

DeepL Proに登録して、プレゼン資料を編集しましょう  
詳しくは[www.DeepL.com/pro](https://www.deepl.com/pro?cta=edit-document)をご覧ください。

分散型のリーダーストキャスティック勾配降下法

深層学習モデルの学習

|  |  |  |
| --- | --- | --- |
| Yunfei Teng∗,1 yt1208@nyu.edu | Wenbo Gao∗,2 wg2279@columbia.edu | Francois Chalus chalusf3@gmail.com |
| Anna Choromanska ac5455@nyu.edu | Donald Goldfarb goldfarb@columbia.edu | Adrian Weller aw665@cam.ac.uk |

アブストラクト

我々は、深層学習モデルを訓練するための通信制約下での分散最適化を考える。我々は、パラメータの更新が2つの力に依存する新しいアルゴリズムを提案する。それは、通常の勾配ステップと、現在最もパフォーマンスの高いワーカー（リーダー）によって指示される修正方向である。我々の手法は、パラメータ平均化スキームEASGD [1]とはいくつかの点で異なる。(i) 本手法の目的定式化は，元の最適化問題と比較して定常点の位置を変更しない． (ii) 異なるローカルミニマムに向かって下降するローカルワーカーを互いに（すなわち，それらのパラメータの平均に）引き寄せることによって生じる収束の減速を回避する． (iii) 本手法の更新は，対称性の呪い（対称的な非凸のランドスケープにおいて，一般化されていない準最適解に捕らわれる現象）を設計的に解消する．本論文では、提案アルゴリズムのバッチ版であるLGD（Leader Gradient Descent）と、そのストキャスティック版であるLSGD（Stochastic Variant）の理論的な解析を行う。最後に、我々のアルゴリズムの非同期バージョンを実装し、マルチリーダーの設定に拡張する。ここでは、それぞれがローカルリーダー（グループ内の最高のパフォーマー）で表されるワーカーのグループを形成し、ローカルリーダーとグローバルリーダー（全ワーカーの中で最高のパフォーマー）の2つの魅力的な力で構成される修正方向で各ワーカーを更新する。このマルチリーダーの設定は、グループを形成するローカルワーカーが1つの計算ノード内に存在し、異なるグループが異なるノードに対応するという、現在のハードウェアアーキテクチャとよく一致しています。畳み込みニューラルネットワークの学習において、我々のアプローチが最先端のベースラインと比較して良好な結果が得られたことを経験的に示した。

# はじめに

深層学習モデルとデータセットのサイズが大きくなるにつれ，その学習を分散した計算環境で並列化することがますます重要になっています．これらのモデルは，画像認識[2]，音声認識[3]，自然言語処理[4]など，機械学習をベースとした多くの最新システムの中核をなしています．本論文では，モデルではなくデータの並列化に焦点を当て，現在最も一般的に用いられている集団通信方式[5]を考慮しています．深層学習におけるデータ並列化の典型的なアプローチ[6, 7]では、複数のワーカーを使用し、異なるデータバッチに対してSGD[8]の亜種を実行します。そのため、有効なバッチサイズはワーカーの数だけ増加します。通信は、すべてのモデルが同期していることを保証し、各ワーカーが残りのすべてのワーカーにパラメータ勾配をブロードキャストするスキームに決定的に依存しています。

\*,1: 均等に貢献。アルゴリズムの開発と深層モデルへの実装。

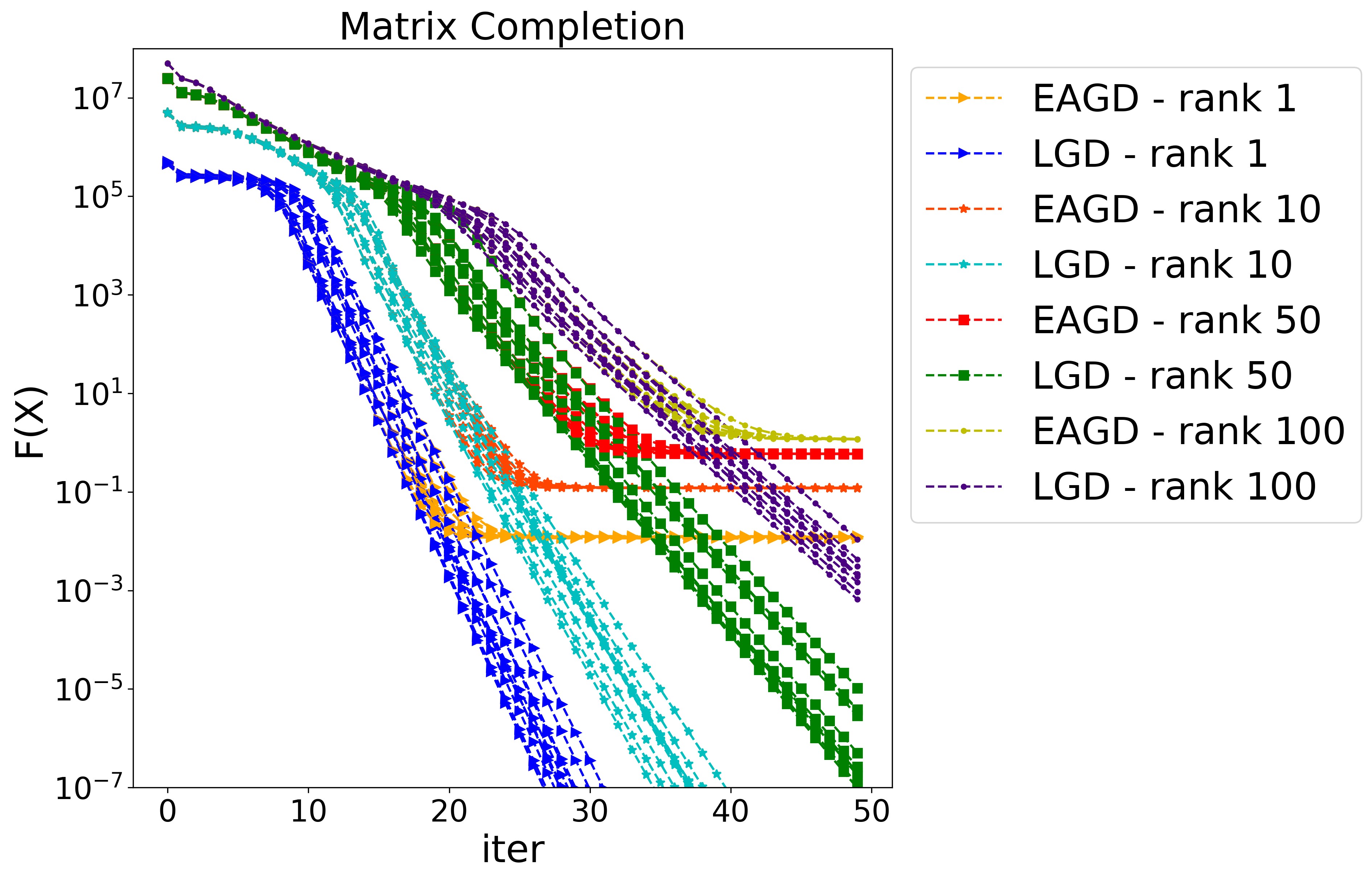
\*,2:同等の貢献。マトリックス補完の理論的解析と実装

33rd Conference on Neural Information Processing Systems (NeurIPS 2019), Vancouver, Canada.

これは、DOWNPOUR[9]やHorovod[11]などの手法がそうであるように、リングトポロジーに基づいて、中央にパラメータサーバを持たない分散型の拡張を行ったものがあります。これらの手法は、不安定性や発散を避けるために、（各バッチを処理した後に）頻繁に通信を行う必要があるため、通信コストがかかります。さらに，大きなバッチサイズで学習を行うと，一般化[12, 13, 14]や収束速度[15, 16]が低下することが多い．

EA(S)GD（Elastic Averaging (Stochastic) Gradient Decent）[1]と呼ばれる別のアプローチでは，ローカルワーカーのパラメータと，時間と空間の移動平均（すなわち，ローカルワーカーが計算したパラメータの移動平均）として計算された中央のパラメータとを結びつける弾性力が導入されている．この方法では，ワーカーが同じパラメータを持つ必要はなく，代わりに定期的にお互いに引き寄せられるように設計されているため，コミュニケーションの頻度を減らすことができます．しかし、EASGDの目的関数には定常点があり、その定常点は基本的な目的関数の定常点ではないため（補足の命題8参照）、これを最適化すると元の問題の解が最適でなくなる可能性があります。さらに、EASGD は平均化 SGD スキーム [17] を並列に拡張したものとみなすことができ、そのため平均化ポリシーの欠点を継承しています。非凸問題では、反復プログラムが (潜在的に大域的に最適な) 異なるローカルミニマムに収束する場合、平均化項によって反復プログラムが誤った方向に引きずられ、ローカルワーカーとマスターの両方の収束速度が大幅に低下します。最適化ランドスケープの対称的な領域では、異なるワーカーに関連する弾性力が互いに相殺され、マスターは異なるミニマムの間や最大値で永久に動けなくなり、ローカルワーカーはローカルミニマムやその上の斜面で動けなくなることがあります。これにより、任意の悪い一般化誤差が生じます。この現象を「対称性の呪い」と呼んでいます。ランドスケープシンメトリーは、深層学習[23, 24, 25, 26]を含む多数の非凸問題[18, 19, 20, 21, 22]でよく見られます。

|  |  |
| --- | --- |
| ーターを使用しています。提案されたアプローチは、複数のコンピュートノードからなる典型的なハードウェアアーキテクチャに容易に適応できます。 | 図1：EAGDとLGDで解く低ランク行列補完問題。次元*d*=1000、4つのランク*r*∈{1*,*10*,*50*,*100}を使用。各アルゴリズムの報告値は、各ステップにおける最高の作業者（合計で8人の作業者を使用）の値である。 |

本論文では、EASGDアップデートを再検討し、シンプルかつ強力な方法で修正することで、元の手法の上記の欠点を克服しました。本論文では、ローカルワーカーのパラメータの平均値に依存する弾性力を、ローカルワーカーとその中の現在のベストパフォーマー（リーダー）を結ぶ魅力的な力に置き換えることを提案する。本手法では，すべてのワーカーのパラメータを互いにブロードキャストすることによる通信オーバーヘッドを削減し，代わりにリーダーのみをブロードキャストすることで，ノード間の通信よりもはるかに高速な通信を実現している．このアプローチは、ローカルリーダーとグローバルリーダーの両方に惹かれるワーカーのグループ（計算ノードごとに1つ）を形成することに依存しており、このハードウェアアーキテクチャにうまく適応しています。通信のオーバーヘッドを減らすために、グローバルリーダーに関連する修正力は、ローカルリーダーに関連する修正力よりも少ない頻度で適用されます。

最後に、我々のL(S)GDアプローチは、EA(S)GDと同様に、ワーカーとリーダーの間の引っ張る力が小さく設定されている場合、最適化ランドスケープの広い谷を探索する傾向がある。この特性は、しばしばオプティマイザの汎化性能の向上につながる[27, 28]。

本論文は以下のように構成されている。第2節ではL(S)GDアプローチを紹介し，第3節では理論的な分析を行い，第4節では実証的な評価を行い，最後に第5節で本論文を結論付ける。理論的な証明と、追加の理論的・実証的な結果は、補足に含まれています。

# リーダー（ストキャスティック）勾配降下法「L(S)GD」アルゴリズム

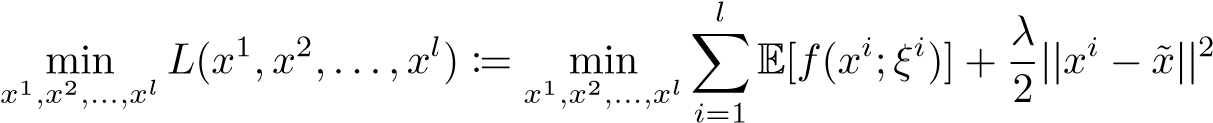
2.1 動機付けの例

図1は、弾性平均法が収束性を損なうことを示しています。この図を得るために、EAGD（Elastic Averaging Gradient Decent）とLGDを次のような形式の行列補完問題に適用しました。この問題は非凸ですが、すべての局所的最小化は大域的最小化であるという性質を持つことが知られています[18]。ランク*r*の4つの選択について、行列補完問題の10個のランダムなインスタンスを生成し、同じ出発点から初期化されたEAGDとLGDでそれぞれを解きました（8人の労働者を使用しています）。各アルゴリズムについて、すべての作業者を対象に、各反復において*最高の*目的値を得るまでの経過を報告します。図1は、各ランクの10回のランダムな実験の結果を示しています。

EAGDは最小化器に近づくと大きく減速することが明らかになっています。一般的に、EAGDの中心Xeは作業者の平均値に近いものになります。これは、すべてのローカルミニマムが大域的に最適であるにもかかわらず、作業者が異なるローカルミニマムに近づいている場合、行列補完問題の解としては不十分です。これにより、各ノードが最小化子*から離れる方向に*引っ張られることになり、EAGDが高精度の解を得ることは極めて困難になります。これに対し、LGDではこの問題はありません。この実験の詳細や、EAGDとLGDの違いを示す他の例は、別紙に掲載しています。

2.2 対称性を崩すアップデート

次に、L(S)GDアルゴリズムの基本的な更新方法を説明します。まず、シングルリーダーの設定で、並列計算環境で損失関数*L*を最小化する問題を考える。この最適化問題は次のように与えられる。

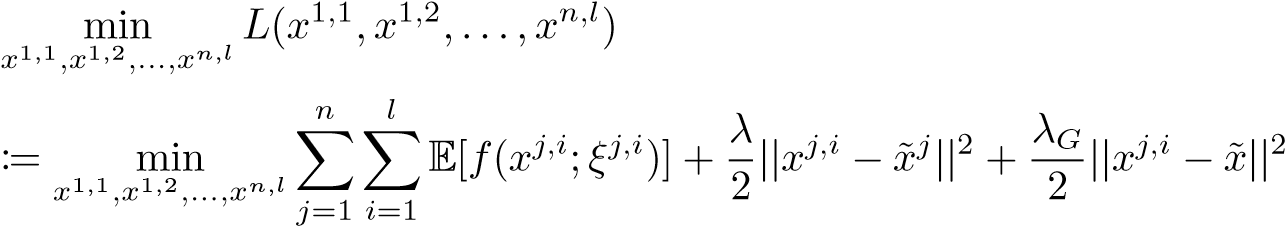
*,* (1)

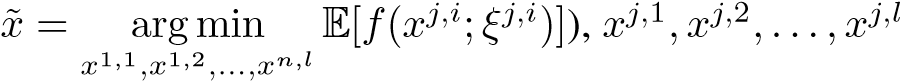
ここで、*l*は作業員の数、x1*,x*2*,...,xl*は作業員のパラメータ、*x*〜はリーダーのパラメータである。最高のパフォーマンスを発揮する作業者、すなわち*x*〜＝argmin E[*f*(*xi*; *ξi*)])、*ξ*は

x1*,x*2*,...,xl*

*λは、*労働者をリーダーに引き寄せる力の強さを示すハイパーパラメータである。理論的なセクションでは，E[*f*(*xi*; *ξi*)]を単に*f*(*xi*)と呼ぶことにする．この定式化は、さらにマルチリーダーの設定に拡張することができる。

この最適化問題は次のような形に修正されます。

*,* (2)

ここで、*n*はグループ数、*l*は各グループの労働者数、*x*〜*j*は*j*番目のグループのローカルリーダー（すなわち、*x*〜*j* = *argminxj,*1*,xj,*2*,...,xj,l* E[*f*(*xj,i; ξj,i*)] ）、*x*〜はグローバルリーダー（ローカルリーダーの中で最も優れた労働者、すなわち、パラメータは

*λ*と*λG*は、ワーカーをローカルリーダーとグローバルリーダーに引きつける力の強さを示すハイパーパラメータである。

LSGDアルゴリズムの更新を以下に示します（*t*は反復を表します）。*xj,i*に対するE[*f*(*xi*; *ξi*)]の確率的勾配を*gtj,i*(LGDの場合、勾配はすべての学習例に対して計算される)、*η*は学習率とすると、式3に示される最初の更新は、式2の目的に対して変数*xj,iに関する*勾配降下ステップを取ることによって得られる。

 (3)

ここで、*x*〜*jt*+1と*x*〜*t*+1は上で定義したローカルリーダーとグローバルリーダーです。

式3は、任意のワーカーの更新を記述するもので、通常の勾配ステップと2つの修正力で構成されています（シングルリーダーの設定では、*λG* = 0となるため、第3項は消滅します）。これらのアルゴリズム 1 LSGD アルゴリズム (非同期)

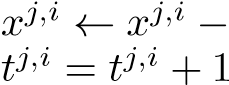
入力：引っ張り係数*λ,λG*、学習率*η*、ローカル/グローバル通信周期*τ,τG* 初期化。

ランダムに初期化されたx1*,*1*,*2*,...,xn,l*

イテレーションカウンターの設定 *tj,i* = 0

セット = argmin = argmin ;

|  |  |  |
| --- | --- | --- |
| *xj,*1*,...,xj,l* 繰り返し | x1*,*1*,...,xn,l* |  |
| すべての*j* = 1*,*2*,....,n*, *i* = 1*,*2*,...,l* を行います。  ランダムなサンプル*ξtj,ij,iの*描画 |  | *⊲* ワーカーごとに並行して行う |

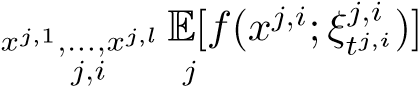
*ηgtj,i*(*xj,i*)

;

*nl*

*nlτ*が(P P *tj,i*)を分割する場合は

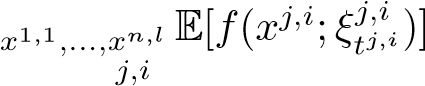
*j*=1 *i*=1

*x*〜*j* = argmin.*⊲* ローカルベストワーカーを決定する *xj,i* ←- *xj,i* - *λ*(*x* - *x*˜ *) ⊲* ローカルベストワーカーに引き上げる end if

*nl*

*nlτG*が(P P *tj,i*)を分割する場合は

*j*=1 *i*=1

*x*〜＝argmin.*⊲* グローバルベストワーカーを決定する *xj,i* ←- *xj,i* - *λG*(*x* - *x*˜*) ⊲* グローバルベストワーカーに引き上げる end if

エンドフォー

終了まで

の力がワーカー間のコミュニケーションメカニズムを構成し、すべてのワーカーを現時点でのベストなローカルおよびグローバル解に引き寄せることで、高速な収束を実現しています。EASGDとは対照的に、LSGDではワーカーが行う更新は対称性の呪いを破り、パフォーマンスの低いワーカーに本質的に影響される平均値にワーカーが引っ張られることに起因する収束の減速を回避します。本論文では、ワーカーを平均化されたパラメータに引き寄せる代わりに、ワーカーをリーダーに引き寄せるメカニズムを提案します。この更新の手法は、深層学習のための確率的勾配最適化の文脈では通常使用されない粒子群最適化アプローチ[29]に似ている。したがって、我々の手法は、確率的な設定と並列コンピューティング環境で深層学習モデルをトレーニングするための専用の粒子群最適化アプローチと見なすことができる。

次に、LSGDアルゴリズムの詳細を説明します。ここでは集団通信方式を採用しています。作業者間のコミュニケーション量を減らすために、作業者をリーダーに引き寄せる頻度は反復ごとよりも少なくすることが望まれます。また、実際には、各ワーカーの速度は異なる可能性があります。遅いワーカーを待たせることなく、通信効率を上げるために、非同期動作モードでアルゴリズムを実装します。この場合、通信期間はすべてのワーカーで計算された反復回数の合計に基づいて決定され、通信は*nlτ*または*nlτG反復*ごとに実行されます。ここで、*τ*および*τG*はそれぞれローカル通信期間およびグローバル通信期間を示します。実際には、上で説明したように、異なるグループに属するワーカー間の通信は、1つのグループ内のワーカー間の通信よりもコストがかかるため、*τG > τ*を使用する。通信が発生すると、集合的な通信スキームを利用するために、すべてのワーカーが同時に更新されます（すなわち、リーダーに引き寄せられます）。通信の間、ワーカーはそれぞれのローカル SGD オプティマイザを実行します。結果として得られる LSGD 法は非常にシンプルであり、Algorithm 1 に示されています。

次のセクションでは、我々のアプローチのシングル・リーダー・バッチ（LGD）とストキャスティック（LSGD）の理論的説明を行います。

# 理論的分析

ここでは、一般性を損なわない範囲で、リーダーが1人であると仮定します。複数のリーダーがいる場合の目的関数は、次のように与えられる。このセクションの証明は補足に譲ります。

3.1 確率 的強凸最適化のコンバージェンス率

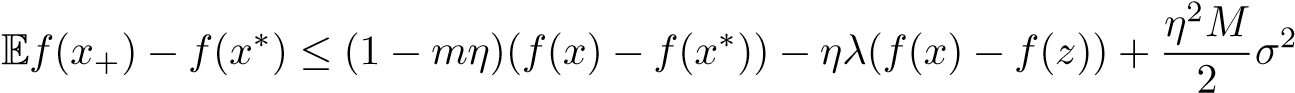
まず、確率的な強凸問題に対して、LSGDがSGDと同じ収束率を得ることを示す[30]。セクション3.3では、LGDがどのような場合に、勾配降下法よりも*優れた*探索方向を得ることができるかを議論する。非凸最適化についてはセクション3.2で説明します。セクション3.1を通して、*fは*通常、以下を満たす。

仮定1 *f*は*M-*Lipschitz-differentiableかつ*m-*strongly convex、すなわち、勾配∇*fは*k∇*f*(*x*)-∇*f*(*y*)k≦*Mkx* - ykを満たし、*fはf*(*y*)≧*f*(*x*)＋∇*f*(*x*)*T*(*y* - *x*)＋を満たす、の一意の最小化について。

3.1.1 コンバージェンス・レート

重要な技術的結果は、LSGDがSGDと同様の期待値の一段階下降を満たし、リーダーの引き込みに対応する追加の項を持つことです。純粋な」LSGDだけでなく、リーダーの更新頻度が低かったり、エラーがあったりするようなより実用的なバリエーションについても統一的な分析を行うために、一般的な繰り返し*x*+ = *x*-*η*(ge(*x*)+*λ*(*x*-*z*))を考えますが、ここで*z*≤は任意のガイド点です。つまり、zはx1*,... ,xpの最小化ではなく、f*(*z*) *f*(*xi*)を満たすものでもありません。ノードは*z*を更新するとき以外は独立して動作するので、各ノードのLSGDステップを個別に分析することができ、簡略化のために*x* = *xi*と書きます。

定理1です。2 *fが仮定1.*2*を満たすとする。* ge(*x*)*を*1 Var(ge(*x*≤))≦*σ* 1+*ν*√k∇*f*≤(*x*)k√*の*≤∇*f*(*x*)*の不偏推定量とし、*1 *zを任意の点とする。η,λがη* (2*M*(*ν* +1))-*とηλ* (2*κ)- ,ηλ* (*κ* 2*m*)-*を満たすとします。すると、LSGDステップは以下を満たす。*

*.* (4)

*新たな項*-*ηλ*(*f*(*x*)-*f*(*z*))*の存在に注意してください。これは、f*(*z*)≦*f*(*x*)の*とき、つまり、リーダーがxよりも優れているときに、収束を早めるものです。*

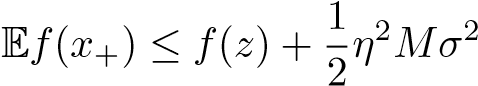
*Ef*(*xk*) - *f*(*x*∗) ≤ *O*(*k*1) と*なります。*

LSGDのレートは、同等の分散型手法のレートと一致しています。Hogwild [31]と

EASGDは、強く凸な目的関数において、以下の収束率を達成しています。多くの分散アルゴリズム（DOWNPOUR[9]を含む）の収束率は公表されていないことに注意してください。

3.1.2コミュニケーション・ ピリオド

実際には、分散したマシン間の通信にはコストがかかります。LSGDアルゴリズムでは、*通信期間τ*を設けており、リーダーは*τ回の反復*ごとにしか更新されないため、各ノードはその間独立して実行することができます。この*τは、*ノード間や時間の経過とともに変化することが許されており、LSGDの非同期型やマルチリーダー型を実現しています。*k*番目の期間中の*j*番目のステップを*xk,j*と書きます。あるk,jに対して*f*(*z)＞f(xk*,*j)*となること*、つまり、*現在の解xk,*jが*最後に選択されたリーダーよりも良くなっていることが起こる可能性があります。この場合、リーダー項*λ*(*x* - *z*)はもはや有益ではなく、代わりに単に*xをz*に向かって引っ張るだけになるかもしれない。この事象が起こるまでに何ステップを踏むかを決定する一般的な方法はない。しかし、*f*(*z*)≧*f*(*x*)であれば、次のことを示すことができる。

*,* (5)

のように、解が陳腐なリーダーよりも*悪く*なることはありません（勾配ノイズまで）。*τ*が無限大になると

LSGDは定理2にあるように*z*よりも定量的に優れた最小化子ofに収束します。これらの事実から、LSGDは元のリーダーが優秀であれば、通信期間が長くても安全に使用できることがわかります。

定理2.*fをm-強凸とし、x*∗*をfの最小化とすると、固定のλ,zに対して*

*.ψの最小化器wは、以下を満たす。*

ここでの理論的な結果とセクション3.1.1では、LSGDアルゴリズムの基本的な2つのケース、つまり、ラウンドごとに通信が行われる「同期」のケースと、通信期間が任意に長い「無限非同期」のケースを取り上げています。未知の期間*τ＞1*の場合、(5)以上の一般的な定量化可能な改善を示すことは困難ですが、(4)、定理2、確率的リーダー選択に関する結果（3.1.3節、7.6節）を組み合わせることで、非同期型LSGDの特定のインスタンスを解析できることに留意しています。

実験では、陳腐化したリーダーの問題を回避するために、別の方法を採用しています。リーダーが良好であることを確認するために、リーダー更新後の最初のステップでのみLSGDステップを実行し、残りの通信期間では標準的なSGDステップを実行します。

3.1.3 確率的なリーダー選択

次に、エラーのあるリーダーを選択した場合の影響を考えます。実際には、深層学習のように、*f*(*x*)を評価するのはコストがかかることが多い。代わりに、値*f*(*xi*)を推定し、最小の推定値を持つ変数として*z*を選択します。形式的には、一様に有界な分散を持つ、*f*(*x*)の不偏な推定量があるとします。各ステップでは，各推定量から1つのサンプル*y*1*,..., ypを抽出する．, yp* が各推定量から抽出され，*z* = {*xi* : *yi* = min≤{*y*1*,...,yp*}}となる．これを√*stochastic leader selectionと呼ぶことに*する．確率的リーダーは，*Ef*(*z*) *f*(*ztrue*) + 4 *pσf* を満たし，*ztrue* は真のリーダーである（補足資料参照）．したがって、ストキャスティック・リーダーによってもたらされる誤差は、最大でも4*ηλ*√*pσf*の相加的な誤差をもたらします。これは*η*2ではなく*ηの*オーダーなので、*λk*も減少しない限り、収束を保証できません。次のような結果が得られます。

Theorem 3.

*しましょう。*

*f*

*が仮定1を満たしているとします。*

e

*g*

(

*x*

)

*は、定理1のようになります。仮にstochastic*

*でのリーダー選択*

e

*f*

(

*x*

)

*持つ*

バール(

e

*f*

(

*x*

))

≤

*σ*

2

*f*

*.もし*

*η,λ*

*は、以下のように固定されています。*

*η*

≤

(2

*M*

(

*ν*

+1))

−

1

*そして*

*ηλ*

≤

(2

*κ*

)

−

1

*,η*

√

*λ*

≤

(

*κ*

√

2

*m*

)

−

1

*とすると*

リムサップ

*k*

→∞

E

*f*

(

*x*

*k*

)

−

*f*

(

*x*

∗

)

≤

1

2

*ηκσ*

2

+

4

*m*

*λ*

√

*pσ*

*f*

*.*

*もし*

*η,λ*

*の割合で減少します。*

*η*

*k*

=Θ(

1

*k*

)

*,λ*

*k*

=Θ(

1

*k*

)

*とすると*

E

*f*

(

*x*

*k*

)

−

*f*

(

*x*

∗

)

≤

*O*

(

1

*k*

)

*.*

通信期間と確率的リーダー選択の精度は、どちらもリーダーの更新コストを削減する方法であり、代用可能である。通信期間が長い場合は、*f*(*xi*)をより高い精度で推定することが、独立して行えるので有効な場合があります。

3.2 非凸の最適化。定常点

前述したように、EASGDには、EASGDの目的関数が、*x* [[1]](#footnote-1)*,...,xp,*xeのどれもが基礎関数*f*の定常点ではないような定常点を持つことがあるという欠点がありますが、LSGDにはこの問題はありません。

定理4*xiが*(x1*,...,xp*)*の中で一意に最小化される点を*Ω*iとすると、Ωiは*(x1*,...,xp*)の中*で一意に最小化される点である。x*∗ = (*w*1*,... ,wp*) ∈Ω*iがLSGD目的関数の定常点であれば、*∇*fi*(*wi*) = 0である*。*

さらに、*通信期間を任意に選択し*た決定論的アルゴリズムLGDでは、以下のような変数*xiが*常に存在することが示されます。

定理5*fが以下に有界であり、M-Lipschitz微分可能であり、LGDのステップサイズが以下のように選択されている**とする。そして、任意の通信期間の選択に対して、xiが無限にリーダーになるようなすべてのiに対して、liminfk* k∇*f*(*xik*)k = 0*が成立する。*

3.3 リーダー選択による検索方向の改善

本節では、LGDが勾配降下法よりも優れた探索方向を得ることができることについて述べる。一般に、LGDのステップが*f*(*x - η*(∇*f*(*x*) + *λ*(*x* - *z*)))≦*f*(*x*-*η*∇*f*(*x*))を満たすかどうかは、*f,x,z,η,λ*の正確な組み合わせに依存し、さらに、*η*の最大許容値がLGDと勾配降下法では異なるため、判断が難しい。代わりに、探索方向の良し悪しを、ニュートン方向とのなす角度で測ります *dN*(*x*)=-(∇2*f*(*x*))-1∇*f*(*x*)ニュートン法は、非特異なヘシアンを持つ局所的な最小化器の周りでは局所的に二次収束し、*η*=1の場合は二次関数に対して1ステップで収束する。したがって、*dNに*近い探索方向が望ましいと考える。*θ*(*u,v*)を*u,v*間の角度とする。*dz* = -(∇*f*(*x*)+*λ*(*x*-*z*)) をリーダー*z*のLGD方向とし、*dG*(*x*) = -∇*f*(*x*)とする。*角度改善集合*は、リーダーの集合 *Iθ*(*x,λ*) = {*z* : *f*(*z*) ≤ *f*(*x),θ*(*dz,dN*(*x*)) ≤ *θ*(*dG*(*x),dN*(*x*))}である。リーダー候補の集合を*E* = {*z* : *f*(*z*)≦*f*(*x*)}とする。我々は、*E*のリーダーの大部分が*Iθ*(*x,λ*)に属することを示すことを目的とする。

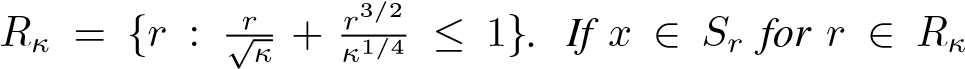
ここでは、条件数*κ*、*dG*(*x*)=-*Ax,dN*(*x*)=-*x*の正定値二次方程式を考える。最初の結果は、*λ*が十分に小さくなると、*E*の少なくとも半分が角度を改善することを示しています。集合の相対的な大きさを測るために、*n*次元の体積Vol(-)を用いる。*E* = {*x* : *xTAx* ≤ 1}で与えられる楕円体*E*は、体積Vol(*E*) = det(*A*)-1*/* [[2]](#footnote-2)Vol(*Sn*)（*Sn*は単位球）を持つ。

定理6*θx*＝*θ*(*dG*(*x),dN*(*x))＞0となるような任意の点をxとし、E＝{*z : *f*(z)≦f(x)}*とする。すると、*lim*λ*→0 Vol( Vol(*E*)2*.*

次に、*λ*が大きい場合について検討する。*dG*(*x),dN*(*x*)の角度が大きい点が存在し、これがLGDによる改善に最も適していることを示す。*r* ≥ 2の場合、*Sr* = {*x* :

.*Sr*はすべての*r* ≥ 2に対して非空であることが示されます。示します。

*x*∈*Sr*に対して、ある範囲の*r*に対して、*Iθ*(*x,λ*)は、*どのようなλの選択に対しても、E*の少なくとも半分であること。

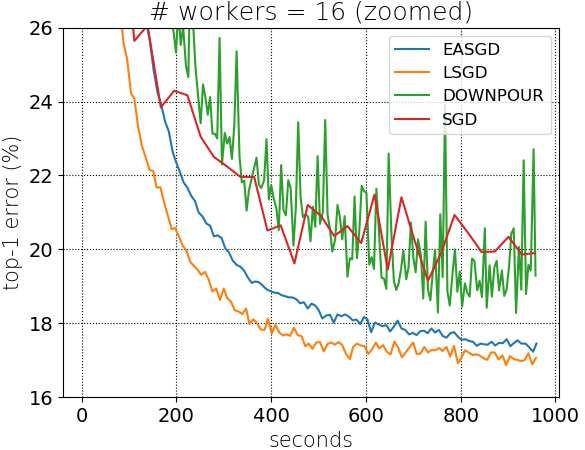
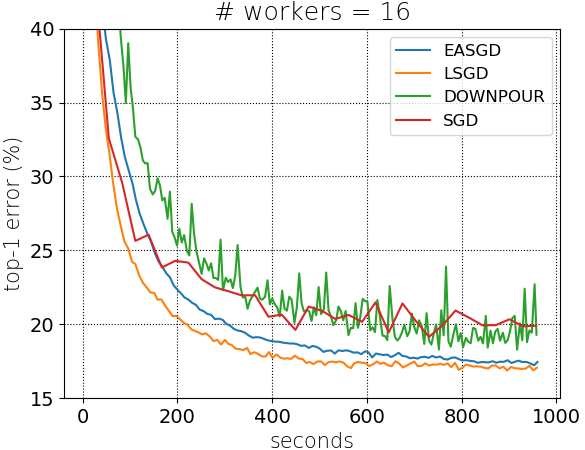
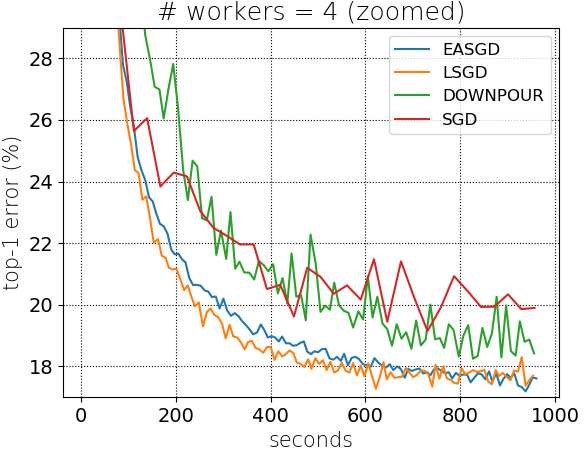
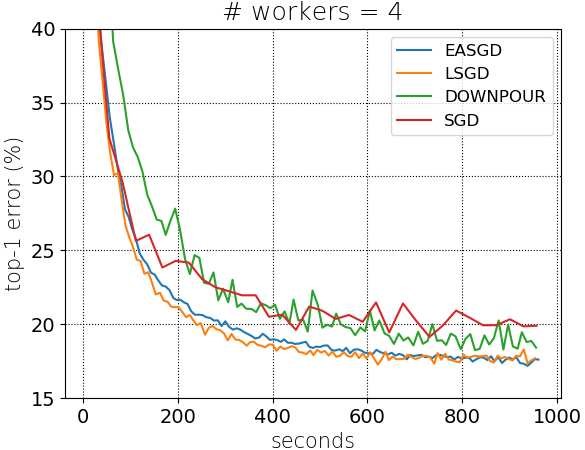
Theorem 7.*とする**と、任意のλ*≧0*に対して*

Vol( Vol(*E).*

なお、定理6および7は、*凸*関数、または目的関数が局所的に凸である局所最小化器の近傍にのみ適用される。非凸のランドスケープでは，ニュートンの方向が鞍点に向かうことがあり[32]，これは望ましくないことです．しかし，定理6と7はこのような状況では適用されないので，これらの結果はLSGDが有害な振る舞いをすることを意味しません．非凸問題では、多くのリーダー候補が*負の曲率*の方向にあり、実際にはサドルポイントから遠ざかるのではないかという直感がありますが、候補の集合が事前に拘束されていないため、これを解析するのは著しく困難です。

# 実験結果

4.1実験の セットアップ

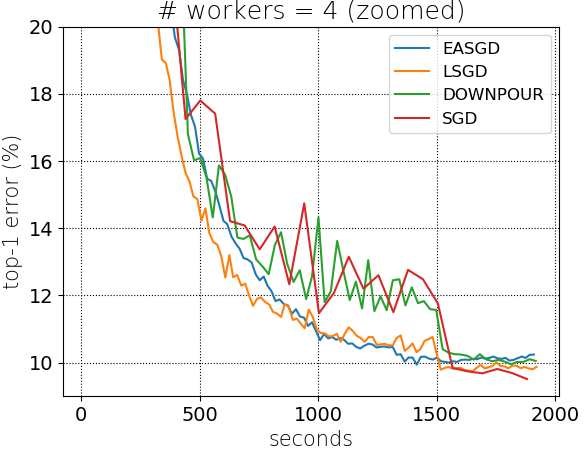
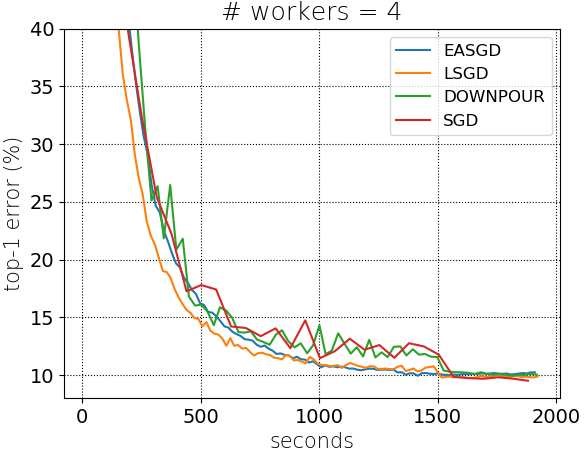


このセクションでは、LSGDの性能を、EASGDやDOWNPOUR（これらの疑似コードは[1]に掲載されています）など、ディープネットワークの並列学習のための最先端の手法や、逐次手法のSGDと比較します。LSGDのコードは、[https://github. com/yunfei-teng/LSGDに掲載されています。](https://github.com/yunfei-teng/LSGD)DOWNPOURでは、安定した収束が得られる典型的な設定であるため、すべての実験で通信周期を1にしています。実験には，CIFAR-10 を使用した．

データセット[33]を3つのベンチマークで検証 図2：CIFAR-10におけるCNN7。中央の可変アーキテクチャーのテストエラー。使用した7層CNN対ウォールクロック時間（左がオリジナルのプロット、右がオリジナルのEASGD紙での拡大図）。テスト損失は補足の図10で報告されている。

(1]のセクション5.1.参照）をCNN7，VGG16[34]，ResNet20[35]と呼び，ImageNet（ILSVRC 2012）のデータセット[36]をResNet50と呼んでいます．

学習時には，過去10回（CIFAR-10）および64回（ImageNet）のデータバッチで計算された学習損失の平均値に基づいて，LSGD法のリーダーを選択します．テスト時には、EASGDとLSGDの中心変数の性能を報告します。LSGDでは、中心変数を



全パラメータの平均 図3：CIFAR-10上のVGG16。センター変数の労働者に対するテストエラー。[*備考*。我々は対壁時計時間であることに注意してほしい（オリジナルのプロットは左、ズームオンはリーダーのパラメータを使って右に引っ張る）。テストの損失は、補足の図12で報告されています。トレーニング時には、平均化されたパラメータをテスト時に報告します。トレーニング時に平均化されたパラメータにワーカーを引っ張ると収束が遅くなる可能性があることが論文で示されており（例：図1）、この問題に対処しています。なお，トレーニング後，作業者が収束して得たパラメータは，ランドスケープの同じ谷間に位置する可能性が高く（[37]参照），その平均値の方が優れた汎化能力を持つと予想されるため（[27, 38]など），テスト時に平均化されたパラメータの結果を報告することにした]．最後に，すべての手法において，減衰係数を10-4に設定した重み減衰を使用する．実験では、4人の作業員（シングルリーダーのLSGD設定）または16人の作業員（4つのグループの作業員を持つマルチリーダーのLSGD設定）を使用します。すべての手法において、同程度の収束率の下で、達成可能な最小のテストエラーにつながる学習率を報告しています（不合理に遅い収束につながる小さな学習率は却下しました）。

イーサネットで相互接続されたGPUノードを使用しています。各GPUノードは4つのGTX 1080 GPUプロセッサを持ち、各ローカルワーカーは1つのGPUプロセッサに対応します。CUDA Toolkit 10.0[[3]](#footnote-3)とNCCL 2を使用しています[[4]](#footnote-4)。分散学習のためにPyTorchをベースにしたソフトウェアパッケージを開発し、リリースする予定です（詳細はセクション9.4で詳しく説明します）。

データ処理とプリフェッチについてはSupplementに記載しています。また、各手法で検討したハイパーパラメータの概要も補足に記載しています。CNN7では学習率を一定にし、VGG16、ResNet20、ResNet50では学習率を落とす（オプティマイザが飽和したときに学習率を10で割る）ことにした。

4.2 実験結果

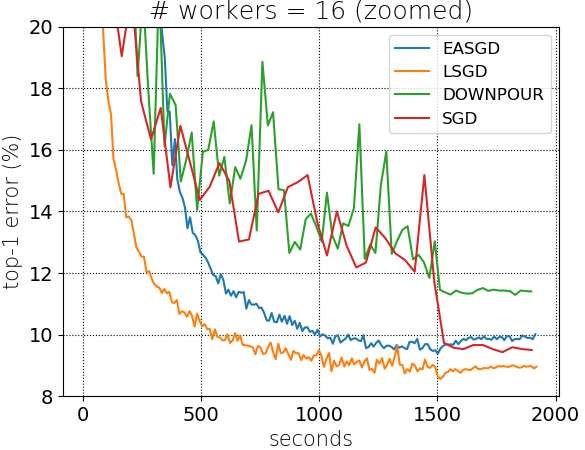
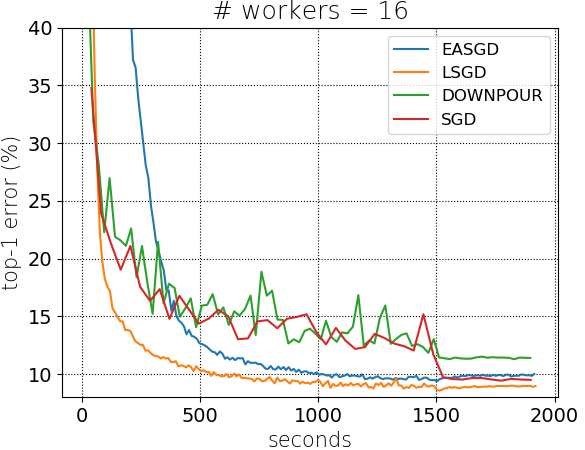
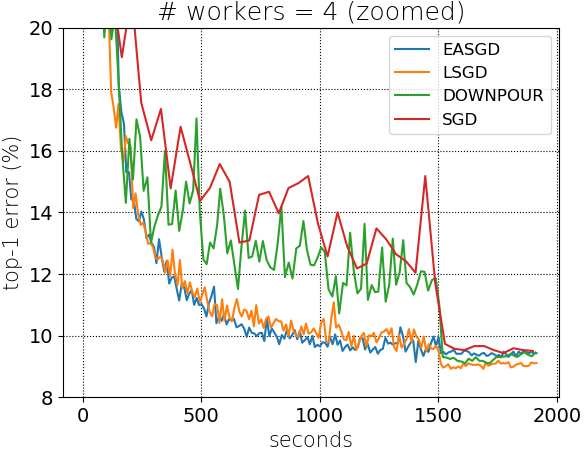
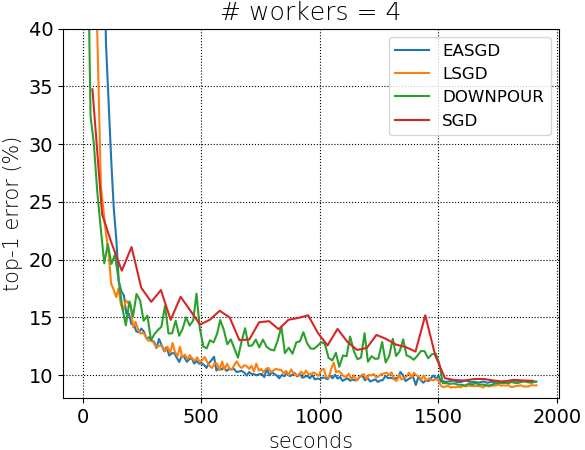
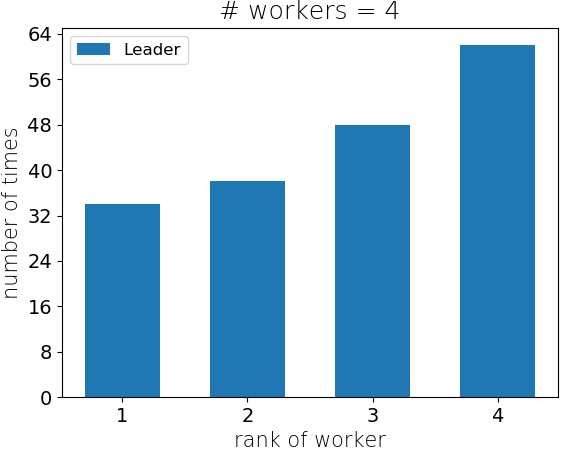
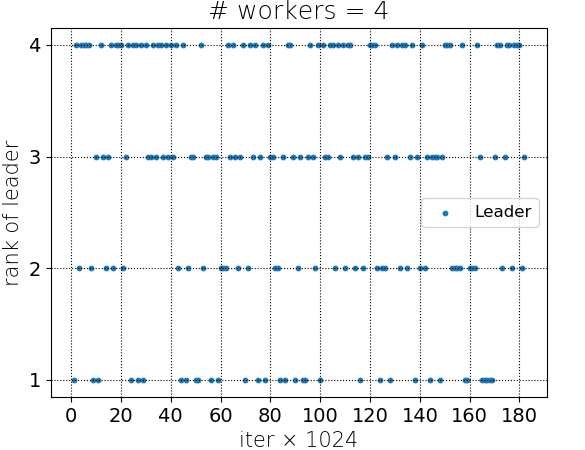


図2では、CIFAR10でCNN7を使って得られた結果を報告しています。EASGDとLSGDを通信周期*τ*=64で実行しました。マルチリーダーのLSGDの場合は*τG* = 128としました。作業員の数は*l* = {4*,*16}とした．我々の手法は，収束速度の点で一貫して競合他社を上回り（16人の場合，EASGDよりも約1*.*5倍速い），16人の場合はより小さな誤差しか得られなかった．

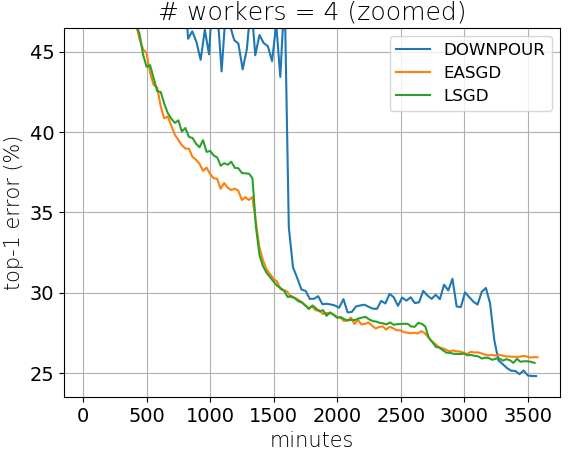
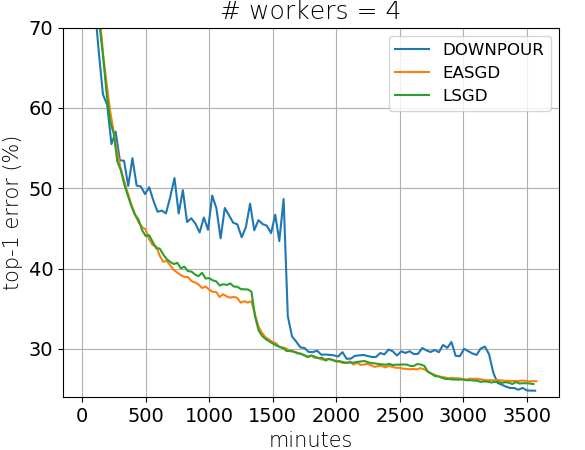
図3では、VGG16とCIFAR-10の結果を示しています。通信周期は64、作業者数は以下の通りです。中央のVari4のテスト誤差。LSGDは壁時計の時間に対してわずかに収束している（左がオリジナルのプロット、右がEASGDよりも高速に拡大して回復している）。テストの損失は補足の図11で報告されています。同時に、収束速度の点でDOWNPOURを大きく上回り、わずかに良い解を得ることができました。

ResNet20とCIFAR-10を用いて、CNN7の場合と同じ通信期間と作業者数の設定で得られた実験結果を図4に示す。作業者数が4人の場合、EASGDと同等の速さで収束しますが、テストエラーはよりよく回復します。この実験では、図5に、CIFAR-10上のResNet20の切り替えパターンを示す。LSGDがリーダーとして認識されていることを示す、リーダーとなったワーカーのアイデンティティ（すなわち順位）と、反復回数（左）と、各ワーカーがリーダーとなった回数（右）を比較している。16人のワーカーでは、EASGDよりも約2倍速く収束し、誤差も大幅に小さくなりました。この実験とCNN7の実験では、LSGD（およびEASGD）は、予想通りDONWPOURやSGDよりも一貫して優れています。



備考1*(1)実験で記録されたリーダー交代のスケジュールは頻繁に交代していること、(2)リーダーポイント自体は最小値から引き離されていないこと、これら2つの事実が相まって、LSGDにおける「引き離し」は有益であることを示唆していると考えます。つまり、ローカルミニマムから引き離されたリーダーではないワーカーが後にリーダーになることで、本来よりもさらに良い解を得ることができる可能性があります。*

最後に，図6にImageNet上で実行したResNet50の実証結果を報告します。作業員の数は4人、通信周期*τ*は64に設定しました。この実験では、我々のアルゴリズムはEASGDと同等の挙動を示しましたが、DOWNPOURよりもはるかに速く収束しました。また、ImageNet上のResNet50では



ImageNet では、SGD は一貫して図 6: ImageNet における ResNet50。中央の変数のテスト誤差は、報告されたすべての手法よりも悪く、壁時計の時間に対しても悪い（左の元のプロットと、拡大された

(SGDによるImageNetの学習 右）。)テストの損失は、補足資料の図13で報告されています。1台のGTX1080 GPUを使って収束するまでには通常1週間ほどかかり、最終的な性能は若干低下します）そのため、SGD曲線は意図的に省略されています（他の手法では2日ほどで収束します）。

# 結論

本論文では、非凸環境における分散最適化のためのLSGDと呼ばれる新しいアルゴリズムを提案する。我々のアプローチは、各反復において、ワーカーを平均ではなく、現在のベストパフォーマーに引き寄せることに依存している。我々は、理論的にも経験的にも、平均値をリーダーに置き換えることを正当化する。また、我々のアルゴリズムについて、収束の証明を含めた徹底的な理論的分析を行う。最後に、我々のアプローチを行列完成問題と深層学習モデルのトレーニングに適用し、これらの学習設定に適していることを実証する。

# 謝辞

WGとDGはNSFの助成金IIS-1838061の一部を受けました。AWは，Darwin CollegeのDavid MacKay Newton research fellowship，The Alan Turing InstituteのEPSRC Grant EP/N510129/1 & TU/B/000074，Leverhulme Trust via the CFIからの支援を受けています．

# リファレンス

1. S.Zhang, A. Choromanska, and Y. LeCun.エラスティックアベレージングSGDによる深層学習．In *NIPS*, 2015.
2. A.Krizhevsky, I. Sutskever, and G. E. Hinton.深層畳み込みニューラルネットワークによるイメージャー分類In *NIPS*.2012.
3. O.Abdel-Hamid, A.-r.Mohamed, H. Jiang, and G. Penn.畳み込みニューラルネットワークの概念を音声認識のためのハイブリッドNN-HMMモデルに適用した。In *ICASSP*, 2012.
4. J.Weston, S. Chopra, and K. Adams.#tagspace:ハッシュタグからのセマンティックエンベッディング。In *EMNLP*, 2014.
5. U.Wickramasinghe and A. Lumsdaine.A survey of methods for collective communication optimization and tuning.*CoRR*, abs/1611.06334, 2016.
6. T.Ben-NunとT. Hoefler.並列・分散深層学習の解明。徹底した並行性分析。*CoRR*, abs/1802.09941, 2018.
7. A.Gholami, A. Azad, P. Jin, K. Keutzer, and A. Buluc.トレーニングニューラルネットワークにおけるモデル、バッチ、ドメインの統合された並列性。*Proceedings of the 30th Syposium on Parallelism in Algorithms and Architectures*, pages 77-86, 2018.
8. L.Bottou.オンライン・アルゴリズムと確率的近似.In *Online Learning and Neural Networks*.Cambridge University Press, 1998.
9. J.Dean, G. Corrado, R. Monga, K. Chen, M. Devin, M. Mao, A. Senior, P. Tucker, K. Yang, Q. V. Le, et al. 大規模分散型ディープネットワーク.In *NIPS*, 2012.
10. X.X.Lian, W.Zhang, C.Zhang, and J.Liu.非同期分散型並列ストキャスティック・グラディエント・ディセンション（非同期分散型並列ストキャスティック・グラディエント・ディセンション）。In *ICML*, 2018.
11. A.Sergeev and M. Del Balso.Horovod: fast and easy distributed deep learning in TensorFlow.*CoRR*, abs/1802.05799, 2018.
12. N.N.S.Keskar, D.Mudigere, J.Nocedal, M.Smelyanskiy, and P.Tak Peter Tang.深層学習のための大規模なバッチトレーニングについて。Generalization gap and sharp minima.In *ICLR*, 2017.
13. S.Jastrze˛bski, Z. Kenton, D. Arpit, N. Ballas, A. Fischer, Y. Bengio, and A. Storkey.Finding flatter minima with sgd.In *ICLR Workshop Track*, 2018.
14. S.S. L. Smith and Q. V. Le.A bayesian perspective on generalization and stochastic gradient descent.In *ICLR*, 2018.
15. S.Ma, R. Bassily, and M. Belkin.The power of interpolation:現代のオーバーパラメトリックな学習におけるsgdの有効性を理解する。In *ICML*, 2018.
16. Y.You, I. Gitman, and B. Ginsburg.Scaling SGD batch size to 32k for imagenet training.In *ICLR*, 2018.
17. B.B. T. Polyak and A. B. Juditsky.平均化による確率的近似の高速化。*SIAM Journal on Control and Optimization*, 30(4):838-855, 1992.
18. X.Li, J. Lu, R. Arora, J. Haupt, H. Liu, Z. Wang, and T. Zhao.Symmetry, saddle points, and global optimization landscape of nonconvex matrix factorization.*IEEE Transactions on Information Theory*, PP:1-1, 03 2019.
19. R.Ge, C. Jin, and Y. Zheng.非凸の低ランク問題におけるスプリアスな局所極小がないこと。A unified geometric analysis.In *ICML*, 2017.
20. J.Sun, Q. Qu, and J. Wright.A geometric analysis of phase retrieval.*Foundations of Computational Mathematics*, 18(5):1131-1198, 2018.
21. J.Sun, Q. Qu, and J. Wright.球面上の完全な辞書復元 I: 概要と幾何学的なイメージ.*IEEE Trans.Information Theory*, 63(2):853-884, 2017.
22. R.Ge, J. D. Lee, and T. Ma.Matrix completion has no spurious local minimum.In *NIPS*, 2016.
23. V.V. Badrinarayanan, B. Mishra, and R. Cipolla.Understanding symmetries in deep networks.*CoRR*, abs/1511.01029, 2015.
24. A.Choromanska, M. Henaff, M. Mathieu, G. Ben Arous, and Y. LeCun.多層ネットワークの損失面について。In *AISTATS*, 2015.
25. S.Liang, R. Sun, Y. Li, and R. Srikant.バイナリ分類のためのニューラルネットワークの損失面の理解。In *ICML*, 2018.
26. K.川口。貧弱なローカルミニマムのない深層学習.In *NIPS*, 2016.
27. P.Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. T. Chayes, L. Sagun, and R. Zecchina.Entropy-SGD: Biasing gradient descent into wide valleys.In *ICLR*, 2017.
28. P.Chaudhari, C. Baldassi, R. Zecchina, S. Soatto, and A. Talwalkar.Parle: stochastic gradient descentの並列化。In *SysML*, 2018.
29. J.Kennedy and R. Eberhart.パーティクルスウォーム最適化。*ICNN*, 1995にて。
30. L.Bottou, F. E. Curtis, and J. Nocedal.大規模機械学習のための最適化手法。*SIAM Review*, 60(2):223-311, 2018.
31. B.レヒト、C.レ、S.ライト、F.ニウ。Hogwild:確率的勾配降下法を並列化するためのロックフリーのアプローチ。in *NIPS*, 2011.
32. Yann Dauphin、Razvan Pascanu、Caglar Gulcehre、Kyunghyun Cho、Surya Ganguli、Yoshua Bengioの各氏。

高次元非凸最適化におけるサドルポイント問題の識別と攻略。で

Z.Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2933-2941.カラン・アソシエイツ社、2014年

1. A.Krizhevsky, V. Nair, and G. Hinton.Cifar-10（カナダ先進研究機関）。
2. K.Simonyan and A. Zisserman.Very deep convolutional networks for large-scale image recognition.In *ICLR*, 2015.
3. K.He, X. Zhang, S. Ren, and J. Sun.画像認識のための深い残差学習．In *CVPR*, 2016.
4. J.Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei.ImageNet:A Large-Scale Hierarchical Image Database.In *CVPR*, 2009.
5. Baldassi, C. et al. Unreasonable effectiveness of learning neural networks:アクセス可能な状態とロバストなアンサンブルから基本的なアルゴリズムのスキームまで。In *PNAS*, 2016.
6. Izmailov, P. et al. Averaging weights leads to wide optima and better generalization. *arXiv:1803.05407*, 2018.
7. C.Szegedy, W.Liu, Y.Jia, P.Sermanet, S.Reed, D.Anguelov, D.Erhan, V.Vanhoucke, and A.Rabinovich.畳み込みでより深くなる。In *CVPR*, 2015.

1. 直感的には、発散していることに注意してください。 [↑](#footnote-ref-1)
2. なお、*λ*1≦*λ*2の場合、*Iθ*(*x,λ*1)⊇*Iθ*(*x,λ*2)となるので、極限はよく定義されています。 [↑](#footnote-ref-2)
3. https://developer.nvidia.com/cuda-zone [↑](#footnote-ref-3)
4. https://developer.nvidia.com/nccl [↑](#footnote-ref-4)