論文輪講
Federated Optimization: Distributed Machine Learning for On-Device Intelligence

arXiv:1610.02527, Jakub Konecný, et al., 2016

杉浦 圭祐

慶應義塾大学理工学部情報工学科 松谷研究室

May 16, 2019

目次

目次

問題の定式化

- 問題の定式化
  - 解くべき問題は次のように定式化される

$$\min_{\boldsymbol{w}\in\mathbb{R}^D} f(\boldsymbol{w}), \qquad f(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\boldsymbol{w}) \tag{1}$$

  - 入出力のデータを $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$、損失関数を $f_i(\boldsymbol{w})$ とする
  - 具体的には、以下のような問題が考えられる

線形回帰：

$$f_i(\boldsymbol{w}) = \frac{1}{2}\left(\boldsymbol{x}_i^T\boldsymbol{w} - y_i\right)^2, \ y_i \in \mathbb{R} \tag{2}$$

ロジスティック回帰：

$$f_i(\boldsymbol{w}) = -\log\left(1 + \exp\left(-y_i\boldsymbol{x}_i^T\boldsymbol{w}\right)\right), \ y_i \in \{-1, 1\} \tag{3}$$

サポートベクタマシン：

$$f_i(\boldsymbol{w}) = \max\left\{0, 1 - y_i\boldsymbol{x}_i^T\boldsymbol{w}\right\}, \ y_i \in \{-1, 1\} \tag{4}$$

問題の定式化

- 問題の定式化
  - 上記は凸最適化問題である
  - 線形回帰、ロジスティック回帰、サポートベクタマシンより複雑なモデルにも適用可能
    $\Rightarrow$ 条件付き確率場や、ニューラルネットワーク
    $\Rightarrow$ 損失関数 $f_i(\boldsymbol{w})$ が非凸で、複雑な形状をしている場合
  - データ数 N が大き過ぎて、単一のノードで学習を進めるのは不可能
    $\Rightarrow$ 分散処理が必要 (データが複数の箇所に分散し、複数の相互接続されたノードで学習を行う)

    $\Rightarrow$ ノード間での通信時間がボトルネックとなる可能性
    $\Rightarrow$ 並列計算の利点を活かすために、モデルの学習を、単一ノードでの計算に適した簡単な問題に分割する必要がある

問題設定

- 最新 (State-of-the-art) の最適化手法
  - シーケンシャルであるため、並列処理には向かない
  - 各イテレーションでの処理は非常に高速だが、イテレーションの実行を何度も繰り返す必要がある
    ⇒ 各イテレーションの実行後に、複数のノード間で通信を行うと、性能が極端に落ちる
    ⇒ 各イテレーションの実行時間より、ノード間の通信時間の方が遥かに大きい

問題設定

- 従来手法における仮定
  1. データは各ノードに均等に分散している
  2. ノード数 K に比べて、1 つのノードが保持しているデータ数の平均 N/K の方が非常に大きい $(K \ll N/K)$
     $\Rightarrow$ 大規模なデータセンタに、データが格納されている想定
  3. 各ノードが、分布をよく表現するデータを保持している
     $\Rightarrow$ 各ノードが、独立同分布 (IID) 標本を持っているという前提

     $\Rightarrow$ 実際には、ノードの地理的な位置によって、各ノードが持つデータが、クラスタに分かれている可能性
     $\Rightarrow$ 各ノードが持つデータが時間的に変動し、ある時点では他のノードと似たようなデータを保持している可能性
     $\Rightarrow$ あるノードに頻繁に現れる特徴が、他のノードでは全く現れない可能性

問題設定

- 従来手法における仮定
  - 従来手法では、上記 3 つの仮定が成立する
    ⇒ Federated Learning では、これらの仮定を全く置かない
  - 従来手法では、最初に、デバイス上のデータを中央のノード (データセンタ等) に集める
    ⇒ 集めたデータをランダムにシャッフルし、複数の計算ノードに均等な数だけ配分して、学習させる
  - Federated Learning では、デバイス上のデータを中央のノードに送信しない
    ⇒ 各デバイスと中央のノードとの通信量が大幅に削減
    ⇒ ユーザのプライバシーを保護し、セキュリティを向上させる
    ⇒ データがデバイス上にしか存在しないので、中央のノードが攻撃されても、ユーザのデータが漏洩する危険性がない (攻撃されそうな箇所の候補が減る)

問題設定

- Federated Learning での問題設定
  - Federated Learning では、次のような現実的な仮定を置く

1. Massively Distributed: データは、多数のノードに分散
   $\Rightarrow$ ノード数 K と、1 つのノードが保持しているデータ数の平均 N/K を比較したとき、$N/K \ll K$ となる可能性

2. Non-IID: 各ノードが保持するデータは、異なる分布からサンプルされた可能性
   $\Rightarrow$ あるノードが保持しているデータは、データ全体の分布を表現しているとは限らない
   $\Rightarrow$ 各ノードが保持するデータは、独立に同一の分布からサンプルされた (IID) とは、仮定し難い

3. Unbalanced: ノードによって、保持しているデータ数は大きく異なる

問題設定

- Federated Learning での問題設定
  - 上記に加えて、この論文では次のような仮定を置く

1. データはスパースである
   ⇒ ある特徴は、ほんの僅かなノードやデータにしか現れない

2. データはモバイル端末上に存在し、プライバシー上の配慮が必要 (Privacy Sensitive)
   ⇒ 入出力データ $\{\boldsymbol{x}_i, y_i\}$ はデバイス上で作られる
   ⇒ 例えば、ユーザが次に入力する単語の予測、ユーザがシェアしそうな写真の予測、ユーザにとってどの通知が重要かの予測

問題設定

3 多数のデバイスが動作するので、事実上無限の計算能力が得られる
⇒ 但し、各デバイスと中央のノード間の、通信コストによって制限される (バンド幅の制限)
⇒ デバイスと中央のノード間の通信を、いかに減らせるかが、性能向上のための鍵となる

4 差分データ $\boldsymbol{\delta} \in \mathbb{R}^D$ が、モデルの学習に使用する唯一の情報である
⇒ 各デバイスは、1 回の Round につき、モデルのパラメータの差分 $\boldsymbol{\delta} \in \mathbb{R}^D$ を計算 (D は、モデルのパラメータの次元)
⇒ 差分 $\boldsymbol{\delta}$ が、デバイスから中央のノードにアップロードされる

問題設定

- Federated Learning で送信される差分データ $\boldsymbol{\delta}$ について
  - $\boldsymbol{\delta}$ は、ユーザのプライベートな情報を依然として含むかもしれないが、訓練データ $\{\boldsymbol{x}_i, y_i\}$ に比べれば無視できるほど小さい
  - モデルのパラメータの差分ベクトル $\boldsymbol{\delta}$ のサイズは、訓練データのサイズには関係ない
    $\Rightarrow$ 訓練データが巨大 (例えばユーザが撮影した動画) であっても、データそのものではなく、差分データのみを中央のノードに送信すればよいため、通信量が大幅に削減できる

問題設定

- 差分ベクトル $\boldsymbol{\delta}$ の使途は、元の訓練データ $\{\boldsymbol{x}_i, y_i\}$ に比べて限られる
  ⇒ モデルの訓練以外に、殆ど使い道がない
  ⇒ 各デバイスから送信された差分データは、モデルの訓練に使用した後は、破棄してよい

  ⇒ ユーザにとっては、アップロードしたデータが、モデルの訓練にしか使われない (想定外の方法で使われない) ことが分かっているので、安心できる (プライバシーの保護につながる)

  ⇒ モデルを訓練する側にとっては、ユーザのデータを保存する際の負担が軽減される
  ⇒ 訓練データであれば、漏洩しないように厳重に管理する必要がある
  ⇔ 差分データであれば、万が一不正にアクセスされても、ユーザの個人情報が漏洩することはない

問題設定

- 論文で提案された訓練アルゴリズムについて
  - Federated Learning のための訓練アルゴリズムを新たに設計した
    $\Leftarrow$ Massively Distributed、Non-IID、Unbalanced
  - 新たな訓練アルゴリズムによって、比較的少ない Round 数 (通信量) で、モデルのパラメータを収束させることができた

目次

基本的な最適化アルゴリズム

- 解くべき問題は、次のように定式化された
  - D はモデルのパラメータの次元数、N は訓練データ数
  - $\boldsymbol{w} \in \mathbb{R}^D$ はパラメータベクトル、$f_i(\boldsymbol{w})$ は損失関数

$$\min_{\boldsymbol{w} \in \mathbb{R}^D} f(\boldsymbol{w}), \qquad f(\boldsymbol{w}) = \frac{1}{N} \sum_{i=1}^{N} f_i(\boldsymbol{w}) \tag{5}$$

- ベースラインアルゴリズム
  - 基本的なアルゴリズムとして、以下を紹介する

  - 勾配降下法 (GD; Gradient Descent)
  - 確率的勾配降下法 (SGD; Stochastic Gradient Descent)

基本的な最適化アルゴリズム

- 勾配降下法 (GD; Gradient Descent)
  - パラメータの更新式は次のようになる

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - h_t \nabla f(\boldsymbol{w}^t) \tag{6}$$

  - $\nabla f(\boldsymbol{w}^t)$ は次のように定義される

$$\begin{aligned} \nabla f(\boldsymbol{w}^t) &\equiv \left. \frac{\partial}{\partial \boldsymbol{w}} f(\boldsymbol{w}) \right|_{\boldsymbol{w}=\boldsymbol{w}^t} & (7) \\ &= \left. \frac{1}{N} \sum_{i=1}^{N} \frac{\partial}{\partial \boldsymbol{w}} f_i(\boldsymbol{w}) \right|_{\boldsymbol{w}=\boldsymbol{w}^t} & (8) \\ &= \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(\boldsymbol{w}^t) & (9) \end{aligned}$$

    損失関数 $f(\boldsymbol{w})$ のパラメータ $\boldsymbol{w}$ による勾配の、$\boldsymbol{w} = \boldsymbol{w}^t$ における値
  - $h_t > 0$ は学習率 (ステップサイズ)

基本的な最適化アルゴリズム

- 勾配降下法 (GD; Gradient Descent) の問題点
  - 勾配 $\nabla f(\boldsymbol{w}^t)$ を求めるためには、N 個の各データについての勾配 $\nabla f_i(\boldsymbol{w}^t)$ を計算する必要がある
    $\Rightarrow$ 1 回のパラメータ更新のために、全データを処理する必要がある
    $\Rightarrow$ データ数 N は非常に大きいため、勾配の計算に時間が掛かり過ぎる

    $\Rightarrow$ 勾配降下法は、遅すぎて使い物にならない

  - モメンタム項を加えることで、アルゴリズムを高速化できる
    $\Rightarrow$ 但し、1 回のパラメータ更新のために、全データを処理する必要はある
    $\Rightarrow$ モメンタム項を加えても、やはり使い物にならない

基本的な最適化アルゴリズム

- 確率的勾配降下法 (SGD; Stochastic Gradient Descent)
  - パラメータの更新式は次のようになる

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - h_t \nabla f_{i_t}(\boldsymbol{w}^t) \tag{10}$$

  - $\nabla f_{i_t}(\boldsymbol{w}^t)$ は次のように定義される

$$\nabla f_{i_t}(\boldsymbol{w}^t) = \left. \frac{\partial}{\partial \boldsymbol{w}} f_{i_t}(\boldsymbol{w}) \right|_{\boldsymbol{w}=\boldsymbol{w}^t} \tag{11}$$

  - i_t は、1 から N の中から適当に選択されたインデックス
    $\Rightarrow$ 時刻 t では、i_t 番目のデータ $\{\boldsymbol{x}_{i_t}, y_{i_t}\}$ とパラメータ $\boldsymbol{w}^t$ を用いて、損失関数 $f_{i_t}(\boldsymbol{w}^t)$ を計算
    $\Rightarrow$ 損失関数の勾配 $\nabla f_{i_t}(\boldsymbol{w}^t)$ を求めて、パラメータ $\boldsymbol{w}$ を勾配の方向に更新

基本的な最適化アルゴリズム

- 確率的勾配降下法 (SGD; Stochastic Gradient Descent)
  - 1 回のパラメータ更新のためには、1 つのデータ点に対する勾配 $\nabla f_{i_t}(\boldsymbol{w}^t)$ だけを求めればよい
    $\Rightarrow$ 勾配降下法とは異なり、全データを処理する必要がない
  - 1 つのデータ点に対する勾配の期待値は、損失関数 $f(\boldsymbol{w})$ の勾配の不偏推定量となっている
    $\Rightarrow \mathbb{E}\left[\nabla f_{i_t}(\boldsymbol{w})\right] = \nabla f(\boldsymbol{w}^t)$ であり、この手法が正当化される

    $\Rightarrow$ 実際には、データ点のサンプリングによって、(真の勾配に対して) ノイズが加わった勾配が得られるため、パラメータの収束が遅くなる
    $\Rightarrow$ 学習率 (ステップサイズ)h_t の設定が重要になる
  - 勾配の計算に用いるデータ点 i_t を、毎回ランダムに選ぶのではなく、全データをランダムな順で取り出し、その勾配を求めてパラメータを更新していく手法がある

基本的な最適化アルゴリズム

- GD と SGD との比較
  - GD ではパラメータの収束が速い $\Leftrightarrow$ SGD は収束が遅い
  - GD では各イテレーションの計算に非常に時間が掛かる
    $\Leftarrow$ 各イテレーションにおいて、全データを処理する必要がある
    $\Leftrightarrow$ SGD では各イテレーションの計算は高速であり、計算時間はデータ数 N に依存しない
  - 今回解こうとしている問題では、パラメータの精度はそこまで高くなくてもよい
    $\Rightarrow$ SGD で十分である (極端な場合には、全データを 1 回ずつ処理するだけで、パラメータが収束)
    $\Leftarrow$ GD の場合は、全データを 1 回ずつ処理して、ようやくパラメータを 1 回更新できる

目次

ランダム化された最適化アルゴリズム

- ランダム化された座標降下法 (RCD; Randomized Coordinate Descent)
  - パラメータの更新式は次のようになる

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - h_{j_t} \nabla_{j_t} f(\boldsymbol{w}^t) \boldsymbol{e}_{j_t} \tag{12}$$

  - j_t は、1 から D(パラメータの次元数) の中から適当に選択された次元
  - h_{j_t} は学習率 (ステップサイズ)、$\boldsymbol{e}_{j_t} \in \mathbb{R}^D$ は次元 j_t 方向の標準基底ベクトル
  - 勾配 $\nabla_j f(\boldsymbol{w}^t)$ は次のように定義される

$$\nabla_j f(\boldsymbol{w}^t) = \frac{\partial}{\partial w_j} f(\boldsymbol{w}^t) \tag{13}$$

$$= \frac{1}{N} \sum_{i=1}^{N} \frac{\partial}{\partial w_j} f_i(\boldsymbol{w}) \Big|_{\boldsymbol{w}=\boldsymbol{w}^t} \tag{14}$$

$$= \frac{1}{N} \sum_{i=1}^{N} \nabla_{w_j} f_i(\boldsymbol{w}^t) \tag{15}$$

ランダム化された最適化アルゴリズム

- ランダム化された座標降下法 (RCD; Randomized Coordinate Descent)
  - 最適化問題 $\min_{\boldsymbol{w} \in \mathbb{R}^D} f(\boldsymbol{w})$ を、幾つかの部分問題に分割
  - $f(\boldsymbol{w})$ を、$j \neq j_t$ であるような変数については固定したまま、ある 1 つの変数 $j_t \in \{1, \ldots, D\}$ について最小化する
    $\Rightarrow$ 一度に 1 つの座標を最適化するため、座標降下法とよばれる

    $\Rightarrow$ 一般に、変数の部分集合について同時に最小化を行うアルゴリズムを、ブロック座標降下法という
  - 座標降下法は、ある 1 つの変数が、他の変数の最適値に影響を与えない場合に効果を発揮
    $\Rightarrow$ 「深層学習」の 8.7.2 節を参照

ランダム化された最適化アルゴリズム

- SDCA; Stochastic Dual Coordinate Ascent
  - 正則化項 $\psi(\boldsymbol{w})$ を付加した、以下の最適化問題を考える

$$\min_{\boldsymbol{w}\in\mathbb{R}^D} f(\boldsymbol{X}\boldsymbol{w}) + \psi(\boldsymbol{w}), \qquad f(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} f_i(y_i, \boldsymbol{x}_i^T\boldsymbol{w}) \tag{16}$$

  - 上記の双対問題 (Dual problem) は、Fenchel 双対定理から、次のようになる (K はデータ $\boldsymbol{x}$ の次元数) [3]

$$-\min_{\boldsymbol{u}\in\mathbb{R}^N} f^*(\boldsymbol{u}) + \psi^*\left(-\frac{\boldsymbol{X}^T\boldsymbol{u}}{N}\right) \tag{17}$$

  - 但し、f^* と ψ^* は、それぞれ f と ψ のルジャンドル変換である

ランダム化された最適化アルゴリズム

- SDCA; Stochastic Dual Coordinate Ascent
  - $f(\boldsymbol{x})$ のルジャンドル変換 $f^*(\boldsymbol{y})$ とは、関数 $f(\boldsymbol{x})$ の変数を、その微分 $\boldsymbol{y} = \boldsymbol{x}'$ に置き換えた関数であり、次のように定義される (関数 $f(\boldsymbol{x})$ を、その傾きの情報から捉えた関数)

$$f^*(\boldsymbol{y}) = \sup_{\boldsymbol{x}} \left\{ \boldsymbol{x}^T \boldsymbol{y} - f(\boldsymbol{x}) \right\} \tag{18}$$

  - 双対問題 (Dual problem) とは、最適化問題における主問題 (Primary problem) の補問題であり、主問題と双対問題は表裏一体の関係にある

ランダム化された最適化アルゴリズム

- SDCA; Stochastic Dual Coordinate Ascent
  - 例えば、以下の問題 P(主問題) を、次のように定める

$$\min_{\boldsymbol{x}} \boldsymbol{c}^T \boldsymbol{x} \quad \text{s.t.} \quad \boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}, \boldsymbol{x} \geq 0 \tag{19}$$

  - 一方の問題 D(双対問題) は、次のようになる

$$\max_{\boldsymbol{y}} \boldsymbol{b}^T \boldsymbol{y} \quad \text{s.t.} \quad \boldsymbol{A}^T \boldsymbol{y} \leq \boldsymbol{c} \tag{20}$$

  - 双対定理より、問題 P に最適解 $\boldsymbol{x}^*$ が存在すれば、問題 D にも最適解 $\boldsymbol{y}^*$ が存在して、$\boldsymbol{c}^T \boldsymbol{x}^* = \boldsymbol{b}^T \boldsymbol{y}^*$ が成立

ランダム化された最適化アルゴリズム

- SDCA; Stochastic Dual Coordinate Ascent
  - 今回の場合、問題 P(主問題) は、(正則化項付きの) 損失関数の最小化である
  - 問題 P の代わりに双対問題 D を解くことができる
  - このとき、問題 P の解が得られるので、最適なパラメータが求まる
  - SDCA は、このような考え方に基づくアルゴリズムである (詳細は省略)
  - 主問題と双対問題を、交互に解くアルゴリズムも考えられる
  - 以降のスライドでは、勾配 $\nabla f(\boldsymbol{w})$ の推定値に含まれるノイズを軽減するアルゴリズムをみていく

分散を抑えた確率的勾配降下法

- SAG; Stochastic Average Gradient
  - 確率的勾配の平均を取るアルゴリズム
  - SAG は、GD(勾配降下法) と SGD(確率的勾配降下法) の中間に位置

  - SAG の各イテレーションでは、$i_t \in \{1, \ldots, N\}$ を選択したうえで、次の処理が実行される

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t - \frac{\alpha_t}{N} \sum_{i=1}^{N} \boldsymbol{y}_i^t \tag{21}$$

$$= \boldsymbol{w}^t - \frac{\alpha_t}{N} \left[\sum_{i=1}^{N} \boldsymbol{y}_i^{t-1} + \left(\boldsymbol{y}_{i_t}^t - \boldsymbol{y}_{i_t}^{t-1} \right) \right] \tag{22}$$

$$= \boldsymbol{w}^t - \frac{\alpha_t}{N} \left[\sum_{i=1}^{N} \boldsymbol{y}_i^{t-1} + \left(\nabla f_{i_t}(\boldsymbol{w}^t) - \boldsymbol{y}_{i_t}^{t-1} \right) \right] \tag{23}$$

分散を抑えた確率的勾配降下法

- 但し、$\boldsymbol{y}_i^t$ は次のように定義される

$$\boldsymbol{y}_i^t = \begin{cases} \nabla f_i(\boldsymbol{w}^t) & (i = i_t) \\ \boldsymbol{y}_i^{t-1} & (\text{それ以外のとき}) \end{cases} \tag{24}$$

- 勾配 $\nabla f_i(\boldsymbol{w}^t)$ は次のように定義される

$$\nabla f_i(\boldsymbol{w}^t) \equiv \left. \frac{\partial}{\partial \boldsymbol{w}} f_i(\boldsymbol{w}) \right|_{\boldsymbol{w}=\boldsymbol{w}^t} \tag{25}$$

分散を抑えた確率的勾配降下法

- SAG; Stochastic Average Gradient
  - GD では、現在のパラメータ $\boldsymbol{w}^t$ を基に、全てのデータについて勾配 $\nabla f_i(\boldsymbol{w}^t)$ を計算
    $\Rightarrow$ その平均 $\sum_{i=1}^{N} \nabla f_i(\boldsymbol{w}^t)$(Full gradient) を使って、パラメータ $\boldsymbol{w}$ を更新
    $\Rightarrow$ 1 回のパラメータの更新には Full gradient が必要であり、全てのデータを使って計算するため、非常に処理が重い

  - SGD では、データ点 $i \in \{1, \ldots, N\}$ についてのみ (1 つのデータについてのみ)、勾配 $\nabla f_i(\boldsymbol{w}^t)$ を計算
    $\Rightarrow$ その勾配 $\nabla f_i(\boldsymbol{w}^t)$ を使って、パラメータ $\boldsymbol{w}$ を更新
    $\Rightarrow$ 1 点における勾配 $\nabla f_i(\boldsymbol{w}^t)$ を用いて、$\sum_{i=1}^{N} \nabla f_i(\boldsymbol{w}^t)$ を近似
    $\Rightarrow$ 1 回のパラメータの更新に必要な、計算量を少なく抑えられる

分散を抑えた確率的勾配降下法

- SAG; Stochastic Average Gradient
  - SAG では、全データに対する勾配 (Full gradient) を、少しずつ更新していく
    $\Rightarrow$ ある 1 つのデータ点 $i_t \in \{1, \ldots, N\}$ について、現在のパラメータ $\boldsymbol{w}^t$ の下で勾配 $\nabla f_{i_t}(\boldsymbol{w}^t)$ を計算
    $\Rightarrow$ 新しく求めた勾配 $\nabla f_{i_t}(\boldsymbol{w}^t)$ を使って、以下の式で Full gradient を更新

$$\sum_{i=1}^{N} \boldsymbol{y}_i^t = \sum_{j \neq i_t} \boldsymbol{y}_j^{t-1} + \nabla f_{i_t}(\boldsymbol{w}^t) \tag{26}$$

    $\Rightarrow$ 更新された Full gradient を使って、パラメータ $\boldsymbol{w}$ を更新

  - SAG では、計算される勾配にはバイアスが含まれる
    $\Rightarrow$ 後述の SAGA で計算される勾配の期待値は、勾配 $f(\boldsymbol{w})$ の不偏推定量に一致

分散を抑えた確率的勾配降下法

- SAG; Stochastic Average Gradient
  - SAG では、各データにおける勾配の履歴 (現在の Full gradient)、従って $\{\boldsymbol{y}_i^t\}_{i=1}^N$ を記憶しなければならないことが分かる
  - 比較的小規模のニューラルネットの学習であっても、勾配を記憶するためのメモリ使用量が大きいため、アルゴリズムは使い物にならなくなる
  - GD の速い収束と、SGD の速い計算時間という、双方の利点を受け継いだアルゴリズム
  - SAG の改良版として、SAGA アルゴリズムが存在 (詳細は省略)
  - 因みに、SAGA が何の略称なのかは不明

分散を抑えた確率的勾配降下法

- SAGA におけるパラメータの更新式は次のようになる [2]

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^{t} - \alpha_t \left[\frac{1}{N} \sum_{i=1}^{N} \boldsymbol{y}_i^{t-1} + \left(\boldsymbol{y}_{i_t}^{t} - \boldsymbol{y}_{i_t}^{t-1} \right) \right] \tag{27}$$

$$= \boldsymbol{w}^{t} - \alpha_t \left[\frac{1}{N} \sum_{i=1}^{N} \boldsymbol{y}_i^{t-1} + \left(\nabla f_{i_t}(\boldsymbol{w}^{t}) - \boldsymbol{y}_{i_t}^{t-1} \right) \right] \tag{28}$$

分散を抑えた確率的勾配降下法

- SVRG; Stochastic Variance Reduced Gradient
  - 二重ループの最適化アルゴリズムである
  - 外側のループでは、Full gradient(全データについての勾配の平均)$\nabla f(\boldsymbol{w}^t)$ を計算

$$\nabla f(\boldsymbol{w}^t) = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\boldsymbol{w}^t) = \frac{1}{N}\sum_{i=1}^{N} \frac{\partial}{\partial \boldsymbol{w}} f_i(\boldsymbol{w})\Big|_{\boldsymbol{w}=\boldsymbol{w}^t} \tag{29}$$

  - 内側のループでは、インデックス $i \in \{1, \ldots, N\}$ を選択し、次の式を用いてパラメータを更新していく

$$\boldsymbol{w} = \boldsymbol{w} - h\left[\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)\right] \tag{30}$$

  - 確率的勾配 $\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t)$ は、$\boldsymbol{w}$ と $\boldsymbol{w}^t$ における勾配の変化 $\nabla f(\boldsymbol{w}) - \nabla f(\boldsymbol{w}^t)$ を推定するための項

分散を抑えた確率的勾配降下法

- SVRG が完全な勾配 $\nabla f(\boldsymbol{w}^t)$ を計算している間、パラメータは一度も更新されないが、同じ時間で、SGD ではパラメータが N 回更新される
  $\Rightarrow$ 最初は、SGD の方が学習が進むことが予測される

分散を抑えた確率的勾配降下法

Algorithm 1: SVRG; Stochastic Variance Reduced Gradient [2]

parameters: $m =$ number of stochastic steps per epoch, $h =$ stepsize (learning rate)

for $s = 0, 1, \ldots$ **do**
  Compute and store full gradient $\nabla f(\boldsymbol{w}^t) = \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(\boldsymbol{w}^t)$
  `// Full pass through data`
  Set $\boldsymbol{w} = \boldsymbol{w}^t$
  for $t = 1$ **to** m **do**
    Pick $i \in \{1, \ldots, N\}$, uniformly at random
    Update using $\boldsymbol{w} = \boldsymbol{w} - h \left[\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t) \right]$
    `// Stochastic update`
  end for
  $\boldsymbol{w}^{t+1} = \boldsymbol{w}$
end for

分散を抑えた確率的勾配降下法

- アルゴリズムについての補足
  - SGD と比較すると SVRG の性能は良く、各イテレーションでの分散が小さい
  - SVRG と、ランダム化された座標降下法 (RCD; Randomized Coordinate Descent) とを結びつけるアルゴリズムが存在
  - SAGA が提唱された論文では、SAGA は SAG と SVRG の中間に位置付けられている
  - SVRG、SAGA、SAG、GD を一般化したアルゴリズムが登場している

目次

Federated Learning のための最適化アルゴリズム

- Federated Learning のための最適化アルゴリズム
  - SVRG(Stochastic Variance Reduced Gradient) アルゴリズムと、DANE(Distributed Approximate Newton) アルゴリズムを調べる
  - SVRG と DANE は一見無関係にみえるが、実は深く関連し合う

  - SVRG を改良した (Naive な)Federated SVRG について説明する
  - (Naive な)Federated SVRG を更に改良したアルゴリズムを、新たに提案する
    ⇐ 各デバイスに保存された訓練データの個数、訓練データのスパース性、各デバイス上の訓練データのパターンの相違について考慮

  - Federated Learning では、訓練データが Massively Distributed、Non-IID、Unbalanced であるという仮定を置く
  - これに加え、スパース性、Privacy Sensitive などの仮定を設けた

Federated Learning のための最適化アルゴリズム

- Federated Learning のためのアルゴリズムに必要な特徴
  1. アルゴリズムの開始時に、パラメータが既に最適値であったなら、そのアルゴリズムを何度実行しても、パラメータの値が変化しない
  2. 訓練データが単一のノードにしかないとき、パラメータが収束するまでに必要な、中央のノードとの通信回数は $\mathcal{O}(1)$ に抑えられること
  3. データの各特徴が、単一のノードにしか現れないとき (解こうとしている問題が完全に分離され、各デバイスがパラメータの一部を学習しているとき)、$\mathcal{O}(1)$ 回の通信回数の後に、パラメータが収束すること

     $\Leftarrow$ データの各次元が、ある 1 つのノードでは 1 になるが、他の全てのノードでは 0 になるような場合
  4. 各ノードが完全に同一なデータセットを有するとき、$\mathcal{O}(1)$ 回の通信回数の後に、パラメータが収束すること

Federated Learning のための最適化アルゴリズム

- Federated Learning のためのアルゴリズムに必要な特徴
  - 「収束する」とは、「十分に精度のある適当な解が得られる」ことを意味している
    $\Rightarrow \mathcal{O}(1)$ 回とは、各デバイスと中央のノード間で、たった 1 度だけやり取りすることに相当

  - (1) は、全ての最適化問題において、考慮する価値がある設定
  - (2) と (3) は、Federated Learning における極端なケース
  - (4) は、従来の最適化問題における設定 (中央の少数のノードが多量のデータを保持している状況)
    $\Rightarrow$ (4) は、Federated Learning においては最も重要でない

Federated Learning のための最適化アルゴリズム

Algorithm 2: SVRG; Stochastic Variance Reduced Gradient (Recall) [2]

parameters: $m =$ number of stochastic steps per epoch, $h =$ stepsize (learning rate)

for $s = 0, 1, \ldots$ **do**
  Compute and store full gradient $\nabla f(\boldsymbol{w}^t) = \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(\boldsymbol{w}^t)$
  `// Full pass through data`
  Set $\boldsymbol{w} = \boldsymbol{w}^t$
  for $t = 1$ **to** m **do**
    Pick $i \in \{1, \ldots, N\}$, uniformly at random
    Update using $\boldsymbol{w} = \boldsymbol{w} - h \left[\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t) \right]$
    `// Stochastic update`
  end for
  $\boldsymbol{w}^{t+1} = \boldsymbol{w}$
end for

Federated Learning のための最適化アルゴリズム

- SVRG; Stochastic Variance Reduced Gradient
  - 外側のループでは、Full gradient(全データについての勾配の平均)$\nabla f(\boldsymbol{w}^t)$ を計算
  - 内側のループでは、確率的勾配によるパラメータの更新を m 回実行
    $\Rightarrow$ m は、データ数 N の 1 倍から 5 倍程度の値に設定
    $\Rightarrow$ m は、実用上は N にすることが多い

  - 内側のループでは、データ点 i における勾配 $\nabla f_i(\boldsymbol{w}), \nabla f_i(\boldsymbol{w}^t)$ を計算

    $\Rightarrow$ これらの勾配の差 $\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t)$ を求めて、$\boldsymbol{w}$ と $\boldsymbol{w}^t$ における Full gradient の差分 $\nabla f(\boldsymbol{w}) - \nabla f(\boldsymbol{w}^t)$ を推定するための項とする

    $\Rightarrow$ $\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)$ は、$\nabla f(\boldsymbol{w})$ の不偏推定量を導く

Federated Learning のための最適化アルゴリズム

- SVRG; Stochastic Variance Reduced Gradient
  - 更新中の $\boldsymbol{w}$ と、固定された $\boldsymbol{w}^t$ との差が小さければ、$\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t)$ も小さくなるので、$\nabla f(\boldsymbol{w})$(の予測値) に加わるノイズも小さくなっていると予想できる
  - 内側のループを実行する度に、$\boldsymbol{w}$ が $\boldsymbol{w}^t$ から離れていくので、$\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t)$ が増大し、従って Full gradient$\nabla f(\boldsymbol{w}^t)$ に加わるノイズが増大
    $\Rightarrow$ 外側のループが実行され、新たな Full gradient$\nabla f(\boldsymbol{w}^{t+1})$ が計算されると、Full gradient に加わるノイズは再び小さくなる
  - 関数 $f = \frac{1}{N}\sum_i f_i$ が λ-strongly convex function で、各 f_i が L-smooth function ならば、収束度合いは次のように表される (詳細は論文を参照)

$$\mathbb{E}\left[f(\boldsymbol{w}^t) - f(\boldsymbol{w}^*)\right] \leq c^t \left[f(\boldsymbol{w}^0) - f(\boldsymbol{w}^*)\right] \tag{31}$$

$\boldsymbol{w}^*$ は $f(\boldsymbol{w})$ を最小化する最適解、$c = \Theta\left(\frac{1}{mh}\right) + \Theta(h)$(詳細は略)

Federated Learning のための問題設定

- Federated Learning のための問題設定
  - 解くべき問題は、経験損失 $f(\boldsymbol{w})$ の最小化であった

$$\min_{\boldsymbol{w}\in\mathbb{R}^D} f(\boldsymbol{w}), \qquad f(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} f_i(\boldsymbol{w}) \tag{32}$$

  - 各 f_i は凸関数で、訓練データ $\{\boldsymbol{x}_i, y_i\}_{i=1}^N$ が与えられている (多数のノードに不均等に分散)

- 分散学習のために、以下の記法を導入する
  - K をノード数とする
  - $\mathcal{P}_k$ を、ノード $k \in \{1, \ldots, K\}$ が持つ訓練データのインデックス集合とする
  - ノード k が持つ訓練データの個数を、$N_k = |\mathcal{P}_k|$ と表す
    $\Rightarrow k \neq l$ のとき $\mathcal{P}_k \cap \mathcal{P}_l = \varnothing$(空集合)、そして $\sum_{k=1}^K N_k = N$

Federated Learning のための問題設定

- Federated Learning のための問題設定
  - 経験損失 $f(\boldsymbol{w})$ を、次の手順で書き直す

$$F_k(\boldsymbol{w}) \equiv \frac{1}{N_k} \sum_{i \in \mathcal{P}_k} f_i(\boldsymbol{w}) \tag{33}$$

$$\begin{aligned} f(\boldsymbol{w}) &= \frac{1}{N} \sum_{i=1}^{N} f_i(\boldsymbol{w}) = \frac{1}{N} \sum_{k=1}^{K} \sum_{i \in \mathcal{P}_k} f_i(\boldsymbol{w}) \\ &= \frac{1}{N} \sum_{k=1}^{K} N_k \cdot \frac{1}{N_k} \sum_{i \in \mathcal{P}_k} f_i(\boldsymbol{w}) = \frac{1}{N} \sum_{k=1}^{K} N_k F_k(\boldsymbol{w}) \end{aligned} \tag{34}$$

  - $F_k(\boldsymbol{w})$ は、各ノード k が最小化すべき目的関数である (凸関数)
  - これより、解くべき問題は次のように書き直される

$$\min_{\boldsymbol{w} \in \mathbb{R}^D} f(\boldsymbol{w}), \qquad f(\boldsymbol{w}) = \sum_{k=1}^{K} \frac{N_k}{N} F_k(\boldsymbol{w}) \tag{35}$$

Federated Learning のための問題設定

- Federated Learning のための問題設定
  - この問題を解くための最も簡単な手法は、次の通りである

$$\boldsymbol{w}_k^{t+1} = \underset{\boldsymbol{w}\in\mathbb{R}^D}{\arg\min}\, F_k(\boldsymbol{w}), \qquad \boldsymbol{w}^{t+1} = \sum_{k=1}^{K} \frac{N_k}{N} \boldsymbol{w}_k^{t+1} \tag{36}$$

  - 各ノード上で目的関数 F_k を最小化し、得られた解 $\boldsymbol{w}_k^{t+1}$ の N_k による重み付け和を $\boldsymbol{w}$ とする

  - この場合、上の問題を一度だけ解けば、解 $\boldsymbol{w}$ が得られる ($\boldsymbol{w}_k^{t+1}$ の右辺は t には依存しない) ので、各デバイスと中央のノードとの一度だけのやりとりで済む

Federated Learning のための問題設定

- Federated Learning のための問題設定
  - 上記のアルゴリズムは動作しない
    $\Leftarrow$ 全体の解 $\boldsymbol{w}$ が、個々の解 $\boldsymbol{w}_k$ の重み付け和になっているとは考えにくい

    $\Leftarrow$ 関数 F_k の形が、全ての k について等しいならば、重み付け和にはなっている
    $\Leftarrow$ 関数の形が全て等しいならば、単一のノード上で $\min_{\boldsymbol{w} \in \mathbb{R}^D} F_1(\boldsymbol{w})$ を解けばよいので、分散アルゴリズムを考える必要はない

  - 分散アルゴリズムを導出したいが、上記のアルゴリズムでは無意味
  - 但し、各ノード k が、目的関数 F_k に含まれる曲がり具合の情報 (Curvature information) を最大限活用できるようにしたい

Federated Learning のための問題設定

- Federated Learning のための問題設定
  - 分散アルゴリズムを導出するために、各 F_k に二次の項 $-\left(\boldsymbol{a}_k^t\right)^T \boldsymbol{w} + \frac{\mu}{2}||\boldsymbol{w} - \boldsymbol{w}^t||^2$ を摂動として加算する
  - そして、各ノードが次の問題を解くようにする

$$\boldsymbol{w}_k^{t+1} = \underset{\boldsymbol{w}\in\mathbb{R}^D}{\arg\max} \left(F_k(\boldsymbol{w}) - \left(\boldsymbol{a}_k^t\right)^T \boldsymbol{w} + \frac{\mu}{2}||\boldsymbol{w} - \boldsymbol{w}^t||^2 \right) \quad (37)$$

$$\boldsymbol{w}^{t+1} = \frac{1}{K}\sum_{k=1}^{K} \boldsymbol{w}_k^{t+1} \quad (38)$$

  - 各ノード k が、目的関数 F_k に含まれる曲がり具合の情報 (Curvature information) を最大限活用できるようにしたい
    $\Rightarrow$ 各ノードが最適化する関数の、ヘッセ行列は $\nabla^2 F_k + \mu \boldsymbol{I}$ となるので、関数 F_k に含まれる勾配の情報は、殆どそのまま保存される

Federated Learning のための問題設定

- Federated Learning のための問題設定
  - 以下の式を解きたいが、ベクトル $\boldsymbol{a}_k^t$ の決め方が分からない

$$\boldsymbol{w}_k^{t+1} = \arg\max_{\boldsymbol{w}\in\mathbb{R}^D} \left(F_k(\boldsymbol{w}) - \left(\boldsymbol{a}_k^t\right)^T \boldsymbol{w} + \frac{\mu}{2}||\boldsymbol{w} - \boldsymbol{w}^t||^2 \right)$$

  - $t \to \infty$ の極限では、$\boldsymbol{w}$ が最適 $(\boldsymbol{w} = \boldsymbol{w}^*)$ であるとき、上式の勾配が 0 となって欲しい

$$\begin{aligned} & \nabla \left(F_k(\boldsymbol{w}) - \left(\boldsymbol{a}_k^t\right)^T \boldsymbol{w} + \frac{\mu}{2}||\boldsymbol{w} - \boldsymbol{w}^t||^2 \right) \\ = \quad & \nabla F_k(\boldsymbol{w}) - \boldsymbol{a}_k^t + \mu\left(\boldsymbol{w} - \boldsymbol{w}^t\right) = 0 \end{aligned}$$

  - 即ち、$t \to \infty$ の極限では、$\boldsymbol{a}_k^t$ は次のようになって欲しい

$$\boldsymbol{a}_k^t = \nabla F_k(\boldsymbol{w}) + \mu\left(\boldsymbol{w} - \boldsymbol{w}^t\right) \simeq \nabla F_k(\boldsymbol{w}^*) \ \left(\because \boldsymbol{w}^* \simeq \boldsymbol{w}^t\right)$$

  - 但し、$\boldsymbol{w}^*$ を知らないので、$\boldsymbol{a}_k^t = \nabla F_k(\boldsymbol{w}^*)$ とはできない
    $\Rightarrow t \to \infty$ で、$\boldsymbol{a}_k^t \to \nabla F_k(\boldsymbol{w}^*)$ となるような更新式を編み出す

Federated Learning のための最適化アルゴリズム

- DANE; Distributed Approximate Newton
  - 先程の最適化問題は、双対問題と深く関連している
    $\Rightarrow$ 但し、上記のような問題がノード数分だけ存在するので、複雑である
  - DANE アルゴリズムでは、個々のノード上で解くための部分問題を構成することに主眼を置く
    $\Leftarrow$ 部分問題は、各ノード上のデータと、完全な勾配 $\nabla f(\boldsymbol{w}^t)$ にのみ依存
    $\Leftarrow$ Full gradient$\nabla f(\boldsymbol{w}^t)$ は、各デバイスと中央のノード間で、1 度だけやり取りすれば計算可能
  - DANE アルゴリズムを次に示す

Federated Learning のための最適化アルゴリズム

Algorithm 3: DANE; Distributed Approximate Newton [1]

input : regularizer $\mu \geq 0$, parameter η (default: $\mu = 0, \eta = 1$)

1: Initialize $\boldsymbol{w}^0$
2: **for** $t = 0, 1, \ldots$ **do**
3: Compute $\nabla f(\boldsymbol{w}^t) = \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(\boldsymbol{w}^t)$
4: Distribute $\nabla f(\boldsymbol{w}^t)$ to all machines
5: For each node $k \in \{1, \ldots, K\}$, solve

$$\boldsymbol{w}_k^{t+1} = \arg\min_{\boldsymbol{w} \in \mathbb{R}^D} \left(F_k(\boldsymbol{w}) - \left(\nabla F_k(\boldsymbol{w}^t) - \eta \nabla f(\boldsymbol{w}^t) \right)^T \boldsymbol{w} + \frac{\mu}{2} ||\boldsymbol{w} - \boldsymbol{w}^t||^2 \right) \tag{39}$$

6: Compute $\boldsymbol{w}^{t+1} = \frac{1}{K} \sum_{k=1}^{K} \boldsymbol{w}_k^{t+1}$
7: **end for**

Federated Learning のための最適化アルゴリズム

- DANE; Distributed Approximate Newton
  - 5 行目では、各ノードが次の部分問題を解いている

$$\boldsymbol{w}_k^{t+1} = \underset{\boldsymbol{w}\in\mathbb{R}^D}{\arg\min} \left(F_k(\boldsymbol{w}) - \left(\nabla F_k(\boldsymbol{w}^t) - \eta \nabla f(\boldsymbol{w}^t)\right)^T \boldsymbol{w} + \frac{\mu}{2}||\boldsymbol{w} - \boldsymbol{w}^t||^2 \right)$$

  - この部分問題の解を得るためのアルゴリズムは、特に指定されていない (何でもよい)
  - 問題を解くうえで、他のノードと通信する必要がない (通信コストを十分に小さくできる)

  - 各ノードが、摂動を加えた最適化問題を解くアルゴリズムの 1 つ
    $\Leftarrow \boldsymbol{a}_k^t$ を、$\boldsymbol{a}_k^t = \nabla F_k(\boldsymbol{w}^t) - \eta \nabla f(\boldsymbol{w}^t)$ と定義している
    $\Leftarrow \boldsymbol{w}^t \to \boldsymbol{w}^*$ ならば $\nabla f(\boldsymbol{w}^t) \to \nabla f(\boldsymbol{w}^*) = 0$ であるので、
    $\boldsymbol{a}_k^t \to \nabla F_k(\boldsymbol{w}^*)$ が成立

Federated Learning のための最適化アルゴリズム

- DANE; Distributed Approximate Newton
  - このアルゴリズムは、Federated Learning のために必要な条件 (2) と (3) を満たさない
  - $\mu = 0, \eta = 1$ ならば、条件 (4) を満たす
  - 任意の μ, η について、条件 (1) を満たす

  - このアルゴリズムでは、関数が 2 回微分可能であること、各ノードが独立同分布な標本を得られることを仮定

  - 正則化パラメータ μ の決め方については、改善の余地がある
    $\Leftarrow \mu = 0$ であれば、ノード数 K が小さいときは速やかに収束するが、K が増えるにつれて、急速に発散しやすくなる
    $\Leftarrow \mu$ を大きくすれば、アルゴリズムは安定する (発散しづらくなる) が、その分パラメータの収束は遅くなる

Federated Learning のための最適化アルゴリズム

- DANE と SVRG を融合したアルゴリズム
  - DANE アルゴリズムは、Federated Learning に適用できない (必要な条件を満たさない)
  - 部分問題の最適解を得る必要がある
    $\Leftarrow$ 簡単な問題なら可能だが、複雑な問題であれば計算コストが掛かり過ぎる
    $\Rightarrow$ 完全な最適解を得るのではなく、近似解を得るようなアルゴリズムに置き換える
    $\Rightarrow$ 部分問題を解くアルゴリズムとして、先程の SVRG を使用

  - SVRG アルゴリズムでは、外側のループの最初で、完全な勾配 $\nabla f(\boldsymbol{w}^t)$ を計算する必要があった (2 行目)
  - 完全な勾配 $\nabla f(\boldsymbol{w}^t)$ は、各ノードが部分問題を解く前の段階で、既に求まっている (3 行目)
    $\Rightarrow$ 各ノードでは、SVRG アルゴリズムの内側のループのみが実行され (後述)、完全な勾配を求める必要はない (既知であるとして扱う)

Federated Learning のための最適化アルゴリズム

Algorithm 4: SVRG; Stochastic Variance Reduced Gradient (Recall) [2]

parameters: $m =$ number of stochastic steps per epoch, $h =$ stepsize (learning rate)

for $s = 0, 1, \ldots$ **do**
  Compute and store full gradient $\nabla f(\boldsymbol{w}^t) = \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(\boldsymbol{w}^t)$
  `// Full pass through data`
  Set $\boldsymbol{w} = \boldsymbol{w}^t$
  for $t = 1$ **to** m **do**
    Pick $i \in \{1, \ldots, N\}$, uniformly at random
    Update using $\boldsymbol{w} = \boldsymbol{w} - h \left[\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t) \right]$
    `// Stochastic update`
  end for
  $\boldsymbol{w}^{t+1} = \boldsymbol{w}$
end for

Federated Learning のための最適化アルゴリズム

Algorithm 5: DANE; Distributed Approximate Newton (Recall) [1]

input : regularizer $\mu \geq 0$, parameter η (default: $\mu = 0, \eta = 1$)

1: Initialize $\boldsymbol{w}^0$
2: **for** $t = 0, 1, \ldots$ **do**
3: Compute $\nabla f(\boldsymbol{w}^t) = \frac{1}{N} \sum_{i=1}^{N} \nabla f_i(\boldsymbol{w}^t)$
4: Distribute $\nabla f(\boldsymbol{w}^t)$ to all machines
5: For each node $k \in \{1, \ldots, K\}$, solve

$$\boldsymbol{w}_k^{t+1} = \arg\min_{\boldsymbol{w} \in \mathbb{R}^D} \left(F_k(\boldsymbol{w}) - \left(\nabla F_k(\boldsymbol{w}^t) - \eta \nabla f(\boldsymbol{w}^t) \right)^T \boldsymbol{w} + \frac{\mu}{2} ||\boldsymbol{w} - \boldsymbol{w}^t||^2 \right) \tag{40}$$

6: Compute $\boldsymbol{w}^{t+1} = \frac{1}{K} \sum_{k=1}^{K} \boldsymbol{w}_k^{t+1}$
7: **end for**

Federated Learning のための最適化アルゴリズム

- DANE と SVRG との関係性
  - DANE アルゴリズムは、ある特定の条件下では、分散化されたバージョンの SVRG アルゴリズムと等価

  - 以下の 2 つのアルゴリズムは等価である
  - 下記の 2 つは、同一のパラメータ列 $\{\boldsymbol{w}^t\}$ を出力

  1. $\mu = 0, \eta = 1$ の下で DANE を実行し、その部分問題は SVRG アルゴリズムで解く
  2. 分散化されたバージョンの SVRG アルゴリズムを解く (後述)

  - 分散化された SVRG アルゴリズム (Naive Federated SVRG) を次に示す

Algorithm 6: Naive Federated SVRG (FSVRG) [1]

parameters: m =number of stochastic steps per epoch, h =stepsize, data partition $\{\mathcal{P}_k\}_{k=1}^K$

Initialize $\boldsymbol{w}^0$
for $t = 0, 1, \ldots,$ **do**
  Compute $\nabla f(\boldsymbol{w}^t) = \frac{1}{N}\sum_{i=1}^N \nabla f_i(\boldsymbol{w}^t)$, distribute to all machines
  for $k = 1$ to K **do in parallel** over nodes k **do**
    Initialize $\boldsymbol{w}_k = \boldsymbol{w}^t$
    for $s = 1$ to m **do**
      Sample $i \in \mathcal{P}_k$ uniformly at random
      Update using $\boldsymbol{w}_k = \boldsymbol{w}_k - h\left[\nabla f_i(\boldsymbol{w}_k) - \nabla f_i(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)\right]$
    end for
  end for
  Update using $\boldsymbol{w}^{t+1} = \boldsymbol{w}^t + \frac{1}{K}\sum_{k=1}^K \left(\boldsymbol{w}_k - \boldsymbol{w}^t\right)$
end for

Federated Learning のための最適化アルゴリズム

- 2 つのアルゴリズムが等価であることの簡潔な証明
  - SVRG アルゴリズムでは、$\nabla f(\boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\boldsymbol{w})$ の不偏推定量を得るために、$\nabla f(\boldsymbol{w}^t) + \nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t)$ という項を用いた
  - $\nabla f(\boldsymbol{w}^t)$ は、外側のループの最初で計算される (適当な時間間隔で計算され、暫くの間固定される)
  - 内側のループを実行する度に、データ $i \in \{1, \ldots, N\}$ について $\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t)$ を計算し、$\nabla f(\boldsymbol{w}^t)$ を補正する
  - DANE における部分問題を、各ノードが SVRG で解くことを考える (SVRG の内側のループのみが実行される)
  - SVRG で解くのは次の問題である ($\mu = 0, \eta = 1$)

$$
\begin{aligned}
& \min_{\boldsymbol{w} \in \mathbb{R}^D} \left(F_k(\boldsymbol{w}) - \left(\nabla F_k(\boldsymbol{w}^t) - \eta \nabla f(\boldsymbol{w}^t) \right)^T \boldsymbol{w} + \frac{\mu}{2} ||\boldsymbol{w} - \boldsymbol{w}^t||^2 \right) \\
= \quad & \min_{\boldsymbol{w} \in \mathbb{R}^D} \left(F_k(\boldsymbol{w}) - \left(\nabla F_k(\boldsymbol{w}^t) - \nabla f(\boldsymbol{w}^t) \right)^T \boldsymbol{w} \right) \qquad (41)
\end{aligned}
$$

Federated Learning のための最適化アルゴリズム

- 2 つのアルゴリズムが等価であることの簡潔な証明
  - SVRG で最小化しようとしている関数の勾配を求める

$$\begin{aligned} & \frac{\partial}{\partial \boldsymbol{w}} \left(F_k(\boldsymbol{w}) - \left(\nabla F_k(\boldsymbol{w}^t) - \nabla f(\boldsymbol{w}^t) \right)^T \boldsymbol{w} \right) \\ = \quad & \nabla F_k(\boldsymbol{w}) - \nabla F_k(\boldsymbol{w}^t) - \nabla f(\boldsymbol{w}^t) \end{aligned} \tag{42}$$

  - 上式について、$\nabla f(\boldsymbol{w}^t)$ は、DANE アルゴリズムの 3 行目で既に求まっている
  - 残りの $\nabla F_k(\boldsymbol{w})$ と $\nabla F_k(\boldsymbol{w}^t)$ の計算について考える

  - $\nabla F_k(\boldsymbol{w}) = \sum_{i \in \mathcal{P}_k} \nabla f_i(\boldsymbol{w})$ であるから、$\nabla F_k(\boldsymbol{w})$ は、$i \in \mathcal{P}_k$ である全ての i についての、勾配 $\nabla f_i(\boldsymbol{w})$ の足し合わせである ($\nabla F_k(\boldsymbol{w}^t)$ についても同様)

  - SVRG の内側のループでは、ある 1 つのインデックス $i \in \mathcal{P}_k$ をランダムに選び出し、パラメータ $\boldsymbol{w}$ を更新する

Federated Learning のための最適化アルゴリズム

- 2 つのアルゴリズムが等価であることの簡潔な証明
  - SVRG の内側のループでは、ある 1 つのインデックス $i \in \mathcal{P}_k$ をランダムに選び出し、パラメータ $\boldsymbol{w}$ を更新する

    $\Rightarrow$ 内側のループでは、1 つのデータ点 i についての確率的勾配だけを使って、$\nabla F_k(\boldsymbol{w}) - \nabla F_k(\boldsymbol{w}^t) - \nabla f(\boldsymbol{w}^t)$ を推定したい

    $\Rightarrow$ $\nabla f(\boldsymbol{w}^t)$ は既知であるから、残りの項 $\nabla F_k(\boldsymbol{w}) - \nabla F_k(\boldsymbol{w}^t)$ を、確率的勾配 $\nabla f_i(\boldsymbol{w}), \nabla f_i(\boldsymbol{w}^t)$ を使って近似する

  - SVRG のときと同じ方法で、次のように近似できる

$$\nabla F_k(\boldsymbol{w}) = \nabla F_k(\boldsymbol{w}^t) + \left(\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t)\right) \tag{43}$$

$$\nabla F_k(\boldsymbol{w}^t) = \nabla F_k(\boldsymbol{w}^t) + \left(\nabla f_i(\boldsymbol{w}^t) - \nabla f_i(\boldsymbol{w}^t)\right) \tag{44}$$

Federated Learning のための最適化アルゴリズム

- 2 つのアルゴリズムが等価であることの簡潔な証明
  - 以下は単に、$f(\boldsymbol{w})$ を $F_k(\boldsymbol{w})$ に置き換えているだけ

$$\nabla F_k(\boldsymbol{w}) = \nabla F_k(\boldsymbol{w}^t) + \left(\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t)\right) \tag{45}$$

  - $\nabla F_k(\boldsymbol{w}^t)$ を、内側のループが実行される度に計算するのは、計算コストの観点から避けたいので、適当な時間間隔で求めて、暫くの間は使い回すことにする

    $\Rightarrow$ 現在のパラメータ $\boldsymbol{w}$ についての勾配 $\nabla F_k(\boldsymbol{w})$ の不偏推定量を得るために、古いパラメータ $\boldsymbol{w}^t$ についての勾配 $\nabla F_k(\boldsymbol{w}^t)$ を、データ i についての項 $\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t)$ で補正する

Federated Learning のための最適化アルゴリズム

- 2 つのアルゴリズムが等価であることの簡潔な証明
  - このような近似の下で、内側のループで計算される勾配は、次のようになる

$$
\begin{aligned}
& \nabla F_k(\boldsymbol{w}) - \nabla F_k(\boldsymbol{w}^t) - \nabla f(\boldsymbol{w}^t) && (46)\\
= \quad & \left[\nabla F_k(\boldsymbol{w}^t) + \nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t)\right] \\
& \qquad - \left[\nabla F_k(\boldsymbol{w}^t) + \nabla f_i(\boldsymbol{w}^t) - \nabla f_i(\boldsymbol{w}^t)\right] - \nabla f(\boldsymbol{w}^t) && (47)\\
= \quad & \nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t) && (48)
\end{aligned}
$$

  - これより、DANE アルゴリズムの部分問題に、SVRG アルゴリズムを適用したとき、各ノードにおけるパラメータの更新式は次のようになる

$$
\boldsymbol{w} = \boldsymbol{w} - h\left[\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)\right] \tag{49}
$$

  - 上の更新式は、FSVRG(Algorithm 6) の 8 行目と等しい
    $\Rightarrow$ DANE($\mu = 0, \eta = 1$) の部分問題に SVRG を組み込んだアルゴリズムは、(Naive な)FSVRG と等価

Federated Learning のための最適化アルゴリズム

- Naive Federated SVRG についての補足
  - 外側のループについて、最後の更新式が次のようになっている

$$\boldsymbol{w}^{t+1} = \boldsymbol{w}^t + \frac{1}{K}\sum_{k=1}^{K}\left(\boldsymbol{w}_k - \boldsymbol{w}^t\right) \tag{50}$$

  - これは、DANE における次の式に等しい

$$\boldsymbol{w}^{t+1} = \frac{1}{K}\sum_{k=1}^{K}\boldsymbol{w}_k^{t+1} \tag{51}$$

  - $(\boldsymbol{w}_k - \boldsymbol{w}^t)$ は、各ノードから中央のサーバに送られる、パラメータの差分を表している
    $\Rightarrow$ $\boldsymbol{w}^t$ が中央のサーバから送られてきたパラメータ、$\boldsymbol{w}_k$ はノード k 上の訓練データを使って更新されたパラメータ
    $\Rightarrow$ $\boldsymbol{w}_k$ をそのまま送っても良いが、差分を送った方がデータを圧縮できる (通信量を削減)

Federated Learning のための最適化アルゴリズム

- SVRG アルゴリズムの導出
  - (Naive な)Federated SVRG を更に改良したアルゴリズムを、ここでは提案する
    ⇐ 各デバイスに保存された訓練データの個数、訓練データのスパース性、データが独立同分布でない (Non-IID) ことについて考慮

  - 各デバイス間では、訓練データの個数が大きく異なる
  - データのある特徴は、ごく少数のノードにだけ出現し、それ以外の大多数のノードには出現しないかもしれない
  - 各デバイス上のデータは、データ全体の分布を表していない (データ全体は、デバイスの地理的な位置やタイムゾーンなどが影響し、特定のパターンによってクラスタ化されているかもしれない)

Federated Learning のための最適化アルゴリズム

- SVRG アルゴリズムの導出 (記法の整理)
  - N: 訓練データの個数, K: ノード数, D: パラメータの次元
  - $\mathcal{P}_k$: ノード k が保持する訓練データのインデックスの集合
    $\Rightarrow \mathcal{P}_k \subseteq \{1, \ldots, N\}, \forall k \neq l \; \mathcal{P}_k \cap \mathcal{P}_l = \varnothing$
  - $N_k = |\mathcal{P}_k|$: ノード k が保持する訓練データの個数
    $\Rightarrow \sum_{k=1}^{K} N_k = N$
  - $N^j = \left|\left\{i \in \{1, \ldots, N\} \,|\boldsymbol{x}_i^T \boldsymbol{e}_j \neq 0\right\}\right|$: j 番目の次元が 0 ではない訓練データの個数
  - $N_k^j = \left|\left\{i \in \mathcal{P}_k |\boldsymbol{x}_i^T \boldsymbol{e}_j \neq 0\right\}\right|$: ノード k が保持する、j 番目の次元が 0 ではない、訓練データの個数
  - $\phi^j = N^j/N$: 全訓練データのうち、j 番目の次元が 0 でないものの割合
  - $\phi_k^j = N_k^j/N_k$: ノード k が保持する全訓練データのうち、j 番目の次元が 0 でないものの割合

Federated Learning のための最適化アルゴリズム

- $s_k^j = \phi^j / \phi_k^j$: 全訓練データと、ノード k が保持する訓練データにおける、j 番目の次元が 0 でない割合の比率
  $\Rightarrow$ s_k^j が小さいとき、ノード k は、次元 j に関する訓練データを、他のノードよりも多く持っていることを意味している
  $\Rightarrow$ s_k^j の逆数は、ノード k が持っている訓練データは、次元 j に関してどの程度レアかを表すと考えられる

- $\boldsymbol{S}_k = \mathrm{diag}\left(s_k^j\right)$: s_k^j を並べた対角行列 $(\boldsymbol{S}_k \in \mathbb{R}^{D \times D})$
- $\omega^j = \left|\left\{\mathcal{P}_k | n_k^j \neq 0\right\}\right|$: 次元 j が 0 ではない訓練データを持っている、ノードの数

- $a^j = K / \omega^j$: 次元 j が 0 でない訓練データが、ノードに出現する割合
  $\Rightarrow$ 次元 j のレア度を表す
- $\boldsymbol{A} = \mathrm{diag}\left(a^j\right)$: a^j を並べた対角行列 $(\boldsymbol{A} \in \mathbb{R}^{D \times D})$

- 提案された Federated SVRG アルゴリズムを、次に示す

Federated Learning のための最適化アルゴリズム

Algorithm 7: Federated SVRG (FSVRG) [1]

parameters: h =stepsize, data partition $\{\mathcal{P}_k\}_{k=1}^K$, diagonal matrix $\boldsymbol{A}, \boldsymbol{S}_k \in \mathbb{R}^{D\times D}$ for $k \in \{1,\ldots,K\}$

for $t = 0, 1, \ldots$ **do**
  Compute $\nabla f(\boldsymbol{w}^t) = \frac{1}{N}\sum_{i=1}^N \nabla f_i(\boldsymbol{w}^t)$
  for $k = 1$ to K **do in parallel** over nodes k **do**
    Initialize $\boldsymbol{w}_k = \boldsymbol{w}^t$, $h_k = h/N_k$
    Let $\{i_s\}_{s=1}^{N_k}$ be random permutation of $\mathcal{P}_k$
    for $s = 1, \ldots, N_k$ **do**
      $\boldsymbol{w}_k = \boldsymbol{w}_k - h_k \left[\boldsymbol{S}_k \left[\nabla f_{i_s}(\boldsymbol{w}_k) - \nabla f_{i_s}(\boldsymbol{w}^t)\right] + \nabla f(\boldsymbol{w}^t)\right]$
    end for
  end for
  $\boldsymbol{w}^t = \boldsymbol{w}^t + \boldsymbol{A}\sum_{k=1}^K \frac{N_k}{N}\left(\boldsymbol{w}_k - \boldsymbol{w}^t\right)$
end for

Federated Learning のための最適化アルゴリズム

- Federated SVRG アルゴリズムの改良点
  1. ノードごとにステップサイズを変更: $h_k = h/N_k$
  2. 各ノードが保持するデータ数に比例した更新量: $\frac{N_k}{N}\left(\boldsymbol{w}_k - \boldsymbol{w}^t\right)$
  3. 確率的勾配の各次元に対するスケーリング: $\boldsymbol{S}_k$
  4. パラメータの差分の各次元に対するスケーリング: $\boldsymbol{A}\left(\boldsymbol{w}_k - \boldsymbol{w}^t\right)$

  - ~~個人的に、このようなアルゴリズムの細工にはあまり魅力を感じない~~
  - ~~このような細工よりも、単純にデータ数を増やせば良いんじゃないか?~~

  - このアルゴリズムは、他のアルゴリズムよりも圧倒的に収束が速い

Federated Learning のための最適化アルゴリズム

1. ノードごとにステップサイズを変更: $h_k = h/N_k$
   - Naive Federated SVRG では、各ノードでのパラメータの更新回数は、m 回に統一されていた
     $\Rightarrow$ しかし、各ノードが保持する訓練データの数は大きく異なるのだから、パラメータの更新回数を、全ノードにわたって同じにするのは良くない
     $\Rightarrow$ それゆえ、各ノードでは、自身が持つ全ての訓練データを使って、パラメータを更新

   - ステップサイズを N_k に反比例するように設定し、各ノードのパラメータ更新量が同程度の大きさになるようにする

Federated Learning のための最適化アルゴリズム

2 各ノードが保持するデータ数に比例した更新量: $\frac{N_k}{N}\left(\boldsymbol{w}_k - \boldsymbol{w}^t\right)$

- Naive Federated SVRG では、以下のようにパラメータ $\boldsymbol{w}$ が更新された

$$
\begin{aligned}
\boldsymbol{w}^{t+1} &= \boldsymbol{w}^t + \frac{1}{K}\sum_{k=1}^{K}\left(\boldsymbol{w}_k - \boldsymbol{w}^t\right) \\
&= \boldsymbol{w}^t + \frac{h}{K}\sum_{k=1}^{K}\sum_{i}\left[\nabla f_i(\boldsymbol{w}_k) - \nabla f_i(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)\right] \\
&= \boldsymbol{w}^t + \frac{h}{K}\sum_{k=1}^{K}\sum_{i}\left[\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)\right] \\
&\simeq \boldsymbol{w}^t + \frac{h}{K}\sum_{k=1}^{K}\mathbb{E}\left[\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)\right] \\
&= \boldsymbol{w}^t + \frac{h}{K}\mathbb{E}\left[\sum_{k=1}^{K}\left(\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)\right)\right]
\end{aligned}
$$

Federated Learning のための最適化アルゴリズム

- 上式における i は、$\mathcal{P}_k$ から重複を許して、適当に選択された m 個のインデックスであるとする (但し、最後の式における i は、$\mathcal{P}_k$ から適当に選択されたインデックスである)
- また、ある時点において、全ての $k \in \{1, \ldots, K\}$ について $\boldsymbol{w}_k = \boldsymbol{w}$ であるとしている

- ここでは、適当な α_k を導入して、$\nabla f(\boldsymbol{w}^t)$ の不偏推定量が得られるようにしたい (i は $\mathcal{P}_k$ から適当に選択されたもの)
  $\Leftarrow$ 勾配降下法などの、目的関数の勾配のみを用いたアルゴリズム (一次法; First-order method) に対して求められる性質

- 即ち、以下が成り立って欲しい ($\boldsymbol{w}^t$ に足される量が、$\nabla f(\boldsymbol{w}^t)$ に比例する)

$$\mathbb{E}\left[\sum_{k=1}^{K} \alpha_k \left(\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)\right)\right] = \nabla f(\boldsymbol{w}^t) \tag{52}$$

Federated Learning のための最適化アルゴリズム

- α_k は次のようにして求められる

$$\begin{aligned}
& \mathbb{E}\left[\sum_{k=1}^{K} \alpha_k \left(\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)\right)\right] \\
= & \sum_{k=1}^{K} \alpha_k \mathbb{E}\left[\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)\right] \\
\simeq & \sum_{k=1}^{K} \alpha_k \frac{1}{N_k} \sum_{i \in \mathcal{P}_k} \left[\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)\right] \qquad (53)
\end{aligned}$$

- $\alpha_k = N_k/N$ とすることで以下を得る

$$= \frac{1}{N} \sum_{k=1}^{K} \sum_{i \in \mathcal{P}_k} \left[\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)\right] \simeq \nabla f(\boldsymbol{w})$$

Federated Learning のための最適化アルゴリズム

- これより、各ノードで計算されたパラメータの差分が、N_k に比例するように、更新量を調節することが考えられる

$$
\begin{aligned}
\boldsymbol{w}^{t+1} &= \boldsymbol{w}^t + \frac{h}{K}\mathbb{E}\left[\sum_{k=1}^{K}\frac{N_k}{N}\left(\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)\right)\right] \\
&= \boldsymbol{w}^t + \frac{h}{K}\frac{N_k}{N}\mathbb{E}\left[\sum_{k=1}^{K}\left(\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)\right)\right]
\end{aligned}
$$

Federated Learning のための最適化アルゴリズム

3 確率的勾配の各次元に対するスケーリング: $\boldsymbol{S}_k$

- 行列 $\boldsymbol{S}_k$ は、$s_k^j = \phi^j / \phi_k^j$ を対角成分にもつ

- データ数が $N = 10^6$、ノード数が $K = 10^3$ であるとする
- 次元 j が非零になるデータは 10^3 個あるが、その全てが、ある 1 つのノード k に集中しているとする
- このとき $s_k^j = \phi^j / \phi_k^j = 10^{-3}$ であるので、ノード k でパラメータ $\boldsymbol{w}$ が更新されるとき、次元 j の更新量は 10^3 分の 1 倍される

- 行列 $\boldsymbol{S}_k$ でスケーリングを行わない場合、ノード k における、次元 j に対する更新量は、他のノードの 10^3 倍程度になる (或いは、パラメータ $\boldsymbol{w}$ の次元 j は、ノード k でしか更新されない)
  $\Rightarrow$ パラメータ $\boldsymbol{w}$ の次元 j について、急速に発散してしまう恐れがある

- パラメータ $\boldsymbol{w}$ の更新に使用する、勾配の (推定量の) 各次元が、大体同じになるように揃える役割 (個人的には、まだよく理解できていない)

Federated Learning のための最適化アルゴリズム

4 パラメータの差分の各次元に対するスケーリング: $\boldsymbol{A}\left(\boldsymbol{w}_k - \boldsymbol{w}^t\right)$

- 行列 $\boldsymbol{A}$ は、$a_j = K/\omega_j$ を対角成分にもつ

- 行列 $\boldsymbol{A}$ の各成分は、ω_j に反比例している
  $\Rightarrow$ 次元 j が非零であるデータを持つノード数 ω_j が少ないほど、パラメータ $\boldsymbol{w}$ の次元 j に対する更新量は大きくなる

  $\Rightarrow \omega_j$ が小さいとき、少ないノードから、次元 j に関する情報が伝達されているので、その情報の価値は高い (パラメータ $\boldsymbol{w}$ の次元 j の更新に際して、大いに役立つ情報である) と考えられる (従って更新量を大きくする)

- 4 つの改善のうち、どれが最も効果的であったかは不明であるほか、これらの改善策が、相互に干渉し合うと思われる
- $h_k, \frac{N_k}{N}, \boldsymbol{S}_k, \boldsymbol{A}$ によって、適切なスケーリングを行うことで、各ノード上の訓練データ数のばらつき、データのスパース性などに対処できそうなのは、直感的には分かる

目次

最適化アルゴリズムの比較

- 最適化アルゴリズムの比較
  - Google+の投稿に、1 つ以上のコメントが付くかどうかを予測する、二値分類のタスク
  - L2 正則化項を含めたロジスティック回帰

  - Google+の投稿から作成したデータセットを、実験に使用
  - 各ユーザの投稿が、各ユーザの使うデバイス上に保存されているという状況を想定
  - 10^2 以上のパブリックな投稿を英語で行っている、10^4 のユーザをランダムに選択
  - 選択されたユーザの投稿を集めて、そのうちの 75% にあたる $N = 2,166,693$ 個を訓練データとした

  - 入力データとなる投稿は Bag-of-words 形式に変換された
    $\Leftarrow$ (全投稿データを基に得られた) 最頻出単語 $20,000$ 語と、それ以外の不明な単語の出現回数と、バイアス項を加えたことで、パラメータ数は $D = 20,002$ となった

最適化アルゴリズムの比較

- 最適化アルゴリズムの比較
  - 各ユーザが投稿する内容は一般に異なるので、各ユーザが保持するデータの特徴も全く異なる
    ⇒ 各ユーザの投稿内容は、全データの分布内で、クラスタを形成していると考えられる
    ⇒ 各ユーザの保持するデータは、独立同分布標本とは仮定できない
    ⇒ もしそのような仮定をすると、全てのユーザが、あらゆる内容の投稿を満遍なくしていることになる

  - 入力データは Bag-of-words 形式であるから、非常にスパースである
    ⇒ 殆どの投稿は、ごく一部の単語しか含んでいないので、入力データの中で非零になる要素数が少ない
    ⇒ 殆どの次元は、非零になる (その次元に対応する特徴が出現する) 確率が低い
    ⇒ 入力データの各特徴の出現確率を、次の図 1 に示す (不明な単語を表す次元の出現率が高くなっているが、これは実際とは異なる)

最適化アルゴリズムの比較

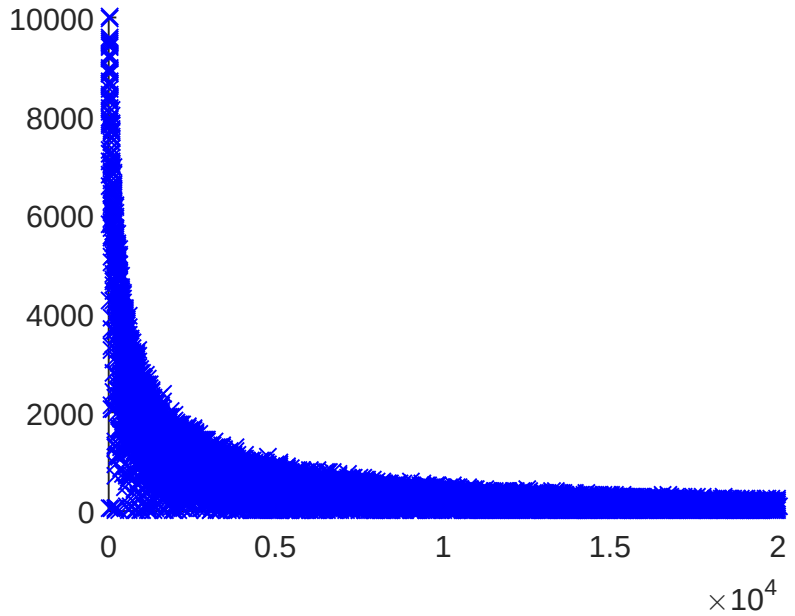

Figure 1: Features vs. appearance on nodes. The x-axis is a feature index, and the y-axis represents the number of nodes where a given feature is present.

図 1: 特徴の出現確率のヒストグラム [1]

最適化アルゴリズムの比較

- 最適化アルゴリズムの比較
  - Google+の投稿に、1 つ以上のコメントが付くかどうかを予測する、二値分類のタスク
  - L2 正則化項を含めたロジスティック回帰の、テストデータに対する誤差 (分類誤り) を示す

  - 全ての投稿に対して、コメントが付かない (-1) という予測をすると、誤差は 33.16%
  - 入力データを全て用いて、通常のロジスティック回帰を行うと、誤差は 26.27%
  - 各ユーザの全ての投稿に対して、同一の予測をすると、誤差は 17.14%
    $\Rightarrow$ 個々の投稿内容ではなく、誰が投稿しているのかに着目して、コメントが付くかどうかの予測を行った方が、精度が良くなることを示唆 (直感的にもそうである)

最適化アルゴリズムの比較

- 最適化アルゴリズムの比較
  - 全ユーザで共通のモデルを基に、各ユーザごとにチューニングすることで、精度を向上させられる可能性
    $\Rightarrow$ もし精度が大幅に向上するのであれば、各ノードごとに、異なるデータのパターンが出現していることも確認できる
  - 各最適化アルゴリズムでの性能比較を行う
  - 提案手法 (Federated SVRG; Algorithm 7) のハイパーパラメータは、ステップサイズ h のみであるが、性能が最も良くなるものを選んだ

最適化アルゴリズムの比較

- 最適化アルゴリズムの比較
  - オフラインアルゴリズムで得られる最大の性能 (OPT)、勾配降下法 (GD)、CoCoA+アルゴリズム (詳細は略)、Federated SVRG アルゴリズム (FSVRG)、ランダムにデータをシャッフルした上で行う Federated SVRG(FSVRGR) の性能を比較
    - GD: 個々のノードが、自身が持つ訓練データを全て使って、勾配降下法を行っている
    - CoCoA+: このスライドでは省略
    - FSVRG: この論文における提案手法
    - FSVRGR: 各ノードが保持するデータ数は変化させない (ばらつきは維持する) が、データを一旦全て集めて、ランダムにシャッフルした上で、個々のノードに戻している (各ノードが保持するデータが IID ではなくても、FSVRG アルゴリズムが動作することを示すために用意してある)
  - 目的関数の推移、またはテストデータに対する分類誤差の推移を縦軸に、そして各デバイスと中央のサーバとの一連のやり取り (Round) の回数を横軸に取ったグラフを、次の図 2 に示す

最適化アルゴリズムの比較

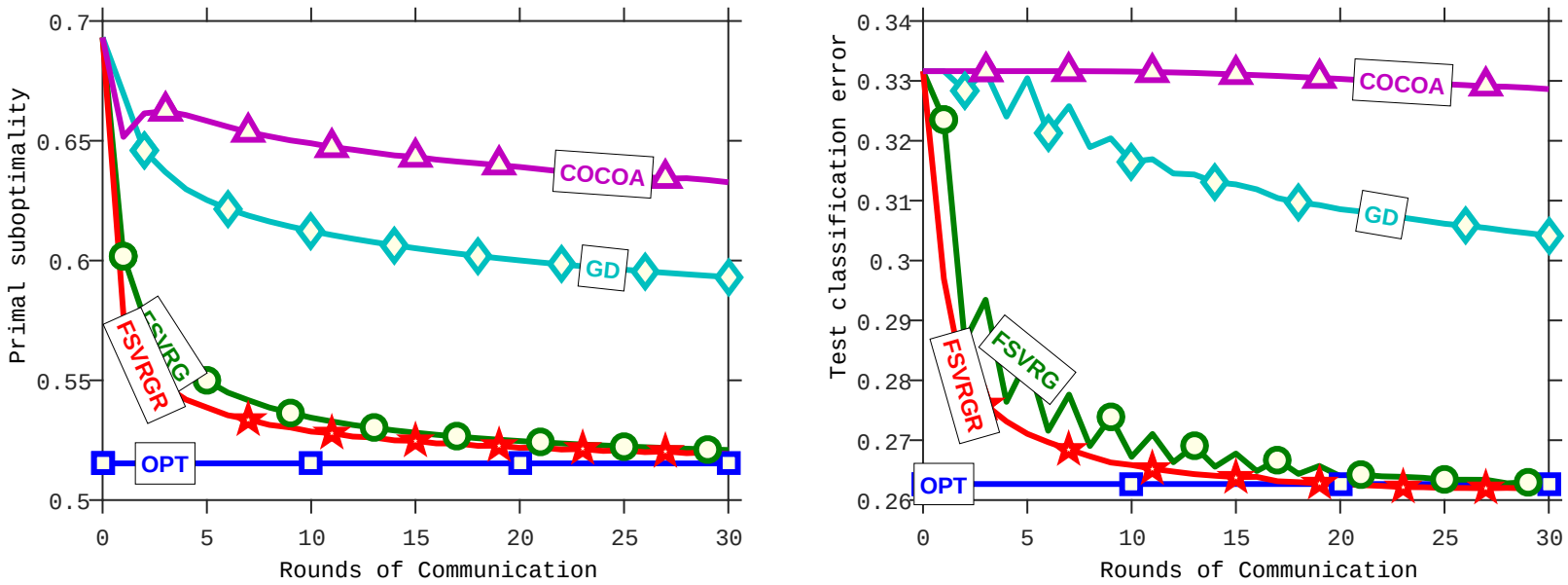

Figure 2: Rounds of communication vs. objective function (left) and test prediction error (right).

図 2: 最適化アルゴリズムの性能比較 [1]

最適化アルゴリズムの比較

- 最適化アルゴリズムの比較
  - FSVRG は、僅か 30 回程度の Round で収束しており、Federated Optimization における課題を最初に解決したアルゴリズムといえる

    ⇒ CoCoA+や、それ以外の通信効率の良い (Communication-Efficient) 分散アルゴリズムでも、収束していない (学習が上手く行っていない)

    ⇒ 学習が上手く行かないのは、他のアルゴリズムでは、全てのデバイスが独立同分布なデータをもつと仮定しているため

  - FSVRG と FSVRGR の結果の差はごく僅かであり、Naive Federated SVRG を改良することで、各デバイスが持つデータの分布に左右されない (ロバストな) アルゴリズムを得られた

目次

結論

- Federated Learning の特徴
  - Massively Distributed、Non-IID、Unbalanced、Privacy Sensitive、Sparse の 5 つが挙げられる
  - これらの性質をもつ厳しい環境下においても、効率的に学習が進むアルゴリズムを設計可能
- 今後の研究の方向性について
  - 非同期版のアルゴリズムの開発や、アルゴリズムの理論的な解明 (特に収束性)
  - 非凸関数の最適化 (ニューラルネットが代表例) についての理論的な解明
  - 全ユーザ間で共有されるモデルと、個々のユーザに特化したモデル双方の活用

参考文献

[1] Jakub Konecný, H. Brendan McMahan, Daniel Ramage, and Peter Richtárik.
Federated optimization: Distributed machine learning for on-device intelligence.
arXiv:1610.02527, 2016.

[2] Pradeep Ravikumar.
Stochastic optimization methods.
http://www.cs.cmu.edu/~pradeepr/convexopt/Lecture_Slides/stochastic_optimization_methods.pdf, 2017.

[3] Suzuki Taiji.
機械学習におけるオンライン確率的最適化の理論.
https://www.slideshare.net/trinmu/stochasticoptim2013, 2013.