

Informs Annual Meeting, Philadelphia  
November 1-4, 2015

# Federated Optimization

## Distributed Optimization Beyond the Datacenter

Jakub Konečný, University of Edinburgh  
Brendan McMahan, Google Inc.

# Outline

---

- ▶ Optimization in Machine Learning
  - ▶ Empirical Risk Minimization
  
- ▶ Distributed Optimization
  - ▶ Motivation
  - ▶ Current challenges
  
- ▶ Federated Optimization
  - ▶ (possible) future of large-scale Machine Learning
  - ▶ Preliminary results
  - ▶ A lot of motivation for future research

# Machine Learning Setting

---

- ▶ Space of input-output pairs  $(x, y) \in \mathcal{X} \times \mathcal{Y}$
- ▶ Unknown distribution  $P(x, y)$
- ▶ A relationship between inputs and outputs  $P(y | x)$
- ▶ Loss function to measure discrepancy between predicted and real output  $\ell(\hat{y}, y)$
  
- ▶ Define Expected Risk

$$E(\phi) = \int \ell(\phi(x), y) dP(x, y) = \mathbb{E}[\ell(\phi(x), y)]$$

# Machine Learning Setting

---

- ▶ Ideal goal: Find  $\phi^*$  such that,

$$\phi^*(x) = \arg \min_{\hat{y}} \mathbb{E}[\ell(\hat{y}, y) \mid x]$$

- ▶ But you cannot even evaluate  $E(\phi)$

- ▶ Define Expected Risk

$$E(\phi) = \int \ell(\phi(x), y) dP(x, y) = \mathbb{E}[\ell(\phi(x), y)]$$

# Machine Learning Setting

---

▶ We at least have IID samples  $(x_i, y_i), i = 1, \dots, n$

▶ Define Empirical Risk

$$E_n(\phi) = \frac{1}{n} \sum_{i=1}^n \ell(\phi(x_i), y_i) = \mathbb{E}_n[\ell(\phi(x), y)]$$

# Machine Learning Setting

---

- ▶ First learning principle – fix a family  $\mathcal{F}$  of candidate prediction functions
- ▶ Find Empirical Minimizer  $\phi_n$

$$\phi_n = \arg \min_{\phi \in \mathcal{F}} E_n(\phi)$$

- ▶ Define Empirical Risk

$$E_n(\phi) = \frac{1}{n} \sum_{i=1}^n \ell(\phi(x_i), y_i) = \mathbb{E}_n[\ell(\phi(x), y)]$$

# Problem structure

---

- ▶ Typical objective

$$\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n f_i(x)$$

- ▶ Many recent fast algorithms for single computer
  - ▶ SGD (Stochastic Gradient Descent)
  - ▶ SDCA (Stochastic Dual Coordinate Ascent)
  - ▶ SAG (Stochastic Avergae Gradient)
  - ▶ SVRG/S2GD (Stochastic Variance Reduced Gradient)
  - ▶ SAGA (Improvement on SAG)
  - ▶ ...

# Traditional efficiency analysis

- ▶ Given algorithm  $\mathcal{A}$ , the time needed is

Total number of iterations needed

Time needed to run one iteration of algorithm  $\mathcal{A}$

$$\text{TIME} = \mathcal{I}_{\mathcal{A}}(\epsilon) \times \mathcal{T}_{\mathcal{A}}$$

Target accuracy

- ▶ Main trend – Stochastic methods
  - ▶ Big  $\mathcal{I}_{\mathcal{A}}(\epsilon)$ , small  $\mathcal{T}_{\mathcal{A}}$





# Distributed Optimization

# Problem structure

---

- ▶ Original objective

$$\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n f_i(x)$$

- ▶ Functions  $f_i(x)$  described by  $n$  data points
- ▶  $n$  can be millions
  - ▶ Articles in major newspaper
- ▶  $n$  can be billions
  - ▶ Users of Facebook
- ▶  $n$  can be trillions
  - ▶ Ad click-through rate predictions

# Problem structure

- ▶ Original objective

$$\min_{x \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^n f_i(x)$$

- ▶ Data distributed across  $K$  machines

- ▶ Partition of  $\{1, \dots, n\}$  to  $\{\mathcal{P}_k\}_{k=1}^K$

- ▶ Number of local data points  $n_k = |\mathcal{P}_k|$

- ▶ Objective can be reformulated as

$$\min_{x \in \mathbb{R}^d} f(x) = \sum_{k=1}^K \frac{n_k}{n} F_k(x) = \sum_{k=1}^K \frac{n_k}{n} \cdot \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(x)$$

Computer  $k$  has direct access only to function  $F_k$

# Algorithms for distributed optimization

---

- ▶ First trend
  - ▶ Naïve distributed variants of serial algorithms
  - ▶ Often inefficient; most time spent on communication

# Communication time

---

▶ Let's say  $v \in \mathbb{R}^{100}$

▶ Time to read  $v$  from memory (RAM)

▶ 100 *ns*

▶ Time to send  $v$  to another machine

▶ 500,000 *ns*


# Distributed efficiency analysis

---

- ▶ There is lot of potential for improvement, if  $c \gg \mathcal{T}_A$  because most of the time is spent on communication

$$\text{TIME} = \mathcal{I}_A(\epsilon) \times (c + \mathcal{T}_A)$$

Time for round of communication



# Algorithms for distributed optimization

---

- ▶ First trend
  - ▶ Naïve distributed variants of serial algorithms
  - ▶ Often inefficient; most time spent on communication
  
- ▶ Second trend
  - ▶ Communication efficient algorithms
  - ▶ Idea: perform a lot of computation locally followed by a round of communication
  - ▶ Very useful in practice
  - ▶ DANE (Shamir et al., 2014)
  - ▶ CoCoA (Jaggi et al., 2014, 2015)
  - ▶ DiSCO (Zhang, Xiao, 2015) (Session WE16)

# DANE

---

- A communication efficient algorithm for

$$\min_{x \in \mathbb{R}^d} f(x) = \sum_{k=1}^K \frac{n_k}{n} F_k(x)$$

---

## Algorithm 1 Distributed Approximate Newton (DANE)

---

- 1: **Input:** starting point  $\tilde{x}_0 \in \mathbb{R}^d$ , regularizer  $\mu$
- 2: **for**  $s = 0, 1, 2, \dots$  **do**
- 3:     Compute  $\nabla f(\tilde{x}_s) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(\tilde{x}_s)$  and distribute to all machines
- 4:     For each node  $k \in \{1, \dots, K\}$ , solve

$$x_k = \arg \min_{x \in \mathbb{R}^d} \left\{ F_k(x) - (\nabla F_k(\tilde{x}_s) - \nabla f(\tilde{x}_s))^T x + \frac{\mu}{2} \|x - \tilde{x}_s\|^2 \right\}$$

- 5:     Compute  $\tilde{x}_{s+1} = \frac{1}{K} \sum_{k=1}^K x_k$
- 6: **end for**



# DANE (necessary assumptions)

---

- ▶ Explicit

- ▶ Each node has access to IID sample of data points
- ▶ Each node has the same number of data points

- ▶ Implicit

- ▶ Number of computers,  $K$ , (relatively) small
- ▶ Size of data available locally,  $n_k$ , (relatively) big
- ▶ In particular,  $n_k \gg K$

- ▶ DANE relies on the assumptions heavily

- ▶ Both in theory, and in practice

# Distributed Optimization (today)

- ▶ Data are stored in datacenters
  - ▶ People worried about “Google knowing everything”
  - ▶ Relatively few computers, full of data

Major issue for big companies

Small  $K$

Big  $n_k$

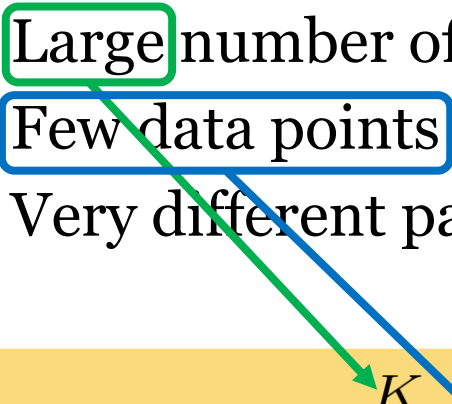
$$\min_{x \in \mathbb{R}^d} f(x) = \sum_{k=1}^K \frac{n_k}{n} F_k(x) = \sum_{k=1}^K \frac{n_k}{n} \cdot \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(x)$$

# Federated Optimization

# Federated Optimization (motivation)

---

- ▶ In order to even better protect privacy
  - ▶ Users keep their data on devices
  - ▶ Users provide computational power of devices
- ▶ Consequences
  - ▶ Large number of computers (devices)
  - ▶ Few data points on each device
  - ▶ Very different patterns in data on different devices


$$\min_{x \in \mathbb{R}^d} f(x) = \sum_{k=1}^K \frac{n_k}{n} F_k(x) = \sum_{k=1}^K \frac{n_k}{n} \cdot \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(x)$$

# DANE reminder

---

---

## Algorithm 1 Distributed Approximate Newton (DANE)

---

- 1: **Input:** starting point  $\tilde{x}_0 \in \mathbb{R}^d$ , regularizer  $\mu$
- 2: **for**  $s = 0, 1, 2, \dots$  **do**
- 3:     Compute  $\nabla f(\tilde{x}_s) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(\tilde{x}_s)$  and distribute to all machines
- 4:     For each node  $k \in \{1, \dots, K\}$ , solve

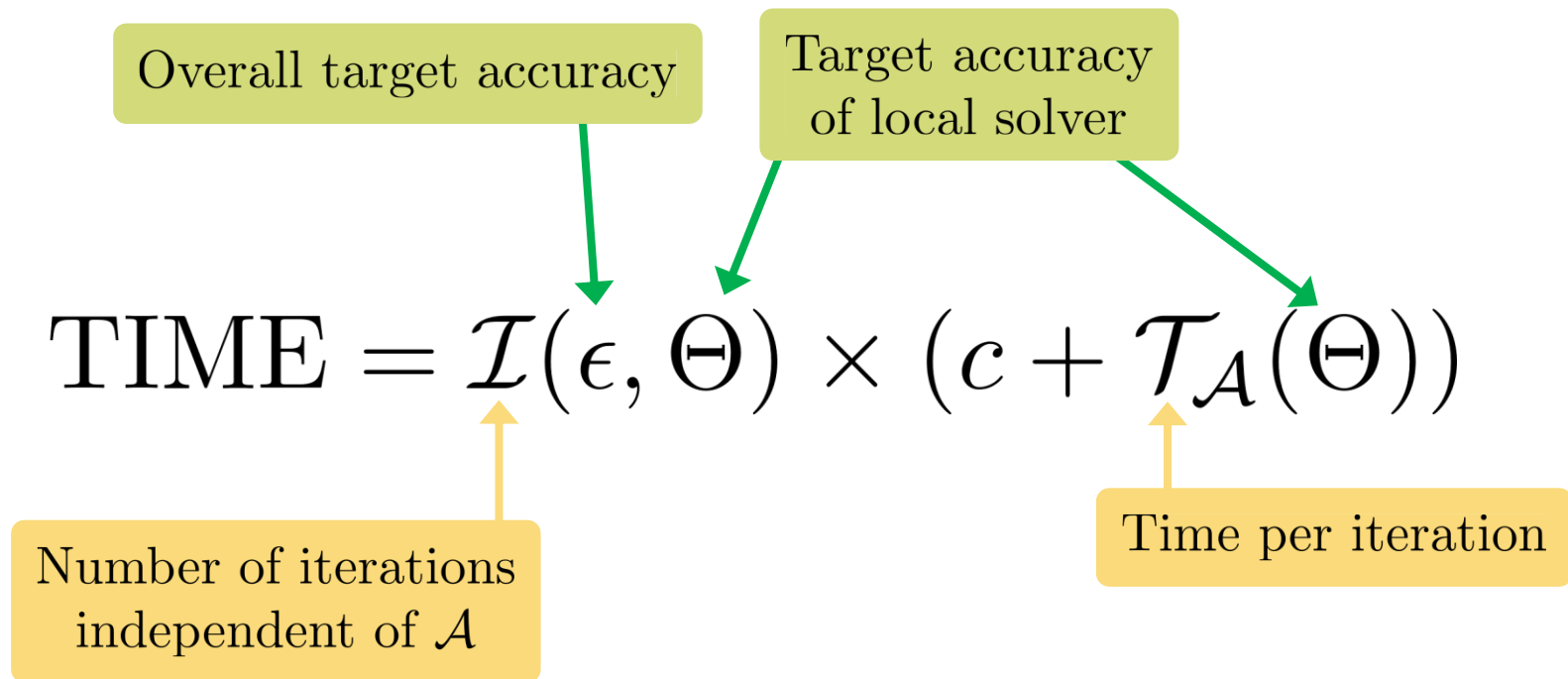
$$x_k = \arg \min_{x \in \mathbb{R}^d} \left\{ F_k(x) - (\nabla F_k(\tilde{x}_s) - \nabla f(\tilde{x}_s))^T x + \frac{\mu}{2} \|x - \tilde{x}_s\|^2 \right\}$$

- 5:     Compute  $\tilde{x}_{s+1} = \frac{1}{K} \sum_{k=1}^K x_k$
  - 6: **end for**
- 

- ▶ Not stable / robust method
- ▶ But solving the subproblem approximately...

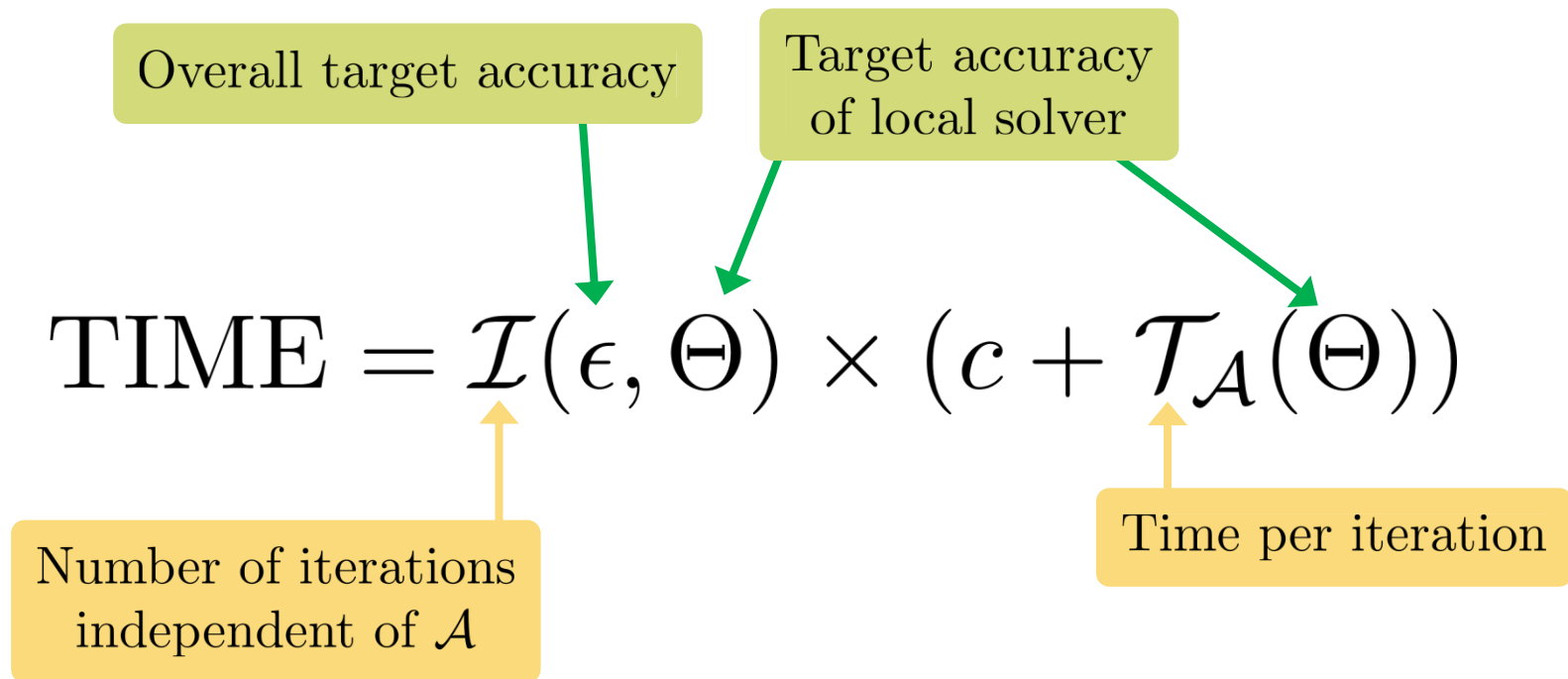
# Efficiency analysis revisited

- ▶ If any algorithm  $\mathcal{A}$  is used to solve the DANE subproblem to relative  $\Theta$  accuracy



# Efficiency analysis revisited – power

- ▶ If  $\mathcal{A}$  chosen as Gradient Descent, run for 1 iteration
  - ▶ Equivalent to naïve distributed Gradient Descent
  - ▶ Greater flexibility with general subproblem



# SVRG/S2GD algorithm

---

---

## Algorithm 1 SVRG

---

```
1: parameters:  $m = \max \#$  of stochastic steps per epoch,  $h = \text{stepsize}$ 
2:  $\tilde{x}_0 = \text{starting point}$ 
3: for  $j = 0, 1, 2, \dots$  do
4:   Precompute  $\nabla f(\tilde{x}_j) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(\tilde{x}_j)$ 
5:    $x \leftarrow \tilde{x}_j$ 
6:   for  $t = 0$  to  $m - 1$  do
7:     Pick  $i \in \{1, 2, \dots, n\}$ , uniformly at random
8:      $x \leftarrow x - h (\nabla f_i(x) - \nabla f_i(\tilde{x}_j) + \nabla f(\tilde{x}_j))$ 
9:   end for
10:   $\tilde{x}_{j+1} \leftarrow x$ 
11: end for
```

---



# Apply SVRG/S2GD as local solver in DANE

---

## ► Equivalent to

---

**Algorithm 1** A distributed version of SVRG/S2GD

---

```
1: parameters:  $m = \#$  of stochastic steps per epoch,  $h =$  stepsize,
   data partition  $\{\mathcal{P}_k\}_{k=1}^K$ , starting point  $\tilde{x}_0$ 
2: for  $s = 0, 1, 2, \dots$  do ▷ Overall iterations
3:   Precompute  $\nabla f(\tilde{x}_s) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(\tilde{x}_s)$ 
4:   for  $k = 1$  to  $K$  do in parallel over nodes  $k$  ▷ Distributed loop
5:     Initialize:  $x_k = \tilde{x}_s$ 
6:     for  $t = 1$  to  $m$  do ▷ Actual update loop
7:       Sample  $i \in \mathcal{P}_k$  uniformly at random
8:        $x_k \leftarrow x_k - h(\nabla f_i(x_k) - \nabla f_i(\tilde{x}_s) + \nabla f(\tilde{x}_s))$ 
9:     end for
10:  end for
11:   $\tilde{x}_{s+1} \leftarrow \tilde{x}_s + \frac{1}{K} \sum_{k=1}^K (x_k - \tilde{x}_s)$  ▷ Aggregate updates from nodes
12: end for
```

---

# Improving the algorithm (sparsity)

## ▶ Sprarsity

- ▶ Data point in  $\mathbb{R}^d$  contains only very small number of non-zero elements
- ▶ Extremely common in large-scale applications

## ▶ Unbiasing stochastic gradients

- ▶ Original update direction

$$\nabla f_i(x_k) - \nabla f_i(\tilde{x}_s) + \nabla f(\tilde{x}_s)$$

- ▶ New update direction

$$S_k [\nabla f_i(x_k) - \nabla f_i(\tilde{x}_s)] + \nabla f(\tilde{x}_s)$$

Diagonal; ratio of global and local appearance frequencies

sparse vector  
 $\in \mathbb{R}^{15}$

# Improving the algorithm (sparsity)

## ▶ Sprarsity

- ▶ Data point in  $\mathbb{R}^d$  contains only very small number of non-zero elements
- ▶ Extremely common in large-scale applications

## ▶ Adaptive aggregation of updates

### ▶ Original aggregation

$$\tilde{x}_{s+1} \leftarrow \tilde{x}_s + \frac{1}{K} \sum_{k=1}^K (x_k - \tilde{x}_s)$$

### ▶ Updated aggregation

$$\tilde{x}_{s+1} \leftarrow \tilde{x}_s + A \sum_{k=1}^K \frac{n_k}{n} (x_k - \tilde{x}_s)$$

Diagonal; Inverse coordinate appearance on computers

Weighted average

sparse vector  
 $\in \mathbb{R}^{15}$

$(1, 0, 0, 0, 3, 0, -2, 0, 0, 0, 0, 0, -1, 0, 0)$

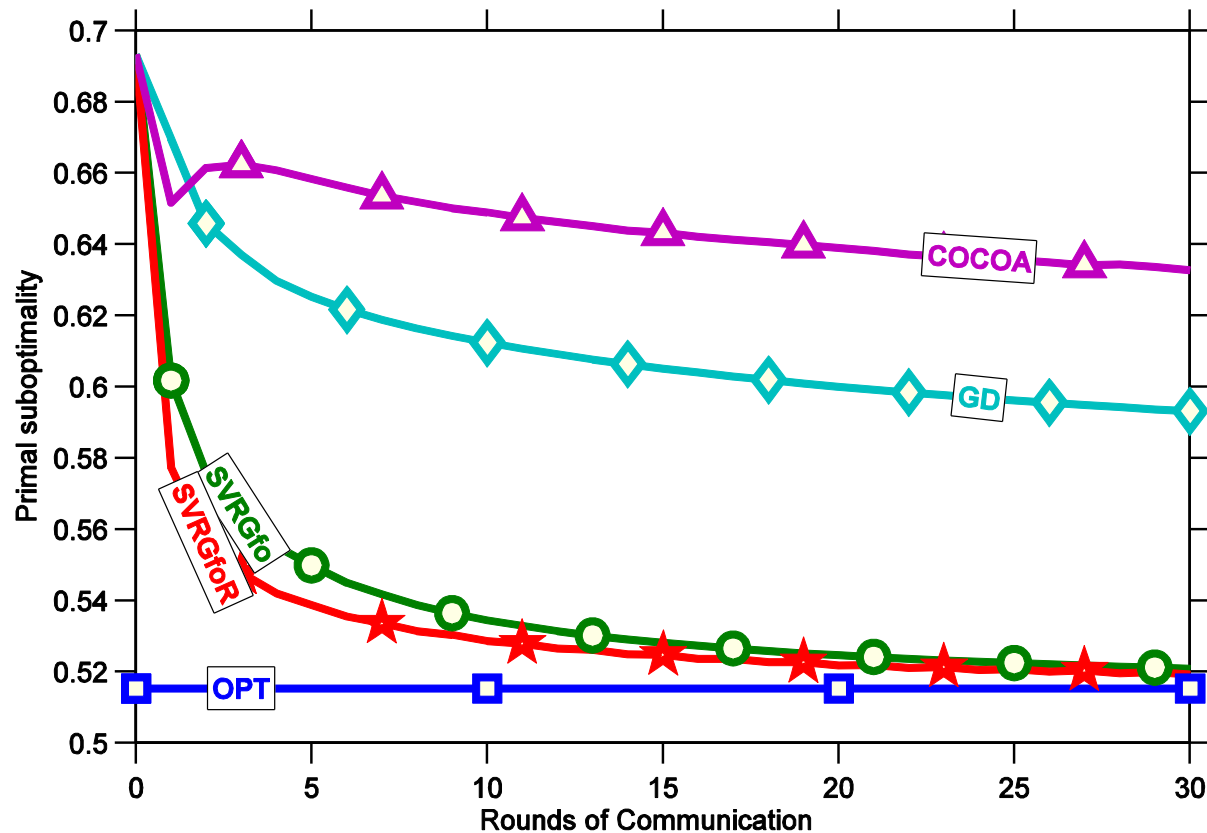
# Experiments

---

- ▶ Setting
  - ▶ Data from public Google+ posts
  - ▶ Randomly chosen 10,000 users with 100 or more posts
  - ▶ Train/test split 75%/25%
  - ▶ Total 2,166,693 training posts
  - ▶ Extremes: 75 and 9,000 posts per user
  - ▶ See (simulate) each user as a separate computer
  - ▶ Simple bag-of-words language model
  - ▶ Dictionary of 20,000 most common English words
- ▶ Prediction task
  - ▶ Post has comment or not (binary classification)

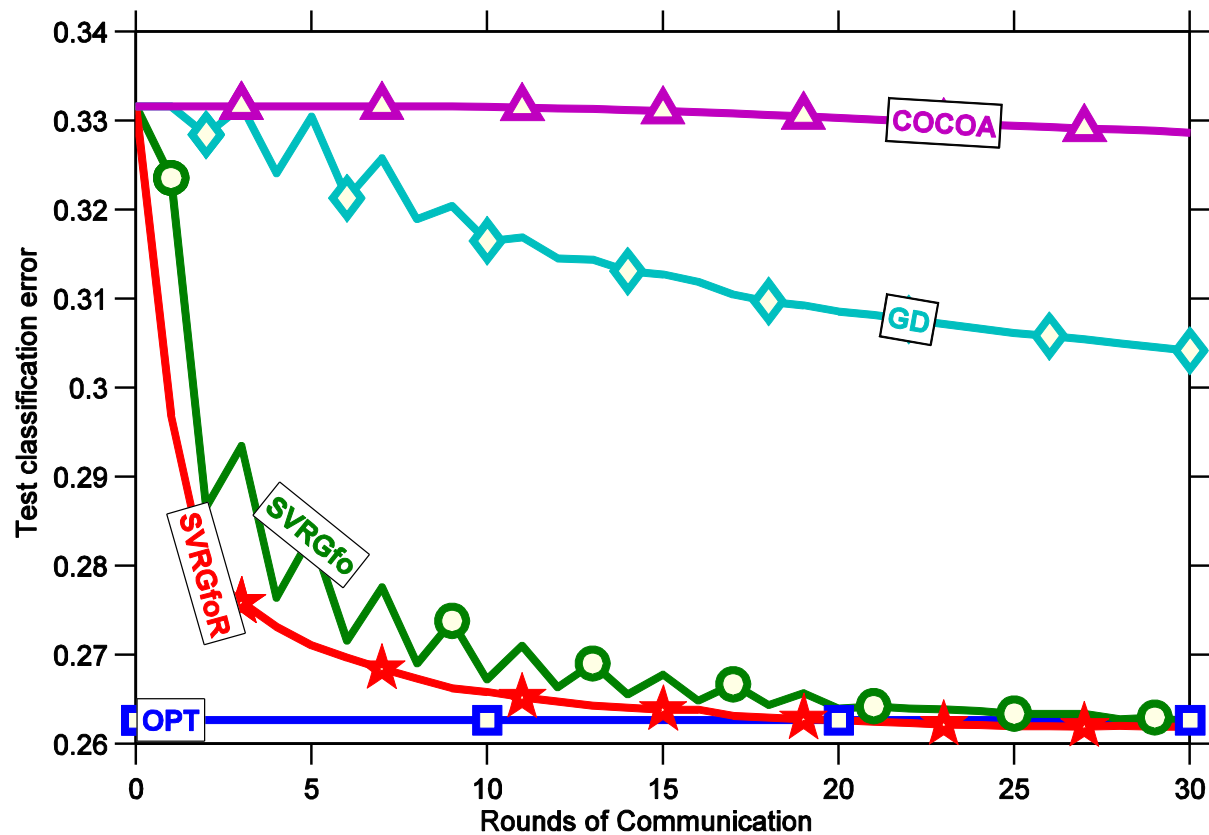
# Experiments

## ► Optimization objective



# Experiments

## ► Test error



# Summary

---

- ▶ Federated Optimization
  - ▶ Very large number of computers  $K$
  - ▶ Small number of data on each computer
  - ▶ Unbalanced – very different data sizes locally
  - ▶ Local data may be clustered; very far from IID
  
- ▶ Potentially industrial reality within few years
- ▶ Introduces many further questions

# Future challenges

---

- ▶ More rigorous experiments
  - ▶ Larger problems
  - ▶ Deep learning
- ▶ Analysis missing!
- ▶ Need for good public dataset
  - ▶ With naturally per-user clustered data
- ▶ Asynchronous algorithms
- ▶ Differential privacy
  - ▶ Google cannot learn *anything*
- ▶ Many parts of ML pipelines need to be redesigned
  - ▶ Work for 1000+ people