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

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 9, 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
    ⇒ 条件付き確率場や、ニューラルネットワーク
    ⇒ 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
    ⇒ 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)

    ⇒ ノード間での通信時間がボトルネックとなる可能性
    ⇒ 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
    $\Rightarrow$ 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
    $\Rightarrow$ 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
   ⇒ 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
   ⇒ 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
   ⇒ 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
$\Rightarrow$ 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)
$\Rightarrow$ 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
$\Rightarrow$ 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)
$\Rightarrow$ 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 δ について
  - δ 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 δ 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 δ の使途は、元の訓練データ $\{x_i, y_i\}$ に比べて限られる
  $\Rightarrow$ モデルの訓練以外に、殆ど使い道がない
  $\Rightarrow$ 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

  $\Rightarrow$ 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)

  $\Rightarrow$ 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
  $\Rightarrow$ 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
  $\Leftrightarrow$ 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 $f(\boldsymbol{w})$ の勾配の不変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

$$\begin{aligned}
\nabla_j f(\boldsymbol{w}^t) &= \frac{\partial}{\partial w_j} f(\boldsymbol{w}^t) && (13) \\
&= \frac{1}{N} \sum_{i=1}^{N} \frac{\partial}{\partial w_j} f_i(\boldsymbol{w}) \bigg|_{\boldsymbol{w}=\boldsymbol{w}^t} && (14) \\
&= \frac{1}{N} \sum_{i=1}^{N} \nabla_{w_j} f_i(\boldsymbol{w}^t) && (15)
\end{aligned}$$

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
    ⇒ 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

    ⇒ 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
    ⇒ 「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) [2]

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

  - 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

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

$$
\begin{aligned}
\boldsymbol{w}^{t+1} &= \boldsymbol{w}^t - \frac{\alpha_t}{N} \sum_{i=1}^{N} \boldsymbol{y}_i^t & (21) \\
&= \boldsymbol{w}^t - \frac{\alpha_t}{N} \left[\sum_{i=1}^{N} \boldsymbol{y}_i^{t-1} + \left(\boldsymbol{y}_{i_t}^t - \boldsymbol{y}_{i_t}^{t-1} \right) \right] & (22) \\
&= \boldsymbol{w}^t - \frac{\alpha_t}{N} \left[\sum_{i=1}^{N} \boldsymbol{y}_i^{t-1} + \left(\nabla f_{i_t}(\boldsymbol{w}^t) - \boldsymbol{y}_{i_t}^{t-1} \right) \right] & (23)
\end{aligned}
$$

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

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

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
    ⇒ 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
    ⇒ 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

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

    ⇒ 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
    ⇒ 後述の SAGA で計算される勾配の期待値は、勾配 $f(\boldsymbol{w})$ の不変推定量に一致

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 [1]

$$\begin{aligned} \boldsymbol{w}^{t+1} &= \boldsymbol{w}^t - \alpha_t \left[\frac{1}{N} \sum_{i=1}^{N} \boldsymbol{y}_i^{t-1} + \left(\boldsymbol{y}_{i_t}^t - \boldsymbol{y}_{i_t}^{t-1} \right) \right] && (27) \\ &= \boldsymbol{w}^t - \alpha_t \left[\frac{1}{N} \sum_{i=1}^{N} \boldsymbol{y}_i^{t-1} + \left(\nabla f_{i_t}(\boldsymbol{w}^t) - \boldsymbol{y}_{i_t}^{t-1} \right) \right] && (28) \end{aligned}$$

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、全データ点についての完全な勾配 $\nabla f(\boldsymbol{w}^t)$ を計算

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

  - 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
  - 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 [1]

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

```
for s = 0, 1, ... do
    Compute and store full gradient ∇f(w^t) = (1/N) Σ_{i=1}^{N} ∇f_i(w^t)
    // Full pass through data
    Set w = w^t
    for t = 1 to m do
        Pick i ∈ {1, ..., N}, uniformly at random
        w = w − h [∇f_i(w) − ∇f_i(w^t) + ∇f(w^t)]    // Stochastic update
    end for
    w^{t+1} = w
end for
```

Where: $\nabla f(\boldsymbol{w}^t) = \frac{1}{N}\sum_{i=1}^{N} \nabla f_i(\boldsymbol{w}^t)$; $\boldsymbol{w} = \boldsymbol{w} - h\left[\nabla f_i(\boldsymbol{w}) - \nabla f_i(\boldsymbol{w}^t) + \nabla f(\boldsymbol{w}^t)\right]$; $\boldsymbol{w}^{t+1} = \boldsymbol{w}$

目次

Federated Learning のための最適化アルゴリズム

-

参考文献

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

[2] 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.