1} = x_t + \eta \frac{\Delta_t}{\sqrt{v_t} + \tau}$

CLIENTOPT = SGD

ADAPTIVE FEDERATED OPTIMIZATION

Algorithm 4 **FEDYOGI** (and **FEDADAM**)

Initialization: $x_0, v_{-1} \geq \tau^2$, decay $\beta_2 \in (0, 1)$
for $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = x_{i,k}^t - \eta_l g_{i,k}^t$
    $\Delta_i^t = x_{i,K}^t - x_{t-1}$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$

통합 후 평균

  $v_t = v_{t-1} - (1-\beta_2)\Delta_t^2 \operatorname{sign}(v_{t-1} - \Delta_t^2)$ **(FEDYOGI)**
  $v_t = \beta_2 v_{t-1} + (1-\beta_2)\Delta_t^2$ **(FEDADAM)**
  $x_{t+1} = x_t + \eta \frac{\Delta_t}{\sqrt{v_t} + \tau}$

ADAPTIVE FEDERATED OPTIMIZATION

Algorithm 4 **FEDYOGI** (and **FEDADAM**)

Initialization: $x_0, v_{-1} \geq \tau^2$, decay $\beta_2 \in (0, 1)$
for $t = 0, \cdots, T-1$ **do**
  Sample subset $\mathcal{S}$ of clients
  $x_{i,0}^t = x_t$
  for each client $i \in \mathcal{S}$ **in parallel do**
    for $k = 0, \cdots, K-1$ **do**
      Compute an unbiased estimate $g_{i,k}^t$ of $\nabla F_i(x_{i,k}^t)$
      $x_{i,k+1}^t = x_{i,k}^t - \eta_l g_{i,k}^t$
    $\Delta_i^t = x_{i,K}^t - x_{t-1}$
  $\Delta_t = \frac{1}{|\mathcal{S}|} \sum_{i \in \mathcal{S}} \Delta_i^t$
  $v_t = v_{t-1} - (1-\beta_2)\Delta_t^2 \operatorname{sign}(v_{t-1} - \Delta_t^2)$ **(FEDYOGI)**
  $v_t = \beta_2 v_{t-1} + (1-\beta_2)\Delta_t^2$ **(FEDADAM)**
  $x_{t+1} = x_t + \eta \frac{\Delta_t}{\sqrt{v_t} + \tau}$

YOGI 또는 ADAM

ADAPTIVE FEDERATED OPTIMIZATION

- 수렴성 분석
  - 생략

EXPERIMENTAL EVALUATION

EXPERIMENTAL EVALUATION

- CIFAR-100
  - 500명의 클라이언트에게 트레이닝 데이터를 무작위로 할당
  - 보다 현실적인 이종성을 주기 위해
  - 계층적 잠재 디리클레 할당(Latent Dirichlet Allocation, LDA) 프로세스를 사용
- 수정된 ResNet-18을 사용
  - 배치 정규화 레이어를 그룹 정규화(group normalization) 레이어로 대체
  - Feature의 일부를 묶어 표준화

EXPERIMENTAL EVALUATION

- EMNIST
  - 숫자와 영문자 데이터셋, 총 62개 분류
  - FL을 위해 저자에 따라 데이터 분류: 글씨체가 서로 다르므로 이종적
- 오토인코더(Autoencoder, AE)
  - MNIST 오토인코더 사용
- 문자 인식(Character Recognition, CR)
  - CNN 모델

EXPERIMENTAL EVALUATION

- Shakespeare
  - 셰익스피어의 글들
  - 다음 ASCII 문자를 예측
  - RNN으로 임베딩, LSTM으로 연결

EXPERIMENTAL EVALUATION

- StackOverflow
  - Q&A와 태그를 포함한 메타데이터 제공
  - 342,477 유니크한 사용자를 포함, 각자를 클라이언트로 사용
- 태그 예측
  - 로지스틱 회귀(Logistic Regression, LR)
- 다음 단어 예측 (Next-word Prediction, NWP)
  - RNN으로 저차원 임베딩, LSTM으로 연결

EXPERIMENTAL EVALUATION

Dataset	Clients	Examples
CIFAR-100	500	50,000
EMNIST-62	3,400	671,585
Shakespeare	715	16,068
StackOverflow	342,477	135,818,730

IMPLEMENTATION

- TensorFlow Federated 사용
- 클라이언트 샘플링
  - 모든 훈련 클라이언트들에 대해 uniformly random하게
  - 라운드별로 교체

IMPLEMENTATION

- 각 클라이언트들이 K 상수 스텝을 수행하는 대신
  - E 클라이언트 에폭을 수행
  - 각 클라이언트의 데이터셋에 대해
  - 각 라운드마다

IMPLEMENTATION

- 가중 평균(weighted average)를 사용
  - 클라이언트의 훈련 샘플의 수에 따른
  - Uniform한 평균 대비 일반적으로 좋은 성능

OPTIMIZER & HYPERPARAMETER TUNING

- FEDAVG 대비 FEDADAGRAD, FEDADAM, FEDYOGI를 비교
  - 서버의 학습률이 조정된다는 특징
  - 학습률이 1 고정이 아님
- 서버 최적화기가
  - 조정된 학습률과
  - 모멘텀 파라미터 0.9를 가지는 SGD와도 비교
  - FEDAVGM으로 표기

OPTIMIZER & HYPERPARAMETER TUNING

- 서버 학습률 η 와
- 클라이언트의 저마다의 학습률 η_l 을
  - 적당한 크기의 그리드(grid)로부터 선출
- 가령 CIFAR-100의 경우:

$$\eta_l \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}$$

$$\eta \in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$$

OPTIMIZER & HYPERPARAMETER TUNING

- 배치 크기는 고정
  - 중앙화된 훈련과 비교할 때,
  - 마찬가지로 같은 배치 크기를 사용

Task	Batch size
CIFAR-100	20
EMNIST AE	20
EMNIST CR	20
Shakespeare	4
StackOverflow LR	100
StackOverflow NWP	16

OPTIMIZER & HYPERPARAMETER TUNING

- 실제 디바이스-간 FL 개발에서는
  - 통신 대역폭이 기본 제약이 됨
- 모든 알고리즘들이 동일한 사이즈의 오브젝트를 주고받기 때문에
  - 통신의 횟수를 대략적으로 추산

RESULT CONSTANT η

RESULT

- 결과를 크게 두 부류로 분석
  - 클라이언트 및 서버의 학습률이 상수
  - 클라이언트 학습률을 라운드 수에 따라 가변

CONSTANT LEARNING RATES

- 각 최적화기에 대해
  - 클라이언트 및 서버의 학습률을
  - 지난 100 라운드에 대한
  - 평균 검증 성능으로 선택

CONSTANT LEARNING RATES

- 마지막 100 라운드 성능에 대한 요약:
  - 1-4행: 정확도 - 높으면 좋음
  - 5행: 확률 (로지스틱 회귀) - 1에 가까우면 좋음
  - 6행: MSE - 낮으면 좋음

FED...	ADAGRAD	ADAM	YOGI	AVGM	AVG
CIFAR-100	23.9	**42.3**	41.6	37.3	26.5
EMNIST CR	**85.7**	**86.0**	**86.1**	**86.3**	85.9
SHAKESPEARE	57.1	**57.4**	**57.6**	**57.5**	57.0
STACKOV... NWP	11.3	**22.1**	**22.2**	13.7	9.5
STACKOV... LR	**0.68**	0.62	0.64	0.22	0.19
EMNIST AE	7.29	16.99	**0.98**	**1.21**	2.63

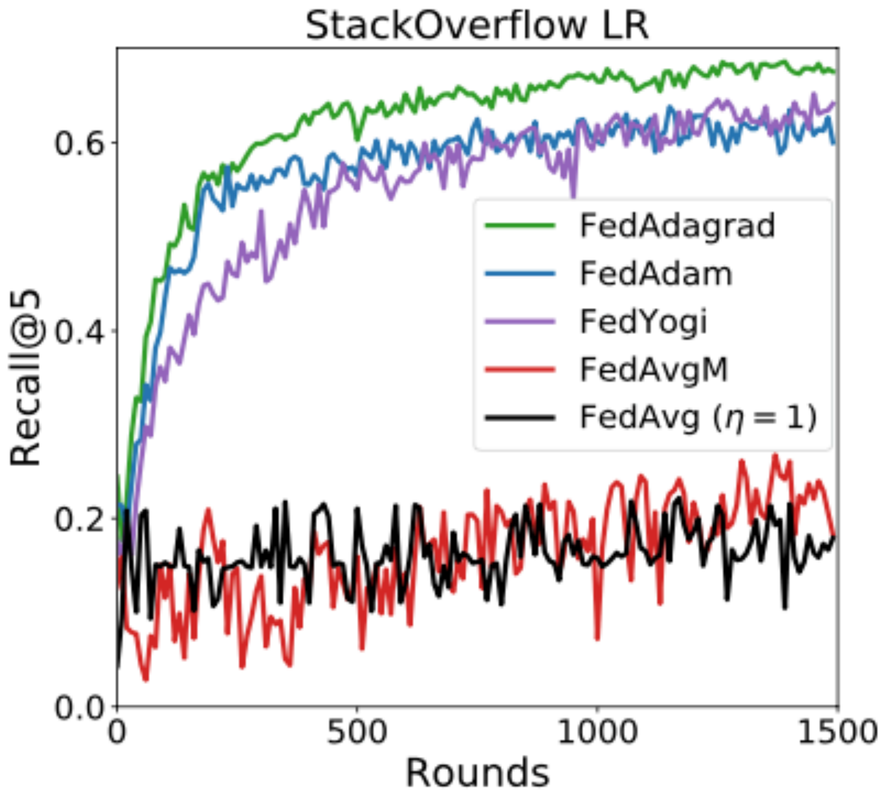
CONSTANT LEARNING RATES

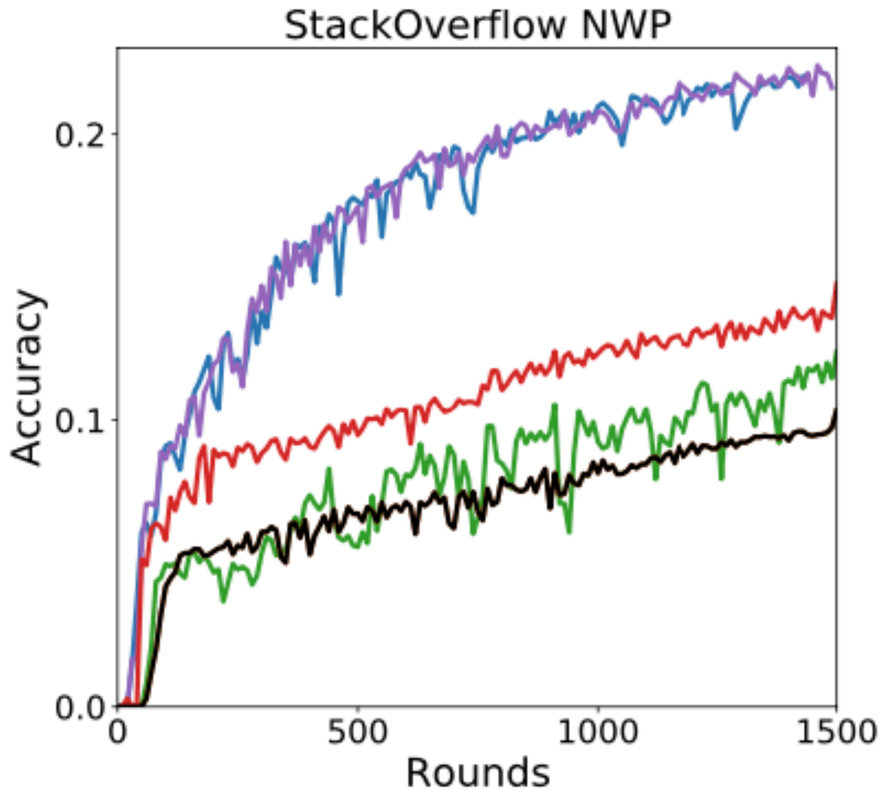
- 적응형 최적화만 사용해도 FEDAVG의 성능이 크게 오름

FED...	ADAGRAD	ADAM	YOGI	AVGM	AVG
CIFAR-100	23.9	**42.3**	41.6	37.3	26.5
EMNIST CR	**85.7**	**86.0**	**86.1**	**86.3**	85.9
SHAKESPEARE	57.1	**57.4**	**57.6**	**57.5**	57.0
STACKOV... NWP	11.3	**22.1**	**22.2**	13.7	9.5
STACKOV... LR	**0.68**	0.62	0.64	0.22	0.19
EMNIST AE	7.29	16.99	**0.98**	**1.21**	2.63

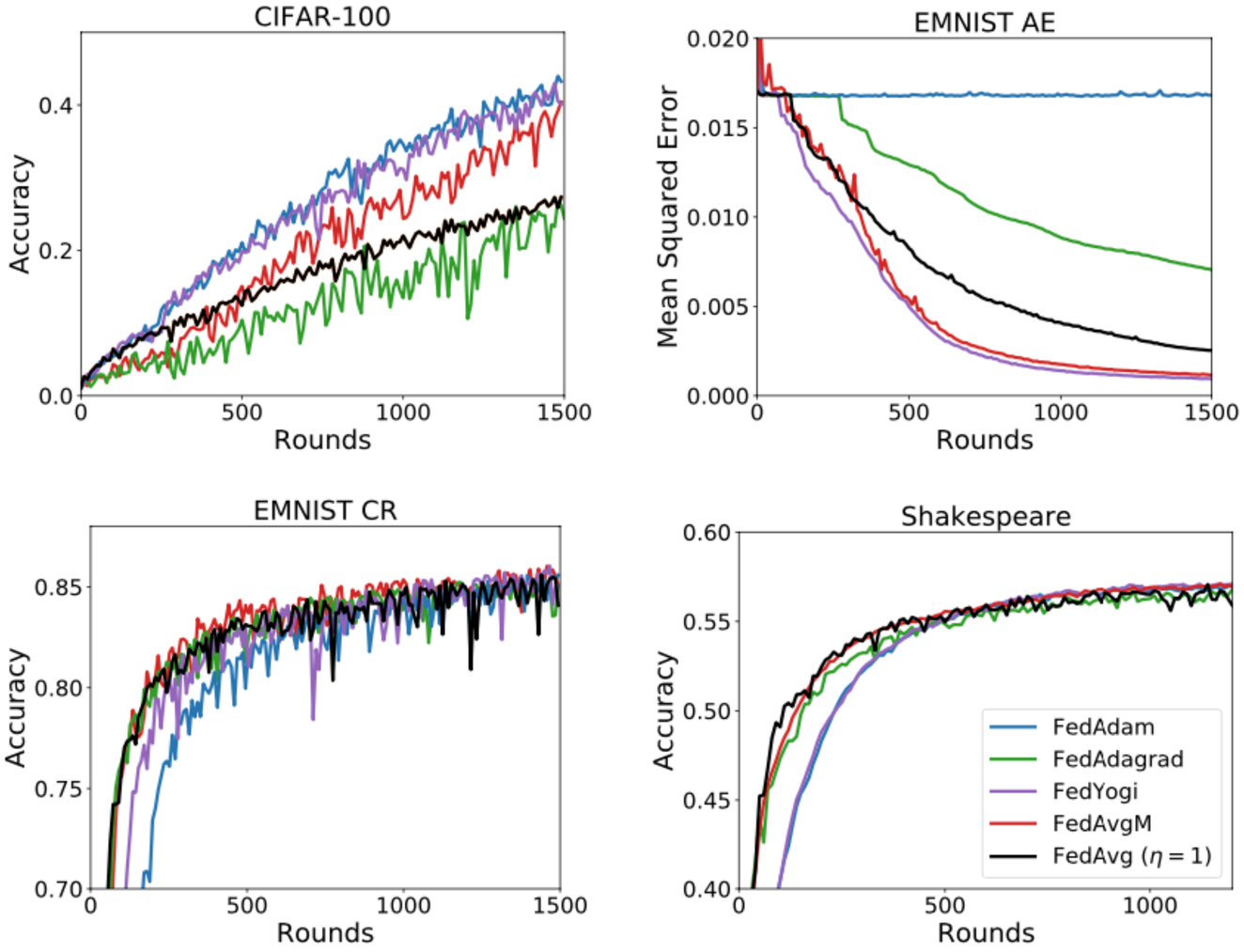
CONSTANT LEARNING RATES

- 테스트셋 성능

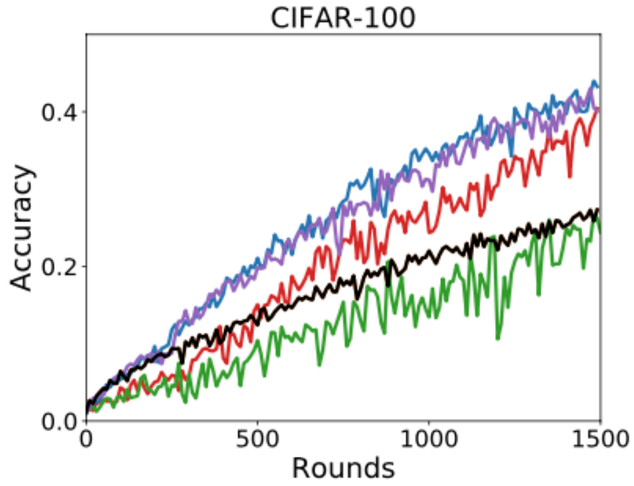


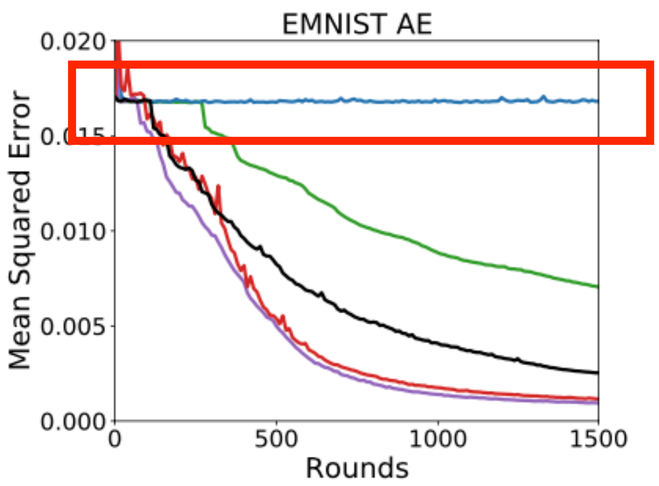


CONSTANT LEARNING RATES



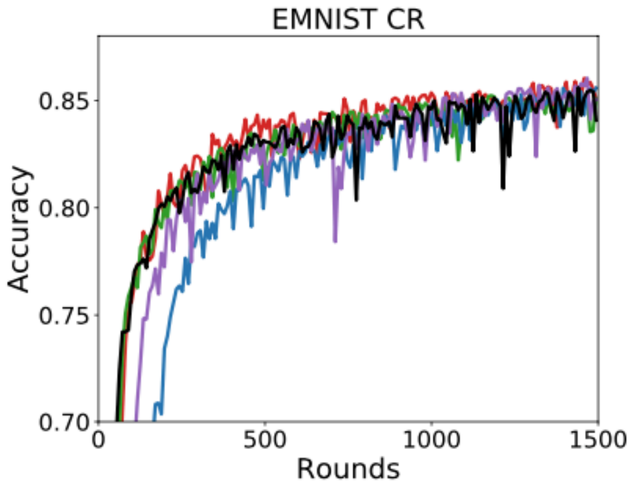
CONSTANT LEARNING RATES

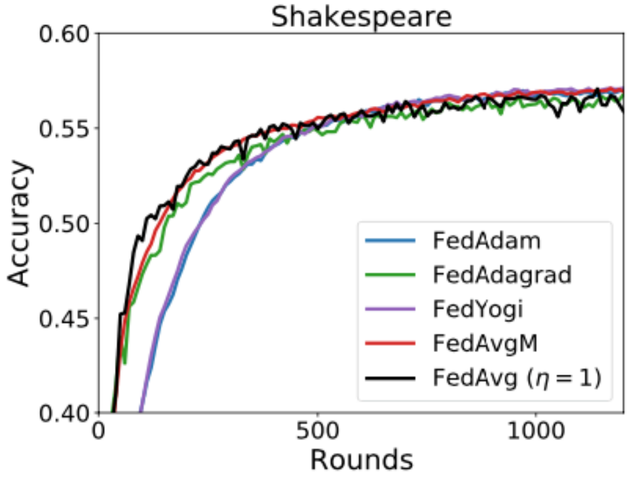




모든 학습률 조합에 대해
수렴에 실패

Saddle Point 때문으로 예상





CONSTANT LEARNING RATES

- 클라이언트 학습률과 서버 학습률의 비(ratio)
- 직관적으로
  - 클라이언트 학습률이 크면
  - 서버는 학습률을 줄여 드리프트를 방지

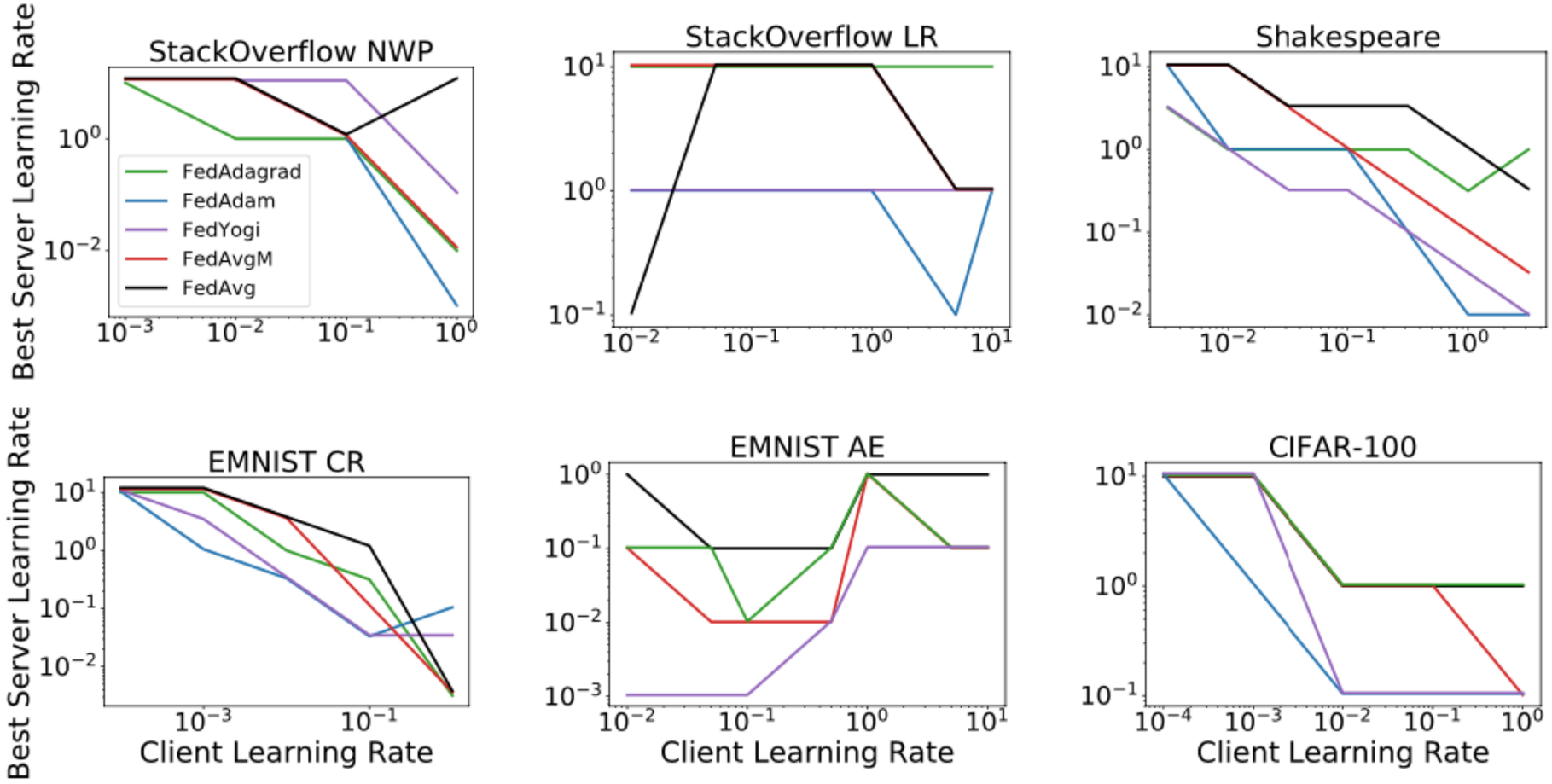
CONSTANT LEARNING RATES

- 직관을 검증하기 위해 최적의 서버 학습률을 찾음
- 대부분의 경우 역(inverse)의 관계에 있음

CONSTANT LEARNING RATES

- 일부의 예외
- StackOverflow LR
  - 각 최적화기에 대해 선호되는 서버 학습률이 존재
- EMNIST AE
  - 학습률 간의 관계가 보다 복잡해보임
  - Saddle point 때문으로 예상

CONSTANT LEARNING RATES



RESULT
RATE DECAY

IMPROVING PERFORMANCE WITH LEARNING RATE DECAY

- INVSQRT
  - Inverse Square Root Decay
  - $\frac{\eta_l}{\sqrt{t}}$
- EXPDECAY
  - Staircase (Discrete) exponential decay schedule
  - 500 라운드마다 η_l 을 0.1씩 감소

IMPROVING PERFORMANCE WITH LEARNING RATE DECAY

- 클라이언트 에폭을 10으로,
- 라운드마다 10 클라이언트를 샘플링

IMPROVING PERFORMANCE WITH LEARNING RATE DECAY

- EMNIST CR
- 중앙화된 학습과 비교
  - 100 에폭마다 중앙화된 최적화기를 정밀히 튜닝
  - 정확도 88%를 획득

IMPROVING PERFORMANCE WITH LEARNING RATE DECAY

- Centralized vs. INVSQRT vs. EXPDECAY 비교
  - 정확도
  - 마지막 100 라운드

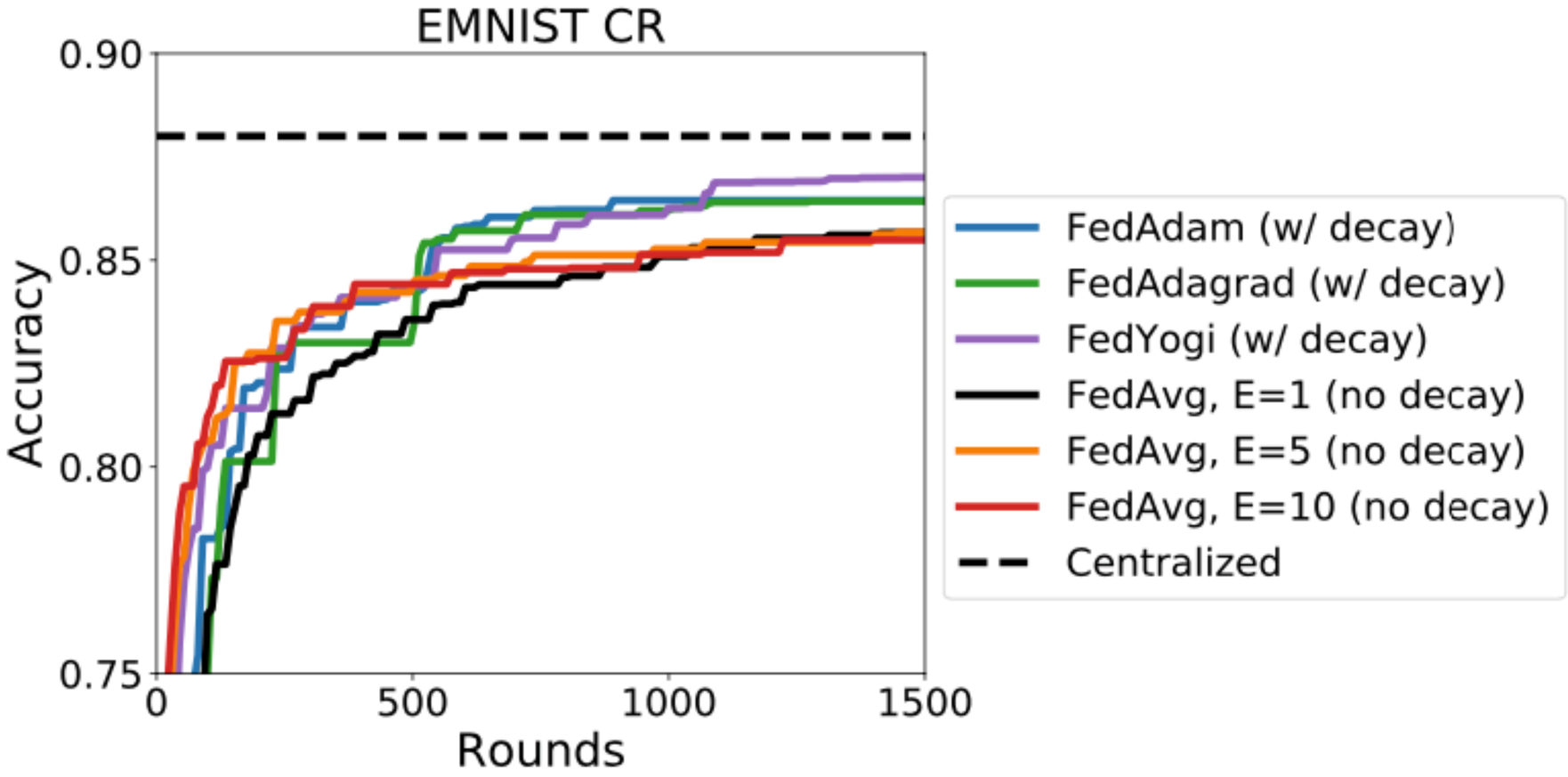
	ADAGRAD	ADAM	YOGI	SGDM	SGD
CENTRALIZED	88.0	87.9	88.0	87.7	87.7
FED...	**ADAGRAD**	**ADAM**	**YOGI**	**AVGM**	**AVG**
CONSTANT η_l	85.7	86.0	86.1	86.3	85.9
INVSQRT	84.8	86.8	86.6	86.1	85.5
EXPDECAY	87.1	87.1	**87.6**	**87.5**	87.1

IMPROVING PERFORMANCE WITH LEARNING RATE DECAY

- EXPDECAY가 모든 연합 최적화기 중 가장 좋은 성능
  - 중앙화된 학습과 거의 유사한 정확도
- EXPDECAY를 사용한 FEDYOGI와 FEDAVGM이 가장 좋은 성능
  - 상수의 클라이언트 학습률에서도 그랬음

IMPROVING PERFORMANCE WITH LEARNING RATE DECAY

- EXPDECAY를 사용한 적응형 최적화기들과 바닐라 FEDAVG의 비교



IMPROVING PERFORMANCE WITH LEARNING RATE DECAY

- EXPDECAY를 사용한 적응형 최적화기들과 바닐라 FEDAVG의 비교
- EXDECAY를 사용한 적응형 최적화기가 모든 버전의 FEDAVG를 압도
  - 비록 FEDAVG에서 에폭 수에 따라서도 성능 차이를 보이지만

CONCLUSION

CONCLUSION

- 적응형 최적화기가
  - 연합 학습의 수렴성을 증대시키는
  - 강력한 도구가 될 수 있음

CONCLUSION

- 본 논문에서의 일반화된 방법으로
  - 다른 (적응형) 최적화기를 구축할 수 있음

CONCLUSION

- 추후 연구에서는
  - 학습률이나 다른 하이퍼파라미터 세팅에 관련된 연구
  - 등이 가능할 것

ADAPTIVE FL

ADAPTIVE FEDERATED OPTIMIZATION