修士論文

高次元悪条件最適化問題に対する
ランダムな次元の制限に基づくCMA-ESとその応用

CMA-ES with Randomized Dimensional Restriction for High Dimensional and Ill-Conditioned Optimization Problems and Its Application

東京大学大学院 情報理工学系研究科
電子情報学専攻
48-176413　清水 洸希

指導教員　豊田 正史 教授

平成31年1月31日　提出

概 要

本論文では，高次元悪条件最適化問題における高速な最適解の探索を目的とした，各反復毎に更新を行う次元をランダムに制限する CMA-ES を提案する．CMA-ES は，進化計算という枠組みの中で進化戦略と呼ばれるアルゴリズムの一種であり，最適化に目的関数の微分を必要としない (derivative-free である) ことから，ブラックボックス最適化に対して有効であることが知られている．高次元悪条件最適化問題についても，分散共分散行列が悪条件性に適応することから有効であるとされてきた．一方で，本論文における実験を通して，100,000 次元に及ぶ Ellipsoid 関数の場合には，分散共分散行列の適応と，全体の最適化を促進する作用のある変数であるステップサイズとの間に乖離が生じることから，最適化に必要な関数の評価回数が良条件な場合と比較して増大してしまうことが明らかとなった．それを受けて，本論文では，各反復毎に各反復毎に更新を行う次元をランダムに制限する CMA-ES を提案し，各反復における見かけの条件数を低減することで，前述の問題を解決した．さらに，一般的に CMA-ES が持つとされる，derivative-free やノイズに対しての頑強性といった性質に着目し，ニューラルネットワークへの適用を試み，ニューラルネットワークにおける勾配消失問題やラベルノイズ問題の解決を図った．

謝辞

進化計算という所属する研究室とは直接的な繋がりのないテーマを選んで研究を進めてきたにも関わらず，修士論文を完成させるまでに至ることができたのは，ひとえに数え切れないほど多くの方々の力添えがあったからであると感じております.

指導教員である豊田正史教授をはじめ，小宮山純平助教，吉永直樹准教授には，分野外であるのにも関わらず，研究の方針から論文やスライドの書き方まで懇切丁寧に指導して頂きました.

喜連川優教授には，世界に誇れるほどに素晴らしい研究室の環境を用意して頂いただけでなく，回数こそ少ないものの直接お会いした折には，時に厳しく時に優しい言葉で研究を励まして頂き，とても励みになりました.

合田和生特任准教授，伊藤雅彦特任准教授，横山大作元助教 (現明治大学理工学部准教授)，安川雅紀特任助教，早水悠登特任助教，梅本和俊特任助教にも軽い雑談から研究の話までしばしばお付き合い頂き，そこでの会話から学びを得ることも多々ありました.

秘書・経理の方々とは，ベランダで園芸を共にできたことが良い思い出です.

研究室の OB・OG 方，先輩方，同期の皆，後輩達は優秀な学生が非常に多く，ミーティングでの発表やコメントは非常に刺激になりました．研究以外でも共にゲームをしたり食事処開拓をしたりと，非常に楽しい時間を過ごすことが出来ました．思うように進捗が出ないときでも継続的に研究室に通うことが出来たのは研究室の学生の皆さんのおかげです.

また，研究室以外の方々にも非常にお世話になりました．冒頭に述べた通り，研究室とは直接的な関連のないテーマではありましたが，学会や研究会では数多くの同分野の研究者方と出会うことができ，そこでの議論から多くの着想を得ることが

出来ました.

新たな出会い以外にも，家族親族やこれまでの友人や仲間達からも多くの助けを得ました.

振り返ってみると，この二年間は人との出会いに非常に恵まれた二年間であり，人と人との繋がりの大切さを今まで以上に学ぶことのできた二年間であったと思います．このように書くと修士課程を修了して研究室から離れるようですが，修了後も博士課程に進学し，この研究室で研究を続けます．博士課程はこれまで以上に険しく長い道のりになるでしょうが，これまでの人生で得た経験・学びを糧に，また新たな出会いや繋がりを糧に乗り越えていきたいと思います.

また 3 年後に謝辞の続きを書けることを願います.

2019 年 1 月 31 日

目 次

図 目 次

表 目 次

第1章　はじめに

1.1　進化計算を用いた最適化

最適化問題は我々人間の生活に非常に深く根付いている．例えば，鉄道のダイヤグラムの作成や，航空機における翼の設計といったものがそれに相当し，これらなくしては実社会は成り立たないといっても過言ではない．

最適化問題とは，与えられた目的関数に対して，それを最小化或いは最大化する変数の組を探索する問題であり，その中でも目的関数が明示的に与えられず，目的関数の導関数や性質が未知であるものをブラックボックス最適化と呼ぶ．前述の例においては，前者は利用者の待ち時間の最小化問題，後者は揚力の最大化問題と換言することができる．ブラックボックス最適化において，考慮しなければならない目的関数の性質として，悪条件性，多峰性，変数間依存性が指摘されている [1] [2]．また，近年を代表する機械学習モデルの一つであるニューラルネットワークを用いた深層学習のような，非常に高次元な最適化問題への適応も急務となっている．

CMA-ES [3] (Covariance Matrix Adaptation Evolution Strategy; 分散共分散行列適応進化戦略) は，ブラックボックス最適化における最も優れたアルゴリズムの一つとして知られている．CMA-ES は，進化計算の枠組みの中でも進化戦略 (ES) と呼ばれるアルゴリズムの一つである．進化計算は，与えられた目的関数に対して，解候補 (個体) をランダムに多数生成し，優れた個体 (目的関数において良い出力を得た個体) を基に新たな個体を生成することで最適化を行うアルゴリズムの枠組みであり，一般に，目的関数の微分を必要としない (derivative-free である) ことや，ノイズに対しての頑強であるといった性質を持つ．ES は，解候補の生成を確率分布を用いて行うもので，確率分布の変数を勾配に基づいて最適化することで，前述の性

質を失うことなく，探索の高速化を可能とした．CMA-ES は，ES における確率分布に多変量正規分布を採用したアルゴリズムである．CMA-ES においては，多変量正規分布の変数である分散共分散行列が，目的関数の悪条件性や変数間の依存性に適応することで，探索効率の向上を可能としている．CMA-ES を改良するための提案は多数行われており，代表的なものとして，時間・空間計算量の低減を目的として，分散共分散行列の制限・近似を行ったもの [4] [5] [6] [7] [8] や，目的関数の多峰性を考慮して探索効率の向上を図ったもの [9] [10] などが挙げられる．以上のような数々の提案により，CMA-ES は，前述の最適化において問題となる目的関数の三つの性質について有効性を高めてきたが，Loshchilov は具体的な原因こそ不明としているものの 100,000 次元の Ellipsoid 関数のような高次元かつ悪条件な目的関数については，従来の CMA-ES では目的関数の値が減少しないことを指摘している [5].

本研究では，実験を通して既存の CMA-ES における前述の問題が，各次元のスケールの違いに適応する分散共分散行列と，次元全体に対してスカラーで定義され全体的な収束速度を高める作用のあるステップサイズが，良条件な場合では近い値を取りながら収束に向かうのに対して，高次元悪条件な場合では値が乖離してしまい，探索を妨げているためであることを確認した．それを受けて，新たに目的関数の次元を各反復毎にランダムに制限し，制限された次元についてのみ更新を行う CMA-ES を提案した．この手法では，次元を制限することで，制限された次元間での見かけ上の条件数を低減できるだけでなく，制限された変数からステップサイズを更新することで，従来ではスカラーでのみ表現可能であったステップサイズを，ベクトルで表現することを可能とし，より局所的な次元のスケールを考慮したステップサイズの更新を行えるようになった．実験では，高次元悪条件なベンチマーク関数を用いた評価を行い，提案手法は解への収束速度や最適解の探索性能において従来の CMA-ES よりもよい結果を得た.

次に，ブラックボックスの実問題への応用として，既存の CMA-ES と提案手法をニューラルネットワークの学習への適用を試みた.

1.2 CMA-ESのニューラルネットワークの学習への適用

ニューラルネットワークは近年において，画像認識や機械翻訳といった分野で，人間に匹敵する [11], あるいは上回る [12] という目覚ましい性能を発揮している．ニューラルネットワークは与えられた入力信号に対して，層ごとに線形写像と非線形写像を繰り返し，出力を求める計算グラフの一種であり，その性能は，層の構造 (ユニット数や結合の有無) と層の持つ重みの値の二つの要素によって定まる．一般にニューラルネットワークの学習は，この重みの値の最適化を指し，手法として確率的勾配降下法 (SGD) が主に用いられる．SGD は非常に単純でありながら，数十万という一般的な最適化問題に対して大規模な次元数を有するニューラルネットワーク の最適化において，優れた性能を発揮している．その一方で，SGD はその性質上，出力層から遠い層が更新されにくくなる勾配消失問題や，目的関数に対して連続性や微分可能性といった制約を与えるといった問題，そして学習データがラベルにノイズを持つ場合に大きく性能を下げるという欠点を有する．今後のニューラルネットワークの産業への応用を考えたとき，実世界の事象に連続性や微分可能性を常に保証することは難しく，また，ラベルノイズについてはデータにおけるアノテーションをクラウドソーシングによって行う場合においてしばしば発生する現象であり，これらは無視することのできない問題であると言える．

本論文では SGD の代わりに CMA-ES を用いた学習を行うことで前述の問題の解決を試みる．1.1 節で述べたように，CMA-ES は，目的関数に対して連続性や微分可能性といった制約を与えない，ノイズに対して頑強性が高いといった性質を持つとされる．また，SGD と異なり，学習において，任意の層を他の層から独立に学習を行うことができる．これらは，SGD と対比したときに非常に有用な特性であると言える．

CMA-ES はこのように，SGD に対して優位な特性を持つ一方で，時間・空間計算量の問題から，これまでにニューラルネットワークの学習に用いられることが極めて少なかった．本研究では，はじめに CMA-ES による学習を GPU を用いた並列化を行うことで，実行時間の問題を軽減し，空間計算量の点では不完全ではあるもの

のニューラルネットワークの学習に対して適用可能であることを示した．次に，ラベルノイズを含むデータセットを用いた学習を行うことで，CMA-ES のノイズへの頑健性の高さを検証した．最後に，CMA-ES と SGD とを組み合わせた学習を行うことで，勾配消失問題の軽減を図った．

1.3 本研究の目的及び貢献

本研究の目的は，高次元悪条件問題において既存の CMA-ES が探索に失敗する原因を突き止め，それを解消する新たなアルゴリズムを提案することにある．また，前述の悪条件性，多峰性，変数間依存性を持ちうるブラックボックス最適化の実問題への応用として，ニューラルネットワークの学習に CMA-ES を適用することで，その特性についてさらなる検討を行う．本研究の貢献を以下にまとめる．

- 高次元悪条件問題における CMA-ES の挙動をベンチマーク関数を用いた実験を通して確認し，探索を妨げている原因についての検討を行った．
- 高次元悪条件問題に対して，ランダムに次元を制限することで各反復内での見かけ上の条件数を低減するアルゴリズムを提案し，ベンチマーク関数を用いてその有効性を確認した．
- ブラックボックス最適化の実問題への応用として，ニューラルネットワークの学習に CMA-ES を適用することで，勾配消失問題やラベルノイズ問題の解決を試み，CMA-ES を用いた学習を行うことによる作用について検討を行った．

1.4 本論文の構成

以下に本論文の構成を記す．

第 2 章 本研究の前提となる，最適化，CMA-ES，ニューラルネットワークについての基礎的な背景知識について説明する．

第 3 章 本研究に関連する，目的関数の性質に対しての CMA-ES の改良に関する研究，進化計算を用いてニューラルネットワークの学習を行う研究，ニューラルネットワークにおけるラベルノイズに関する研究について紹介する．

第 4 章 ベンチマーク関数を用いた予備実験を通して既存の CMA-ES の問題点を探求し，それを踏まえてランダムに次元を制限する CMA-ES を提案し，その性能について実験及び考察を行う．

第 5 章 ブラックボックス最適化の実問題として，CMA-ES をニューラルネットワークの学習に適用し，勾配消失問題やラベルノイズ問題に対しての振る舞いを確認する．

第 6 章 本研究全体のまとめと今後の展望及び課題について総括を行う．

第2章　基礎知識

2.1　最適化問題における関数の性質

最適化問題は，一つの目的関数の最適化を行う単目的最適化と，二つ以上の (相反する) 目的関数の最適化を行う多目的最適化に分けられるが，本論文では単目的最適化について扱う．

単目的最適化では，目的関数 $f(\boldsymbol{x})$ と入力 $\boldsymbol{x} \in \mathbb{R}^d$ が与えられたとき，$f(x)$ を最小化あるいは最大化する $\boldsymbol{x}$ の探索を行う．最適化において，考慮しなければならない目的関数の性質として次の三つが挙げられる [1] [2].

i). 悪条件性:
目的関数が凸二次関数である場合，目的関数を正定値対称行列 $\boldsymbol{H}$ を用いて，$f(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x}$ と表すことができる．このとき目的関数の条件数 $Cond(f(\boldsymbol{x}))$ を $\boldsymbol{H}$ の固有値のうち最小のものと最大のものを用いて，

$$Cond(f(\boldsymbol{x})) = \frac{\lambda_{max}}{\lambda_{min}} \tag{2.1}$$

と定義する．一般にこの条件数が 10^6 を超えるような関数を悪条件関数とする．悪条件関数では，各次元のスケールのしやすさが大きく異なるため，それを考慮したアルゴリズムが求められる．

ii). 多峰性 (大谷構造):
目的関数が凸関数ではなく，最適解以外に局所解を持つ場合に，多峰性関数と呼ぶ．局所解に陥ることなく最適化に到達するためには，変数空間を網羅的に探索する必要があるが，網羅性と探索時間は相反するため，目的関数が大規模な場合には網羅性だけでなく効率性も考慮する必要がある．

iii). 変数間依存性:
目的関数の各次元について依存関係がなく，各次元を独立に最適化を行うことができる場合，変数分離可能という．変数間に依存関係がある場合，一方の変数の値が他方の変数の制約条件となり得るため，依存関係を考慮した最適化をしなければならない．

2.2 進化計算

進化計算は，チャールズ・ダーウィンが 1859 年に著書『種の起源』で提唱した自然選択に基づく進化論に着想を得て考案された最適化アルゴリズムである．進化計算のアルゴリズムは一般に，遺伝的アルゴリズム，遺伝的プログラミング，進化戦略，進化的プログラミングの四つに大別されるが，基本的な設計思想は共通である．本節では，はじめに進化計算の設計思想について説明し，次に単目的最適化において最も優れたアルゴリズムの一つである CMA-ES について述べる．最後に，進化計算のアルゴリズムを確率分布のパラメタ空間上における自然勾配法として一般化した IGO について記す．

2.2.1 進化計算アルゴリズムの設計思想

前述の通り，すべての進化計算アルゴリズムは，自然選択を模して設計されている．具体的には，以下の五つのステップを経て目的関数の最適化を行う．以下，解の候補を個体 (indivisual)，その個体の数を集団数 (population size)，反復回数を表す数値を世代 (generation) と表現する．

i). 初期化:
目的関数の解の候補となる個体をランダムに生成し，初期化を行う．

ii). 評価:
生成された個体を目的関数を用いて評価を行う．

iii). 選択:
評価された個体の順位に基づいて，生存させる個体を決定する．

iv). 生成:
生存した個体を用いて，次の世代の個体群を生成する．

v). 終了条件評価:
予め与えられた終了条件を満たした場合，終了し最も優れた個体を解として出力する．満たさない場合は，2. 以降のステップを繰り返す．

2.2.2 CMA-ES

本項では，CMA-ES のうち最も一般的とされる [13]，$(\mu/\mu_w, \lambda)$-CMA-ES について概要を述べ，次に sep-CMA-ES について述べる．最後に CMA-ES のハイパーパラメタの設定について述べる．

CMA-ES は，一般の進化戦略とは異なり，解の候補を直接保持せず，多変量正規分布 $N(\boldsymbol{m}^{(t)}, \sigma^2\boldsymbol{C}^{(t)})$ により解集団を生成し，それらを用いて平均 $\boldsymbol{m}^{(t)}$ と分散共分散行列 $\boldsymbol{C}^{(t)}$ を最適化することで解の探索を行う．ここで t はステップ数とする．

目的関数 $f(\boldsymbol{x}), \boldsymbol{x} \in \mathbb{R}^d$ を最適化する場合の，具体的なアルゴリズムについて以下に記す．はじめに，変数の初期化を行う．平均 $\boldsymbol{m}^{(0)} \in \mathbb{R}^d$，分散共分散行列 $\boldsymbol{C}^{(0)} \in \mathbb{R}^{d\times d}$，ステップサイズ $\sigma^{(0)} \in R$ をそれぞれ探索領域に応じて決める．また，分散共分散行列とステップサイズのそれぞれの進化パス $\boldsymbol{p}_{\boldsymbol{c}}^{(0)} \in \mathbb{R}^d$ および $\boldsymbol{p}_{\boldsymbol{\sigma}}^{(0)} \in \mathbb{R}^d$ を $\mathbf{0}$ とする．そして，あらかじめ定めた終了条件を満たすまで以下のステップを繰り返す．

[Step 1.] 式 (1) にように，正規分布 $N(\mathbf{0} \in \mathbb{R}^d, \boldsymbol{I} \in \mathbb{R}^{d\times d})$ から，λ 個の個体 $\boldsymbol{z} \in \mathbb{R}^{\lambda\times d}$ を生成する．$\boldsymbol{C}^{(\boldsymbol{t})}$ を平方分解し，$\boldsymbol{B}^{(\boldsymbol{t})}$ とし (式 (2))，$\boldsymbol{z}^{(t)}$ との行列積から $\boldsymbol{y}^{(t)} \in \mathbb{R}^{\lambda\times d}$ を求める (式 (3))．そして，式 (4) のように，$\sigma^{(t)}$ を $\boldsymbol{y}^{(t)}$ の各要素にかけ，平均 $\boldsymbol{m}^{(t)}$ との和から解集団 $\boldsymbol{x}^{(t)} \in \mathbb{R}^{\lambda\times d}$ を生成する．

$$\boldsymbol{z}^{(t)} \sim N(\boldsymbol{0}, \boldsymbol{I}) \tag{2.2}$$

$$\boldsymbol{B}^{(t)} = \sqrt{\boldsymbol{C}^{(t)}} \tag{2.3}$$

$$\boldsymbol{y}^{(t)} = \boldsymbol{z}^{(t)}\boldsymbol{B}^{(t)} \tag{2.4}$$

$$\boldsymbol{x}^{(t)} = \boldsymbol{m}^{(t)} + \sigma^{(t)}\boldsymbol{y}^{(t)} \tag{2.5}$$

[Step 2.] Step 1. で生成した解集団 $\boldsymbol{x}^{(t)}$ の各個体について $f(\boldsymbol{x}_i), (i = 1, ..., \lambda)$ から評価値を計算し，昇順に $\boldsymbol{x}^{(t)}$, $\boldsymbol{y}^{(t)}$, $\boldsymbol{z}^{(t)}$ を個体 (λ) の軸について並び替える．

[Step 3.] 各個体の順位についての重み $\boldsymbol{w} \in \mathbb{R}^{\lambda}$ を用いて，式 (5), (6) にように $\boldsymbol{w}$ と $\boldsymbol{x}^{(t)}$, $\boldsymbol{y}^{(t)}$ の個体の軸についての内積をとり，$\boldsymbol{dy}^{(t)}$, $\boldsymbol{dz}^{(t)} \in \mathbb{R}^d$ を求める．ここで，重み $\boldsymbol{w}$ は，$1 < \mu < \lambda$ として，$w_1 \geq \cdots \geq w_\mu > 0$, $w_{\mu+1}, \cdots, w_\lambda = 0$, $\|\boldsymbol{w}\|_1 = 0$ を満たすものとする．

$$\boldsymbol{dy}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{y}^{(t)} \tag{2.6}$$

$$\boldsymbol{dz}^{(t)} = \boldsymbol{w} \cdot \boldsymbol{z}^{(t)} \tag{2.7}$$

[Step 4.] Step 3. で求めた $\boldsymbol{dy}^{(t)}$, $\boldsymbol{dz}^{(t)}$ を用いて式 (7), (8), (9) のように進化パスを計算する．進化パスは，過去の世代における個体の移動方向を累積することで，最適化の安定性を高める作用がある．ここで，$\mu_w = \frac{1}{\|\boldsymbol{w}\|}$, $\chi_d = \mathbb{E}[\|N(\boldsymbol{0}, \boldsymbol{I})\|] \simeq \sqrt{d}(1 - \frac{1}{4d} + \frac{1}{21d^2}) \in R$ とし，$c_\sigma, c_c \in R$ を各進化パスの学習率とする．

$$h_\sigma^{(t+1)} = \begin{cases} 1 & \|\boldsymbol{p}_{\boldsymbol{\sigma}}^{(t+1)}\| < (1.4 + \frac{2}{n+1})\chi_d \\ 0 & else \end{cases} \tag{2.8}$$

$$\boldsymbol{p}_{\boldsymbol{\sigma}}^{(t+1)} = (1 - c_\sigma)\boldsymbol{p}_{\boldsymbol{\sigma}}^{(t)} + \sqrt{c_\sigma(2 - c_\sigma)\mu_w}\boldsymbol{dz} \tag{2.9}$$

$$\begin{aligned} \boldsymbol{p}_{\boldsymbol{c}}^{(t+1)} &= (1 - c_c)\boldsymbol{p}_{\boldsymbol{c}}^{(t)} \\ &+ h_\sigma^{(t+1)}\sqrt{c_c(2 - c_c)\mu_w}\boldsymbol{dy} \end{aligned} \tag{2.10}$$

[Step 5.] Step 4. で求めた進化パスを用いて式 (10), (11), (12) のように多変量正規分布の変数の更新を行う．ここで，$OP(\cdot) \in \mathbb{R}^{n\times n}$ は入力ベクトルとそれ自身の直積行列とする．また，$\eta_m \in R$ を平均 $\boldsymbol{m}$ の学習率, η_{c_1}, $\eta_{c_\mu} \in R$ をそれぞれ分散共分散行列 $\boldsymbol{C}$ の rank-one update, rank-μ update の学習率とし，$d_\sigma \in R$ をステップサイズの減衰率とする．

$$\boldsymbol{m}^{(t+1)} = \boldsymbol{m}^{(t)} + \eta_m \sigma^{(t)} \boldsymbol{dy}^{(t)} \tag{2.11}$$

$$\sigma^{(t+1)} = \sigma^{(t)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|\boldsymbol{p}_{\boldsymbol{\sigma}}^{(t+1)}\|}{\chi_d} - 1\right)\right) \tag{2.12}$$

$$\begin{aligned}\boldsymbol{C}^{(t+1)} &= \boldsymbol{C}^{(t)} + \eta_{c_1}\left(OP(\boldsymbol{p}_{\boldsymbol{c}}^{(t+1)}) - \boldsymbol{C}^{(t)}\right) \\ &+ \eta_{c_\mu}\boldsymbol{w} \circ \left(OP(\boldsymbol{y}^{(\boldsymbol{t})}) - \boldsymbol{C}^{(t)}\right)\end{aligned} \tag{2.13}$$

以上の Step 1.∼Step 5. を繰り返すことで，平均ベクトル $\boldsymbol{m}$ が解に，ステップサイズ σ が 0 に，分散共分散行列 $\boldsymbol{C}$ が $\mathbf{0}$ に収束することで，多変量正規分布の各次元がディラックのデルタ関数へ近似され，解が一意に求まる．

2.2.2.1 sep-CMA-ES

sep-CMA-ES [4] は目的関数が高次元である場合に，時間・空間計算量の点から CMA-ES の適用が難しいことから，CMA-ES の分散共分散行列を対角成分に限定することで，その問題の解消を図ったアルゴリズムである．sep-CMA-ES では時間計算量・空間計算量ともに O(n) となる．

また，sep-CMA-ES では分散共分散行列の学習率を増加させており，目的関数の入力次元を d として，rank-one update, rank-μ update ともに $(d+2)/3$ を係数にかける．

2.2.2.2 ハイパーパラメタ

CMA-ES はアルゴリズム中のハイパーパラメタがいくつか存在するが，それらは目的関数の入力次元によって定めている．本論文中では，文献 [14] に示されている

推奨値を用いる．以下目的関数の入力次元を d とする．

$$\lambda = 4 + 3\lfloor \ln(d) \rfloor \tag{2.14}$$

$$\mu = \lfloor \frac{\lambda}{2} \rfloor \tag{2.15}$$

$$w_i = \ln(\frac{\lambda+1}{2}) - \ln(i) \tag{2.16}$$

$$\boldsymbol{w} = \frac{[w_1, w_2, \cdots, w_\lambda]}{\|[w_1, w_2, \cdots, w_\lambda]\|} \tag{2.17}$$

$$c_\sigma = \frac{\mu_w + 2}{d + \mu_w + 5} \tag{2.18}$$

$$c_c = \frac{4 + \frac{\mu_w}{d}}{d + 4 + \frac{2\mu_w}{d}} \tag{2.19}$$

$$d_\sigma = 1 + c_\sigma + 2\max\left(0, \sqrt{\frac{\mu_w - 1}{d+1}} - 1\right) \tag{2.20}$$

$$\eta_m = 1 \tag{2.21}$$

$$\eta_{c_1} = \frac{2}{(d+1.3)^2 + \mu_w} \tag{2.22}$$

$$\eta_{c_\mu} = \min\left(1 - \eta_{c_1}, \frac{2(\mu_w - 2 + \frac{1}{\mu_w})}{(d+2)^2 + \mu_w}\right) \tag{2.23}$$

第3章　関連研究

本章では，はじめに，最適化において考慮しなければならない目的関数の性質について，CMA-ES を改良した研究を紹介する．次に，進化計算をニューラルネットワークの学習に用いた研究を，最後に，ニューラルネットワークの学習におけるラベルノイズについての研究を紹介する．

3.1　目的関数の各性質に対しての CMA-ES の改良

第 2 章で述べたように，目的関数の性質は主に，変数間依存性 (Separability), 悪条件性 (Ill-Conditioning), 多峰性・大谷構造 (Multimodality) に分類される．

変数間依存性については，CMA-ES は分散共分散行列行列を用いていることで，依存性を考慮した個体の生成を行うことができるため，有効であるといえる．悪条件性については，現在の CMA-ES ステップサイズ更新手法の主流である Cumulative Step-size Adaptation (CSA) [15] [16] は，ステップサイズを進化パスのマハラノビス距離を用いて更新することから，変数間のスケールの違いに対して比較的ロバストな最適化を可能としている．多峰性・大谷構造については，個体数の増加 [17]，再スタート毎に個体数を変化させる再スタート戦略 [9]，最適化の度合いに応じて世代毎に個体数を変化させる Population Size Adaptation (PSA) [10] が提案されている．

単純に高次元 (learge-scale) な目的関数についても，前述の sep-CMA-ES をはじめとして，LM-CMA [5], VD-CMA [7] 等，数多く研究されており，10000 次元程度の目的関数について時間・空間計算量を削減しつつ変数間の共起を考慮した最適化を可能としている．

本論文で提案する手法では，目的関数が高次元かつ悪条件性が非常に強い場合に

ついて，既存の CSA を踏まえた上で，各世代でランダムに次元を制限しその次元について最適化を行うことで，制限された次元内での見かけの条件数を低減すると共に，ステップサイズをベクトルでの表現を可能とし，より変数間のスケールの違いを考慮した最適化を行うことができる．

3.2 進化計算を用いたニューラルネットワークの学習

ニューラルネットワークの学習に進化計算用いた研究は数多くあるが, 勾配法と進化計算とを組み合わせた研究は非常に少ない．はじめに，進化計算のみを用いた研究を紹介し, 次に勾配法と進化計算とを組み合わせた研究を紹介する．

進化計算による最適化が勾配法と同等の性能を残せることを始めて示したものとして, Morse らの研究 [18] が挙げられる．Morse らは, 一般的な進化計算の手法が個体値の評価を全ての学習データ (full-batch) を用いて行うのに対して, mini-batch と呼ばれる少ないトレーニングデータに区切った評価を行うことで, 時系列データを用いた実験においては, 二乗誤差による評価で SGD をも上回る性能を挙げることに成功した．一方で, 実験に用いたニューラルネットワーク の大きさが中間層のユニット数で高々1000 程度と小さいことは著者らも論文中で触れており, 高次元な場合については検討の余地が残されている．

大規模なニューラルネットワークにおいて, 進化計算を用いた最適化で SGD に並んだものとして, Zhang らの研究 [19] が挙げられる．Zhang らは, OpenAI ES という進化戦略を基にした手法を提案し，正規分布からサンプリングした重みの更新値と，それを加えた場合の目的関数の評価値に応じた報酬から，勾配を計算することで学習を行っている．MNIST の分類問題に対して, 勾配法を上回ることはできなかったものの同等の結果を残し，高次元な場合についても進化計算による手法が一定の成果を発揮することを示している．一方で，勾配法には及ばなかったことから，進化計算による勾配値から勾配法による真の勾配値を推定するアプローチを取っており，進化計算を用いた学習を行うことによる優位性は示せてはいない．

SGD と ES を組み合わせたものとして，Magoulas らの研究 [20] が挙げられる．

Magoulas らは, SGD によって最適化された重みの値を, Differential Evolution Strategy (DES) を用いて組み換えることで, SGD のみを用いた場合よりもよい結果を残した.

本研究では，CMA-ES を勾配法と組み合わせることによって，CMA-ES の適用可能な次元数の上限の問題と，勾配法の勾配消失問題の二点の解決を図っており，

3.3 学習データにおけるラベルノイズ

ラベルノイズとは，学習データについて誤った教師ラベルが付与されたものを指す．損失関数に対して，ラベルノイズに関する正則化を与えることで，性能の向上を図った研究として，Patrini らによる研究 [21] と Ghoshr らによる研究 [22] が挙げられる．

Patrini らは，ノイズの発生確率が既知である場合に，その情報を正則化項として損失関数に与えることで，ラベルノイズ下での性能を向上させた．ノイズの発生確率が未知の場合でも，それを推定する手法も提案しているが，その場合はネットワークに工夫をしなければノイズを過学習してしまうという課題も存在する．Patrini らの手法は損失関数をクロスエントロピー関数に限定しているという課題があったが，Ghoshr らは平均二乗誤差や平均絶対誤差に対してもラベルノイズ化での最適化を可能とした．しかし，損失関数の制約は大きく，マルチラベル問題のように一つの入力データに対して複数の正解ラベルが付与されているような場合には適用することができない.

Goldberger ら [23] は，ニューラルネットワークの構造において，ラベルノイズに対する適応層を追加することで性能の向上を図ったが，層が増えることにより学習時間が増大するという欠点も存在する.

以上のように，ラベルノイズに対しての既存の取り組みは，いずれも損失関数や層の構造に対してのアプローチであり，SGD や ADAM といった学習アルゴリズムへの提案は知りうる限り見られない.

本研究では，SGD や ADAM のような勾配法をベースとしたアルゴリズムではな

く，一般的にノイズに対して頑健であるとされる進化計算の手法である CMA-ES を用いることで，既存の研究とは異なる視点からの取り組みを行っている．また，アルゴリズムによるアプローチであるため，既存手法と組み合わせることも可能である．

実験では，CMA-ES による最適化の頑強性を明確に検証するため，これらの既存研究との組み合わせは行わずに実験を行った．

第4章　高次元悪条件最適化問題

悪条件関数を効率的に最適化を行うためには，関数の各次元のスケールの違いを考慮する必要がある．CMA-ES における最適化では，分散共分散行列の各次元が目的関数の各次元のスケールに適応することから，悪条件関数の最適化について優れるとされる．一方で，100,000 次元の Ellipsoid 関数のような高次元悪条件関数の場合には，10,000 回の関数の評価を行なっても目的関数の値が下がらない現象が報告されている．

本章では，4.1 節において，Ellipsoid 関数と Star 型 Rosenbrock 関数の二つの高次元悪条件関数を用い，従来の CMA-ES における前述の現象についての原因を検討する．そして 4.2 節において，高次元悪条件関数に対して有効な，ランダムな次元の制限に基づく CMA-ES の提案し，その性能について実験及び考察を行う．

4.1　予備実験: 高次元悪条件問題に対する CMA-ES の挙動

本節では，高次元かつ悪条件な目的関数における CMA-ES の挙動を確認する．本論文中で用いるベンチマーク関数を表 4.1 に示す.

4.1.1　Ellipsoid 関数を用いた実験

はじめに，sep-CMA-ES を用いて，Loshchilov による実験 [5] の再現を行った．目的関数には 100000 次元の Ellipsoid 関数を用い，初期値を，$\boldsymbol{m}^{(0)} = [-5, 5]^{100000}$, $\boldsymbol{C}^{(0)} = \boldsymbol{I}$, $\sigma^{(0)} = 1.0$, $\lambda = 4 + 3\lfloor \ln(d) \rfloor$ と設定した．比較として，同じ初期値を用い

表 4.1: Benchmark functions

Function	Definition
Sphere	$\sum_{i=1}^{n} x_i^2$
Ellipsoid	$\sum_{i=1}^{n} (1000^{\frac{i-1}{n-1}} x_i)^2$
Rosenbrock (Star)	$\sum_{i=2}^{n} (100(x_1 - x_i^2)^2 + (1 - x_i)^2)$

て，100,000 次元の Sphere 関数に適用した．目的関数の目標値を 10^{-10}，最大評価回数を $\lambda * 10^7$ と設定した．また，初期値については，以降の全ての実験で同一の設定を用いた．

評価回数 (Func evals) に対する目的関数の値 (Objective)，ステップサイズ (Sigma)，分散 (Covariance) の変化を，Sphere 関数については図 4.1 に，Ellipsoid 関数については図 4.2 に示した．目標値には Sphere 関数では約 $1.5 * 10^7$ 回，Ellipsoid 関数では約 $6.5 * 10^7$ 回の評価回数で到達した．目的関数の値の減少はともに約 $1.0 * 10^6$ 回付近から始まっており，Ellipsoid 関数の方が評価回数あたりの最適化速度が遅いことがわかる．また，Sphere 関数の場合では，ステップサイズと分散共分散行列が共に単調に減少しているのに対し，Ellipsoid 関数では，分散共分散行列の対角成分の平均が上昇している一方でステップサイズは下がり続けていることが読み取れる．この乖離が Sphere 関数の場合と比較して Ellipsoid 関数の最適化速度を下げていると推測され，この乖離を抑制することでこの差を軽減することができると考えられる．

4.1.2 Star 型 Rosenbrock 関数を用いた実験

本節では，前節の推測を確かめるために，10,000 次元の Star 型 Rosenbrock 関数を用いた実験を行う．Ellipsoid 関数が変数間で 10^6 ずつ徐々にスケールが変化していくのに対して，Star 型 Rosenbrock 関数では，1 つ目の変数とそれ以降の変数との間にスケールの差が生じるため，高次元になるほどより悪条件になると考えられる．

前節と同様に結果を図 4.3 に示した．目標値には到達せず，最大評価回数時の目的関数の値は約 2252.2 となった．ステップサイズに注目すると，$1.0 * 10^6$ 回の評価

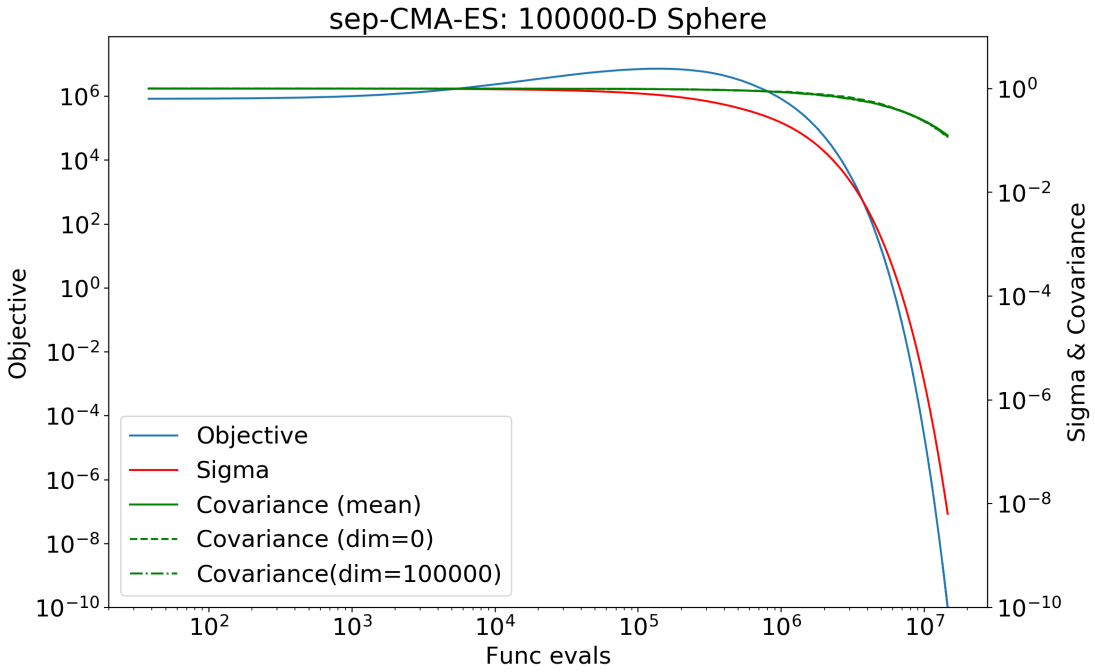

図 4.1: 100,000 次元の Sphere 関数の最適化 (sep-CMA-ES)

回数付近で，1 つ目の変数の分散と共に収束に向かってしまっている．その後，1 つ目の変数の分散も上昇に転じ，全ての次元の分散の平均が大きく上昇しているのに対して，ステップサイズは収束してしまい，最適化が緩やかになっていることがわかる．Star 型 Rosenbrock 関数は 1 つ目の変数の影響が非常に大きく，1 つ目の変数の収束がステップサイズに大きな影響を及ぼしたと考えられる．

4.2 提案手法: ランダムな次元の制限に基づく CMA-ES

本章では，予備実験の結果を受け，ステップサイズの大域的な収束性を維持しつつ，分散共分散行列の局所的な収束を考慮した変数の更新を行う CMA-ES のアルゴ

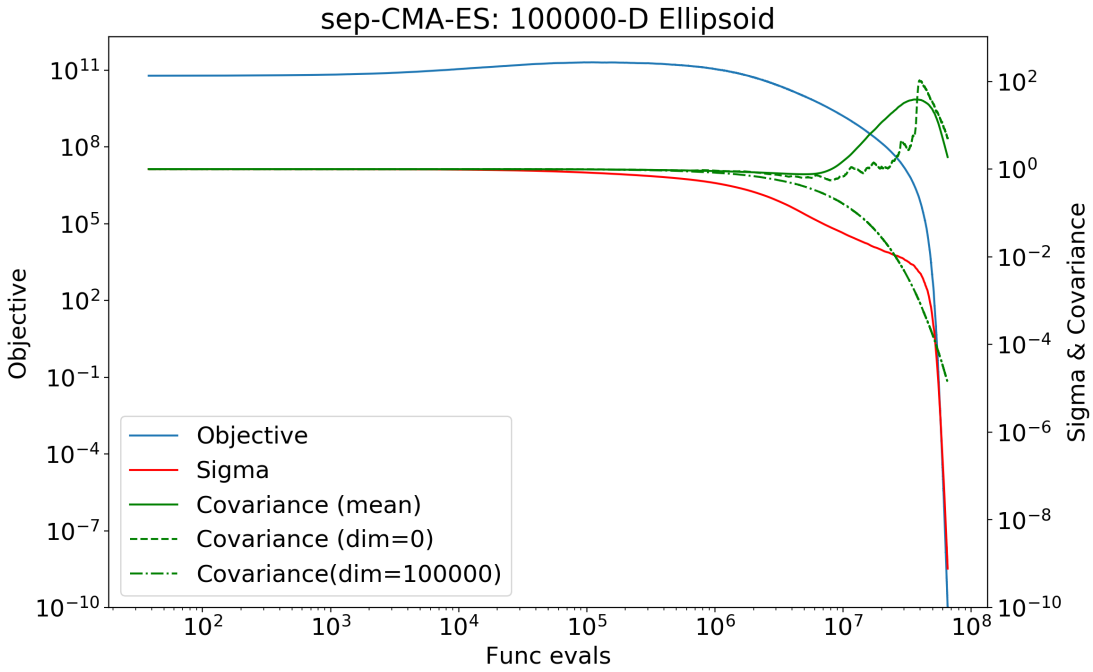

図 4.2: 100,000 次元の Ellipsoid 関数の最適化 (sep-CMA-ES)

リズムについて提案を行う.

4.2.1 更新次元のランダムな制限

本論文では，各世代で更新する次元をランダムに制限する CMA-ES を提案する．具体的には，目的関数の入力次元をインデックス化し，そのインデックスの要素をシャッフルする．そして，各ステップであらかじめ設定した制限次元をインデックスから取り出す．インデックスの最後の要素まで参照を終えたら，再びインデックスをシャッフルする．この手法は CMA-ES と sep-CMA-ES の双方に適用でき，アルゴリズムの変更点は同様なため，CMA-ES への適用について以下の項で述べる．

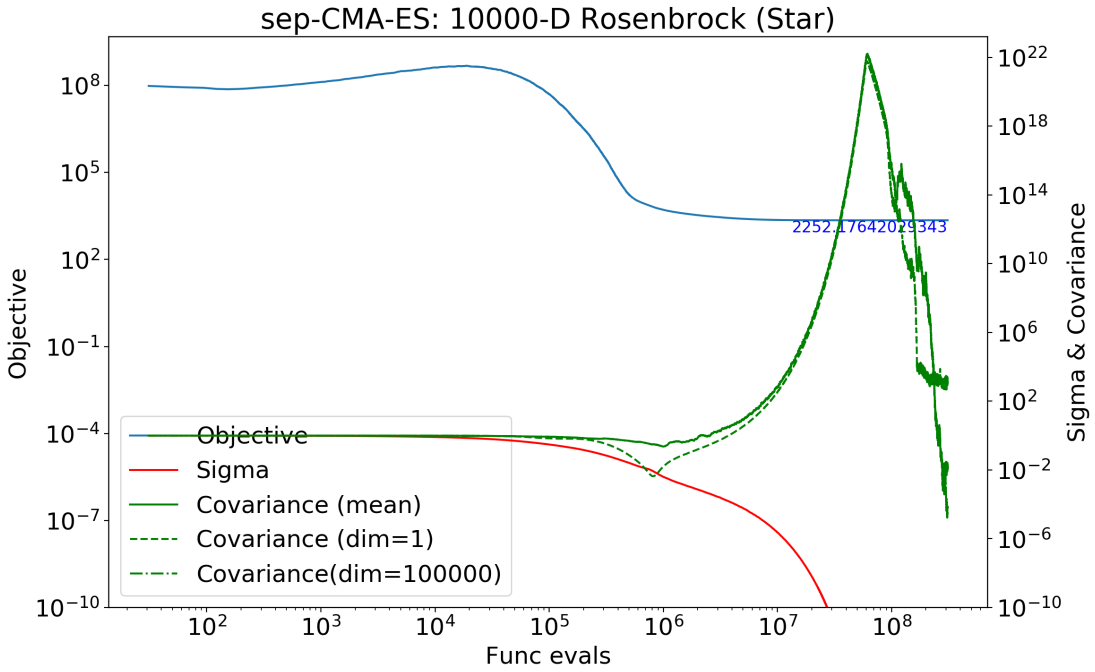

図 4.3: 10,000 次元の Star 型 Rosenbrock 関数の最適化 (sep-CMA-ES)

4.2.1.1 CMA-ES への適用

目的関数を $f(\boldsymbol{x}), \boldsymbol{x} \in \mathbb{R}^d$ とする．はじめに，通常の CMA-ES と同様に，平均 $\boldsymbol{m}^{(0)} \in \mathbb{R}^d$，分散共分散行列 $\boldsymbol{C}^{(0)} \in \mathbb{R}^{d\times d}$，ステップサイズ $\boldsymbol{\sigma}^{(0)} \in R^n$ をそれぞれ探索領域に応じて決め，分散共分散行列とステップサイズのそれぞれの進化パス $\boldsymbol{p}_{\boldsymbol{c}}^{(0)} \in \mathbb{R}^d$ および $\boldsymbol{p}_{\boldsymbol{\sigma}}^{(0)} \in \mathbb{R}^d$ を $\mathbf{0}$ と初期化する．ステップサイズは，通常の CMA-ES ではスカラーで定義するが，提案手法ではベクトルで定義する．また，目的関数の入力次元長のインデックスベクトル $\boldsymbol{\iota} \in \mathbb{R}^d$ を新たに導入し，あらかじめ要素をシャッフルし，インデックスの参照を終えた要素の終端地点 $e \in R$ を 0 と初期化する．そして，あらかじめ定めた終了条件を満たすまで以下のステップを繰り返す．

[Step 1.] 予め設定された，各世代で制限する次元数を $s \in R$ とする．また，この s は世代ごとに異なる値をとることも可能であるが，本論文中では固定値とする．$\boldsymbol{\iota}$

から，前世代での終了地点 e から $e+s$ までを取り出し，当世代でのインデックスベクトル $\boldsymbol{\iota'} \in R^s$ を生成する．ただし，$e+s>d$ の場合，e から d までのベクトル $\boldsymbol{\iota'} \in R^{d-e}$ とし，$\boldsymbol{\iota}$ の要素を全てシャッフルし，$e=0$ とする．

[Step 2.] 生成した当世代でのインデックスベクトル $\boldsymbol{\iota'}$ に基づいて，各変数から部分ベクトルおよび部分行列を生成する．具体的には，平均 $\boldsymbol{m'} \in \mathbb{R}^s$，分散共分散行列 $\boldsymbol{C'} \in \mathbb{R}^{s\times s}$，ステップサイズ $\boldsymbol{\sigma'} \in \mathbb{R}^s$，進化パス $\boldsymbol{p'_c} \in \mathbb{R}^s$, $\boldsymbol{p'_\sigma} \in \mathbb{R}^s$ となる．分散共分散行列については，指定されたインデックスの対角成分およびそれらの共起成分からなる対称行列を抽出する．

[Step 3.] 正規分布 $N(\boldsymbol{0} \in \mathbb{R}^s, \boldsymbol{I} \in \mathbb{R}^{s\times s})$ から，λ 個の個体 $\boldsymbol{z}^{(t)} \in \mathbb{R}^{\lambda\times s}$ を生成する．$\boldsymbol{C'}$ を平方分解し，$\boldsymbol{B'}$ とし，$\boldsymbol{z}^{(t)}$ との行列積から $\boldsymbol{y}^{(t)} \in \mathbb{R}^{\lambda\times s}$ を求める．そして，$\boldsymbol{\sigma'}$ を $\boldsymbol{y}^{(t)}$ の各要素にかけ，平均 $\boldsymbol{m}^{(t)}$ の $\boldsymbol{\iota'}$ の要素との和から解集団 $\boldsymbol{x}^{(t)} \in \mathbb{R}^{\lambda\times d}$ を生成する．このとき，$\boldsymbol{x}^{(t)}$ は通常の CMA-ES と同じ次元の行列となるが，$\boldsymbol{\iota'}$ 以外のインデックスの次元については，各個体全て同じ値 ($\boldsymbol{m}^{(t)}$ の値) を取る．

[Step 4.] 通常の CMA-ES と同様に，目的関数を用いて各個体を評価・順位づけし，昇順に並び替える．また，同様に重み $\boldsymbol{w} \in \mathbb{R}^\lambda$ との個体の軸 (λ の軸) についての内積 $\boldsymbol{dy}^{(t)}$, $\boldsymbol{dz}^{(t)} \in \mathbb{R}^s$ を求める．

[Step 5.] Step 4. で求めた $\boldsymbol{dy}^{(t)}$, $\boldsymbol{dz}^{(t)}$ をもとに，進化パスを更新する．このとき，$\boldsymbol{\iota'}$ で指定された要素のみ更新を行い，その他の要素については元の値を維持する．

[Step 6.] 進化パスをもとに平均 $\boldsymbol{m}^{(t)}$，分散共分散行列 $\boldsymbol{C}^{(t)}$，ステップサイズ $\boldsymbol{\sigma}^{(t)}$ の更新を行う．ここでも，Step 4. と同様に $\boldsymbol{\iota'}$ で指定された要素のみ更新を行い，その他の要素については元の値を維持する．

以上の Step 1.∼Step 6. を通常の CMA-ES と同様に終了条件を満たすまで繰り返す．sep-CMA-ES についても分散共分散行列の対角成分から $\boldsymbol{\iota'}$ に対応する次元を抽出することで CMA-ES の場合と同様に適用することができる．

4.2.1.2 ハイパーパラメタ

提案手法ではハイパーパラメタについて，通常の CMA-ES と同様の式を用いる．ただし，ハイパーパラメタの引数となる次元の値については，目的関数の入力次元数 d ではなく，各世代ごとに制限された次元数 s を用いる．また，sep-CMA-ES への適用の場合は，sep-CMA-ES と同様に分散共分散行列の rank-one update, rank-μ update 双方の学習率に $(s+2)/3$ の係数をかける．

4.2.2 手法の特徴

進化計算において，最適化を行う変数を制限する手法は，Zhang らの OpenAI ES [19] でも用いられており，目的関数が高次元である場合に，集団サイズを維持した状態で更新する次元を減らすことができる．

関数の各性質に対する有効性について，関数が悪条件である場合には，変数を制限することで，制限された次元内での条件数の期待値が軽減されることが期待される．一方で，変数間依存性がある関数については，依存関係のある次元が同じ次元の組に制限される必要性があると考えられ，依存関係が強い場合には性能の悪化が生じることが考えられる．

次元の制限の方法について，次元のインデックスを最後まで参照した際に，インデックスベクトルの要素をシャッフルすることで，制限された次元にふくまれる次元の組が毎回変わるように設計を行った．これは Rosenbrock 関数のように次元間に依存性がある場合に有効であると考えられる．また，インデックスベクトルを先頭から順番に参照していくことで，全ての次元が均一に制限される．これは，制限される回数にばらつきが発生した場合に，制限されなかった次元の共起成分が維持され続け，世代を重ねるごとに対角成分に対して大きな値をとって最適化を妨げることを防ぐためである．

また，ステップサイズをベクトルに変更し，制限された次元を用いて更新を行なっている点が，従来の CMA-ES との最大の相違点である．従来の CMA-ES ではステップサイズを全ての次元の進化パスのマハラノビス距離を用いて更新していたため，

大域的な収束を早める一方で，目的関数の各次元のスケールが大きく異なっていた場合に，分散共分散行列の局所的な収束との乖離が大きいという問題が予備実験から明らかになった．提案手法では，制限した次元の進化パスのマハラノビス距離を用いることで，大域的な収束を維持しつつ，分散共分散行列の局所的な収束にも対応できると考えられる．

最後に，時間・空間計算量について考える．空間計算量については元の分散共分散行列の値を保持するため $O(n^2)$ (sep-CMA-ES の場合は $O(n)$) で変わらない．一方で，時間計算量については，分散共分散行列の固有値分解について，制限した次元のオーダーで行うことができる．

4.3 評価実験と考察

本章では，提案手法の高次元悪条件関数における性能を確認する．悪条件な目的関数として，Ellipsoid 関数および Star 型の Rosenbrock 関数を用いた．一般的によく知られる Chain 型の Rosenbrock 関数は隣り合う変数間の依存性が強い関数として知られているが，Star 型の場合は，1 つ目の変数とそれ以降の変数との間に強い依存性を持つ [24] ことから，1 つ目とそれ以降の変数について依存の度合いが異なり，悪条件な関数とみなすことができる．実装には cupy を用い，GPU による並列計算を行った．本論文で実験に用いた計算機環境を表 4.2 に示す．

4.3.1 高次元悪条件関数を用いた実験

本節では，提案手法, sep 型の提案手法, sep-CMA-ES, VD-CMA, L-BFGS 法の性能の比較を行う．L-BFGS 法は勾配法をベースにした最適化アルゴリズムの一つであり目的関数の導関数を必要とする．今回の実験では，目的関数の各次元についての一次偏導関数を既知の情報として与え，一回の反復あたりの評価回数を，目的関数の評価回数に各次元の導関数の評価回数を加えた．本節の実験では，提案手法の制限次元を 100 次元と設定した．

4.3.1.1 最適化性能の比較

はじめに，予備実験と同様の設定と目的関数を用いて実験を行った．

各関数について，評価回数 (Func evals) に対する目的関数の値 (Objective)，ステップサイズ (Sigma)，分散 (Covariance) の変化を，Sphere 関数については図 4.4 に，Ellipsoid 関数については図 4.5 に，Star 型 Rosenbrock 関数については図 4.6 に示す．

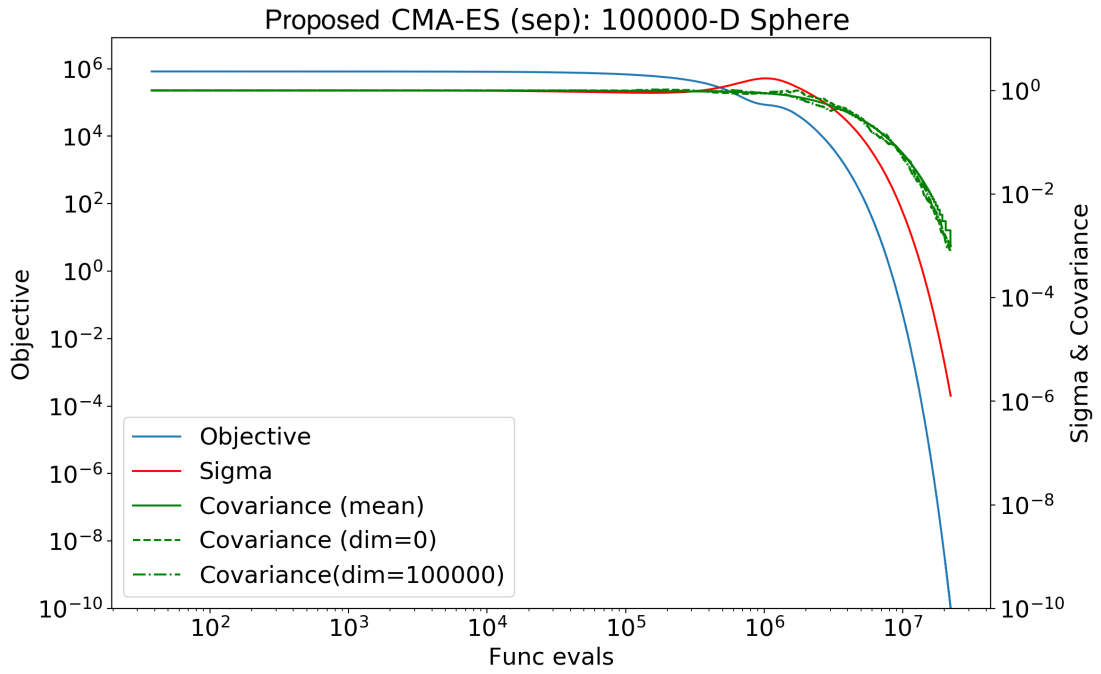

図 4.4: 100,000 次元の Sphere 関数の最適化 (提案手法)

Sphere 関数では，約 $2.5 * 10^7$ 解の評価回数で目標値へと到達し，sep-CMA-ES より多くの評価回数が必要となった．Ellipsoid 関数では，ステップサイズと分散との乖離が小さくなった結果，約 $4.5 * 10^7$ 回で目標値に到達し，sep-CMA-ES よりも約 $2.0 * 10^7$ 回程度評価回数が少なくなった．Star 型の Rosenbrock 関数では，目標値には到達しなかったが，最大評価回数時に目的関数の値が約 59.76 となり，sep-CMA-ES

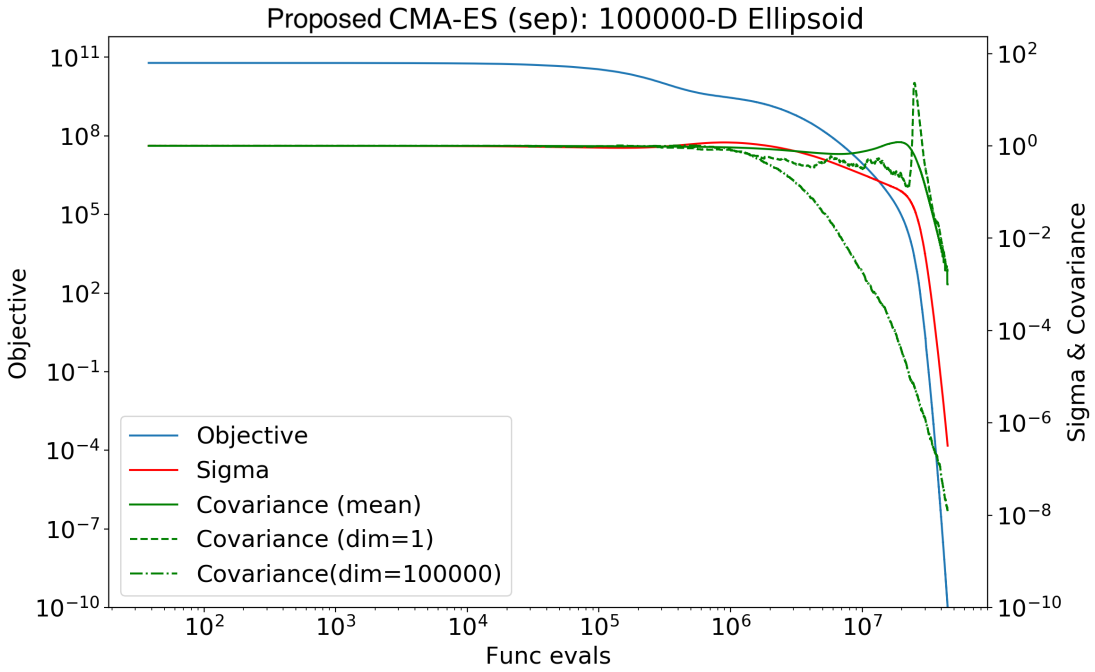

図 4.5: 100,000 次元の Ellipsoid 関数の最適化 (提案手法)

の 1/40 程度まで最適化することに成功した．こちらの場合でも，ステップサイズと分散の乖離が，sep-CMA-ES よりも小さくなったことが原因であると考えられる．

また，各関数について，評価回数あたりの目的関数値の推移を，Sphere 関数については図 4.7, Ellipsoid 関数については図 4.8, Star 型 Rosenbrock 関数については図 4.9 に示した．

Sphere 関数では，L-BFGS 法が最も少ない評価回数で目標値に達し，良条件・単峰性・変数名のない関数については，勾配法が優れていることが伺える．

Ellipsoid 関数では，提案手法が最も少ない評価回数で目標値に達した．sep-CMA-ES および VD-CMA も L-BFGS 法を上回ったことから，分散共分散行列とステップサイズを用いることで，目的関数の導関数を用いるよりも，悪条件に適応した最適化が可能となっていることが伺える．

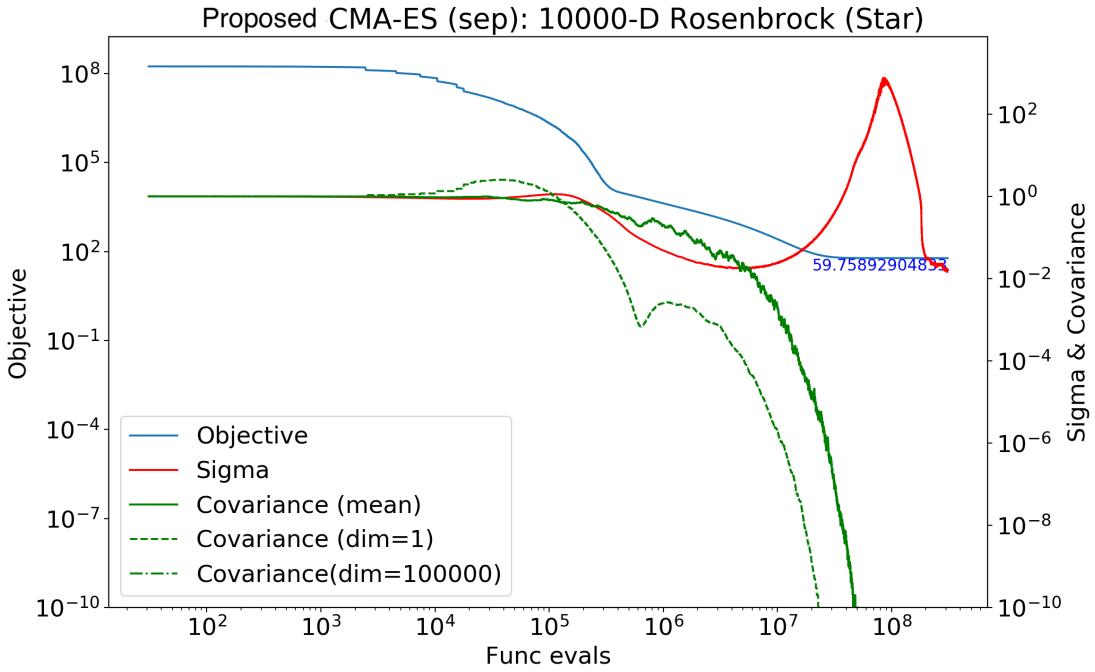

図 4.6: 10,000 次元の Star 型 Rosenbrock 関数の最適化 (提案手法)

Star 型 Rosenbrock 関数では，提案手法が最も良い解に到達した．分散共分散行列の対角成分のみを使用した提案手法においても VD-CMA を上回ったが，これについては，Star 型 Rosenbrock 関数は高次元になるほど，分散共分散行列中の依存関係のある変数が占める割合が減り変数間依存性が弱くなる一方で，悪条件さは増すためであると考えられる．

4.3.1.2 計算時間の比較

各関数について，時間に対する目的関数の値および評価回数の変化を図 4.10, 図 4.11 に示す．目的関数の値の変化は前節での結果とほぼ同様であった．一方で，評価回数の変化は，更新する次元数を制限しているにも関わらず，sep-CMA-ES を下回った．これは，各変数での更新する次元の抽出およびインデックスの要素のシャッ

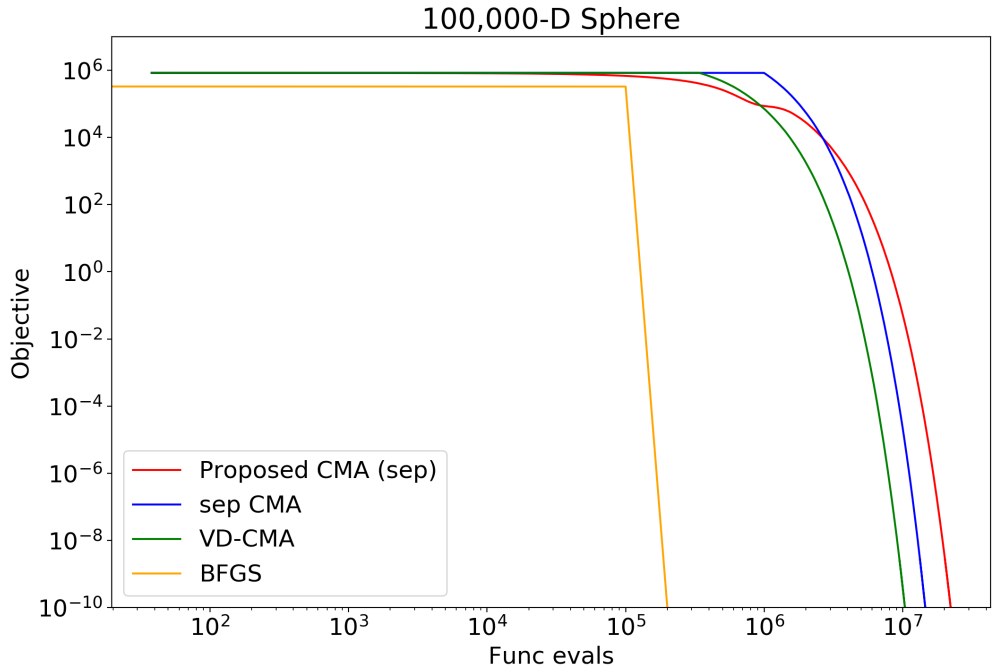

図 4.7: 100,000 次元の Sphere 関数の最適化における各種法の評価回数あたりの目的関数値の比較

フルに起因するものであると考えられる．

4.3.2 次元数の変化についての比較

本節では，目的関数の条件数を固定し次元数を変化させた場合について，提案手法と sep-CMA-ES の性能の比較を行う．目的関数には，Ellipsoid 関数を用い，次元を 100 から 10,000 まで変化させ，各次元について目標値に達するまでの評価回数を計測した．結果を図 4.12 に示した．次元数が，100 と 1,000 の間で評価回数が逆転が生じた．また，次元数が変化しても二つの手法間での評価回数の差に大きな差はみられない．このことから，次元数は提案手法における制限された次元内での条件数の期待値に影響は見られるものの，主要な要因ではないと考えられる．

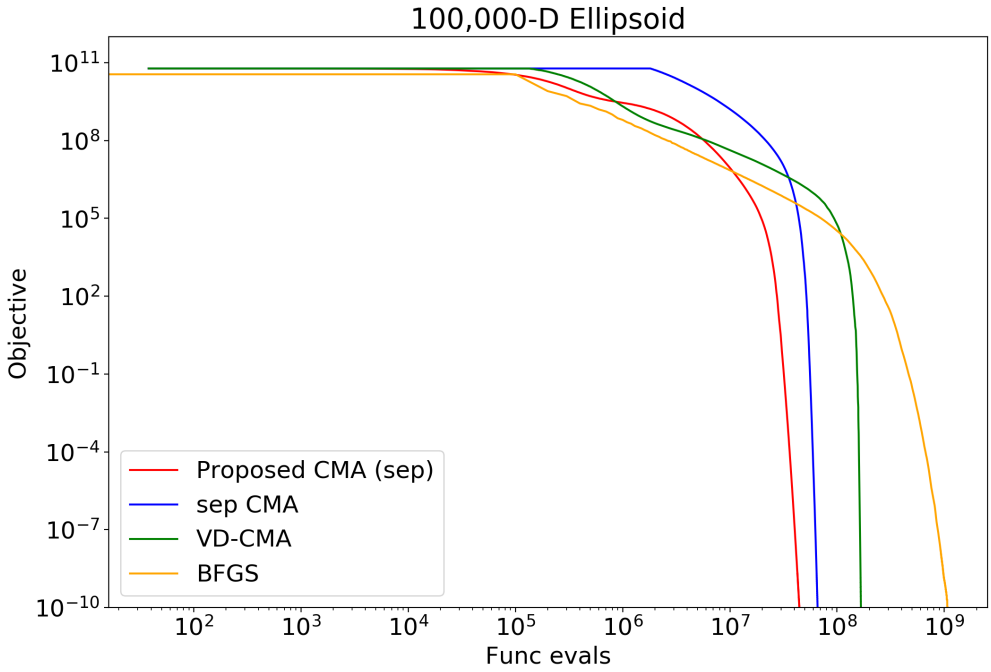

図 4.8: 100,000 次元の Ellipsoid 関数の最適化における各種法の評価回数あたりの目的関数値の比較

4.3.3 条件数の変化についての比較

本節では，目的関数の次元数を固定し条件数を変化させた場合について，提案手法と sep-CMA-ES の性能の比較を行う．目的関数には，条件数を操作するために Ellipsoid 関数の係数を修正した関数を用いた．100 次元の修正 Ellipsoid 関数では，条件数を 10^6 から 10^8 まで変化させ，各条件数について目標値に達するまでの評価回数を記録した．また，1,000 次元の場合では，条件数を 10^3 から 10^6 まで変化させ，同様に結果を記録した．

各条件数に対しての目標値到達までの評価回数の推移を，100 次元の場合について図 4.13 に，1,000 次元の場合について図 4.14 に示した．いずれの次元の場合でも，条件数が大きくなるにつれて提案手法は sep-CMA-ES に比べて評価回数が少なくな

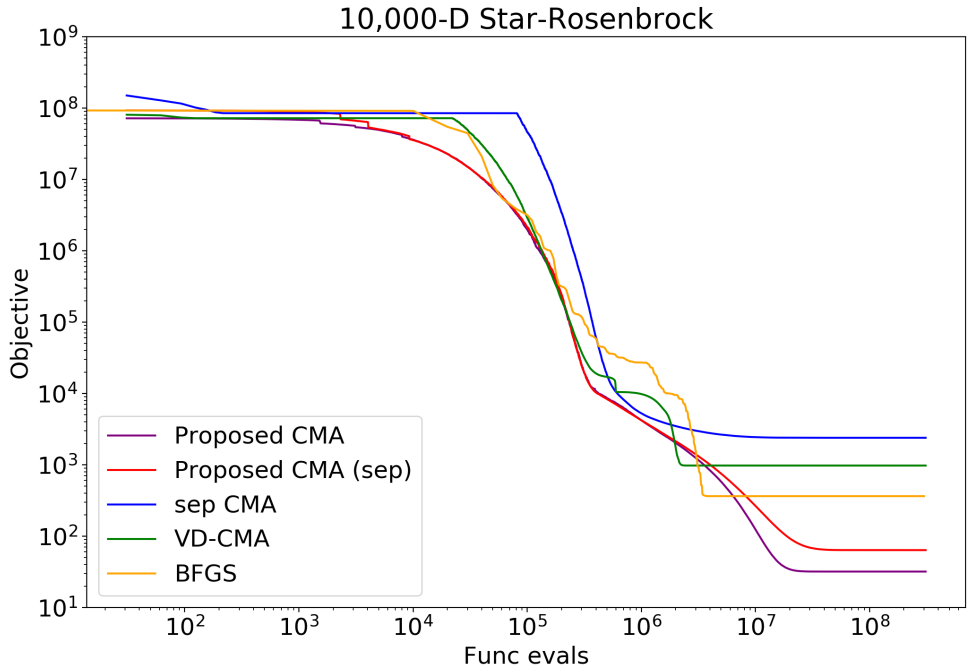

図 4.9: 100,000 次元の Star 型 Rosenbrock 関数の最適化における各種法の評価回数あたりの目的関数値の比較

ることが伺える．また，100 次元の場合では，条件数が 10^6 から 10^7 の間で二つの手法間で評価回数の逆転が起きるのに対し，1,000 次元の場合では，条件数が 10^3 から 10^4 の間で逆転が起きており，次元数によって大きな差が見られた．

4.3.4 次元の制限方法についての比較

本節では，提案手法における次元の制限方法について，ランダムに制限する方法と，制限次元を先頭から順番に固定する方法で比較を行う．

はじめに，10^5 次元の Ellipsoid 関数を用いた比較を行う．それぞれの制限方法についての評価回数と目的関数値，ステップサイズ，分散共分散行列の関係を図 4.15 に示す．図より，制限する次元を先頭から順番に固定した場合が，より少ない評価

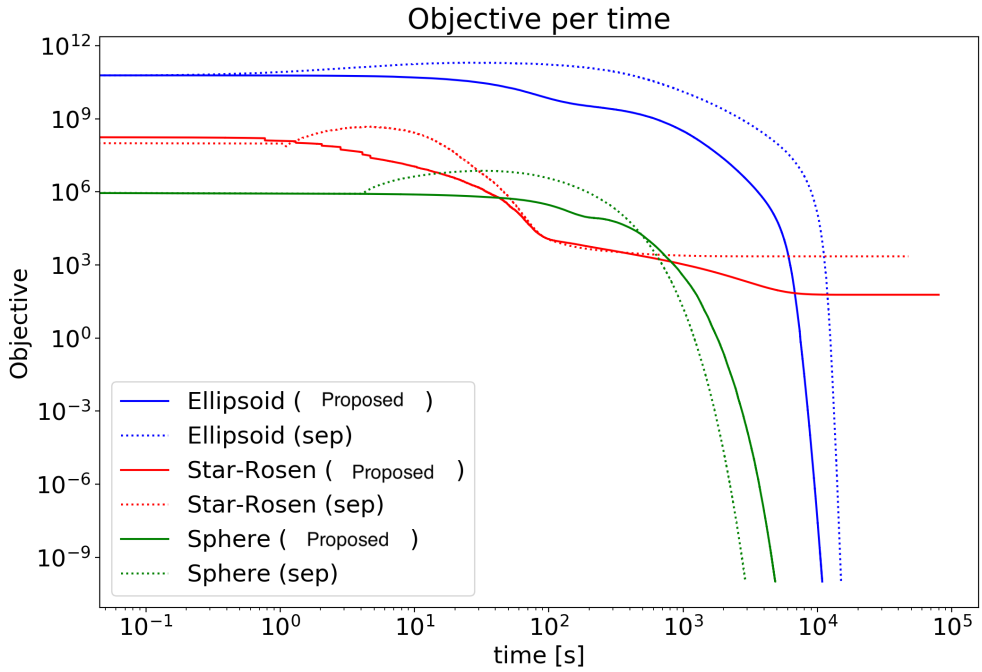

図 4.10: 時間毎の目的関数の値の推移

回数で目標値に到達した．Ellipsoid 関数は各次元について，単調に係数が大きくなるため，順番に次元を制限することで，制限次元内での条件数を大きく減少させることができる．今回の場合では，100 次元での制限を行ったため，制限次元内での条件数は 100 と，Ellipsoid 関数本来の条件数である 10^6 の 0.1%低減されている．このことから，制限次元内での条件数が評価回数の削減に非常に大きな影響を与えていると推測できる．

次に，10^5 次元の Ellipsoid 関数について，各次元の係数をシャッフルし，同様の実験を行った．それぞれの制限方法についての評価回数と目的関数値，ステップサイズ，分散共分散行列の関係を図 4.16 に示す．図より，どちらの制限方法でも同じ程度の評価回数で目的地に到達し，ステップサイズや分散共分散行列も同様の推移となった．Ellipsoid 関数の係数をシャッフルしたことにより，制限する次元を固定

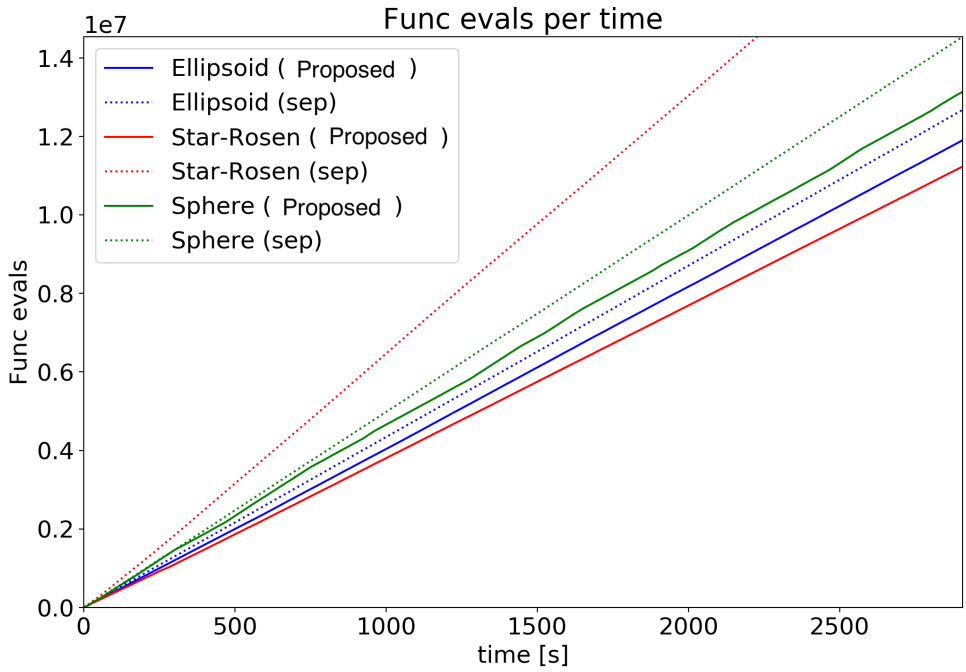

図 4.11: 時間毎の評価回数の推移

した場合でも，毎回制限する次元の組をランダムに変える場合と同程度の条件数になったと考えられる．

最後に，10^4 次元の Star 型 Rosenbrock 関数について，同様の実験を行った．それぞれの制限方法についての評価回数と目的関数値，ステップサイズ，分散共分散行列の関係を図 4.17 に示す．図より，ランダムに次元を制限した場合が，より良い目的関数値に到達した．Star 型 Rosenbrock 関数は，一つ目の次元の変数とその他の次元の変数との間に依存関係があるため，次元を固定した場合では，一つ目の変数と同じ組に制限される変数が一部に限られてしまったことにより，最適化に悪影響が生じたと考えられる．

以上の実験を通して，変数間のスケールの近さや依存関係が既知であれば，それらが同じ組に含まれるように次元を制限することで，最適化性能の向上を図れるこ

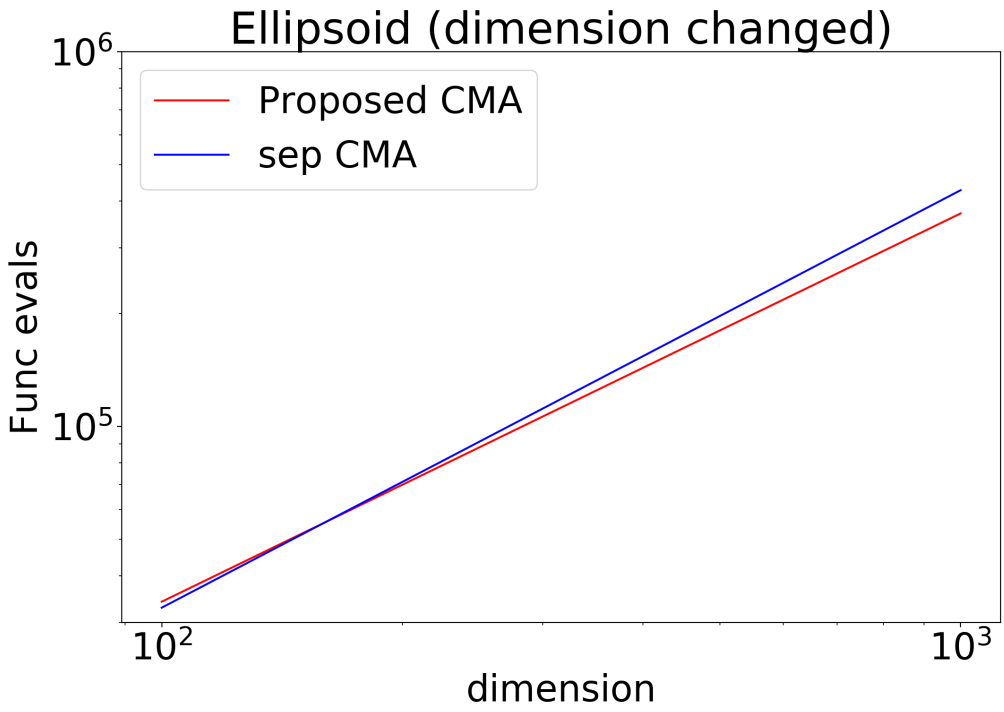

図 4.12: Ellipsoid 関数における次元毎の評価回数の推移

とが確認できた．一方で，ブラックボックス最適化では，目的関数における性質は未知であることから，ランダムに制限することが最も有効であると言える．

4.3.5 制限次元の大きさについての比較

本節では，提案手法の制限次元の大きさの違いついての性能の比較を行う．100,000 次元の Ellipsoid 関数と 10,000 次元の Star 型 Rosenbrock 関数に対して，制限次元を 10, 100, 1,000 と変えて実験を行った．Ellipsoid 関数に対しては sep 型の提案手法を用い，Star 型 Rosenbrock 関数については通常の提案手法を用いた．目標値は 10^{-10} とした．

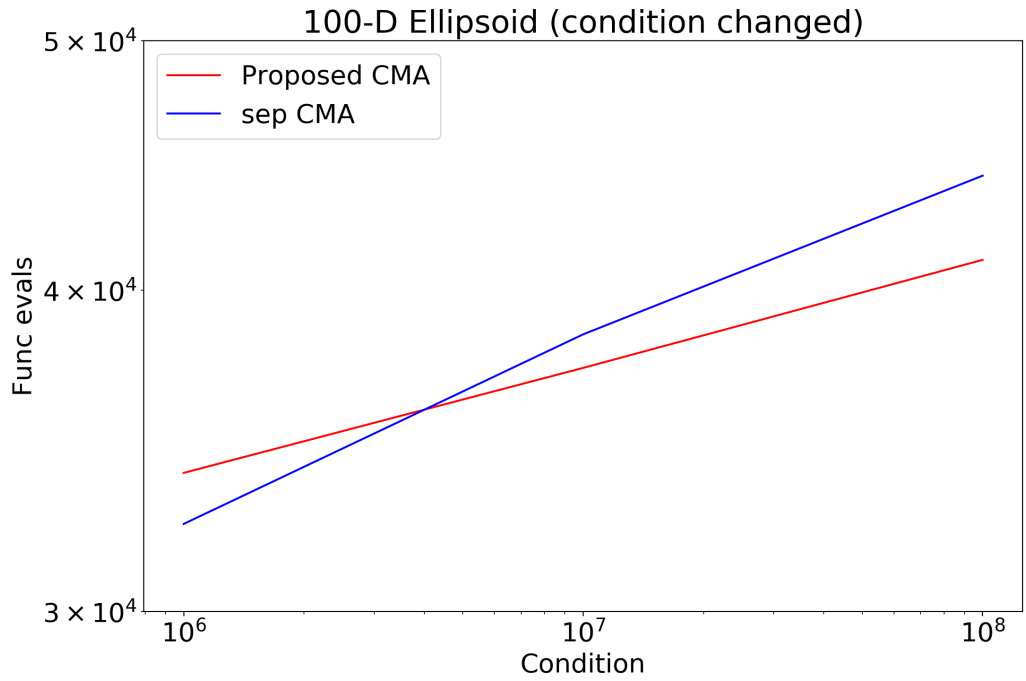

図 4.13: 100 次元の Ellipsoid 関数における条件数毎の評価回数の推移

4.3.5.1 最適化性能の比較

評価回数あたりの目的関数の値を図 4.18 に示す．Ellipsoid 関数では，制限次元を 100 としたものが最も少ない評価回数で目標値に到達した．一方で，Star 型 Rosenbrock 関数では，目標値に到達したものはなかったが，制限次元を 10 としたものが最もよい値を得た．

以上より，さらなる検証が必要ではあるが，目的関数の入力次元数に対して 0.1% 程度を制限次元にするとよいと考えられる．

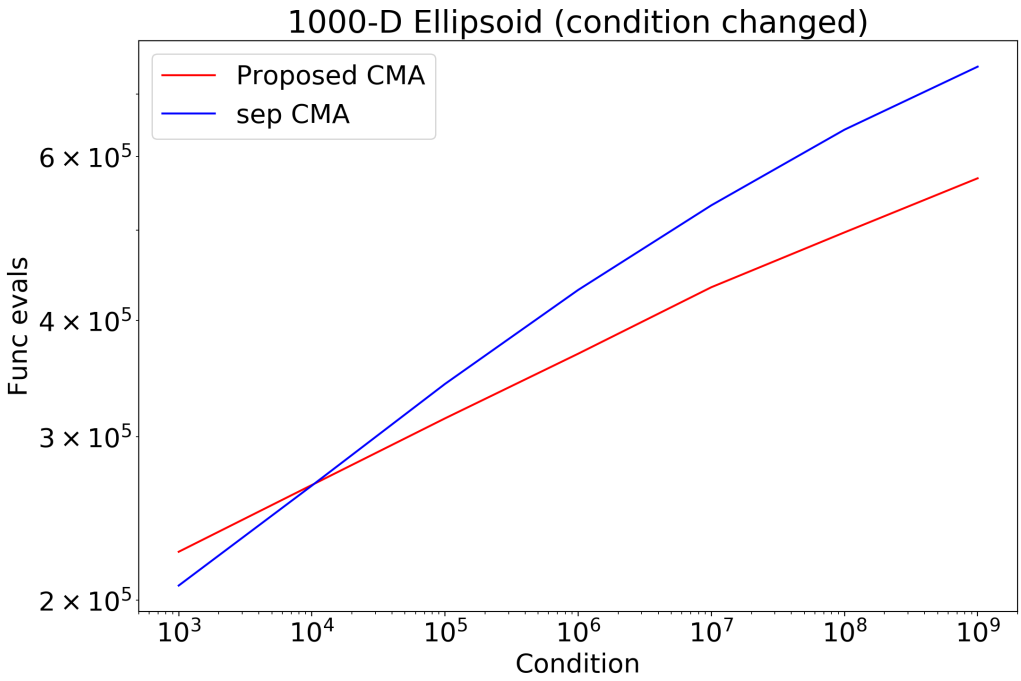

図 4.14: 1,000 次元の Ellipsoid 関数における条件数毎の評価回数の推移

4.3.5.2 計算時間の比較

制限次元数の違いによる時間毎の推移について，目的関数値の推移を図 4.19 に，評価回数の推移を図 4.20 に示す．時間毎の目的関数の値の変化は，評価回数毎の場合とあまり変化はなく，Ellipsoid 関数と Star 型 Rosenbrock 関数共に，制限する次元を全体の次元の 0.1%としたものが良い結果を得た．時間毎の評価回数の変化を見ると，Star 型の Rosenbrock 関数では，制限次元が小さいものほど速いのに対して，Ellipsoid 関数では制限次元を 100 としたものがわずかではあるが最も速くなっている．

時間毎の評価回数における制限次元数と速度の逆転については，提案手法が sep 型であるか否かによって生じていると考えられるが，今後さらなる検証を行う必要がある．

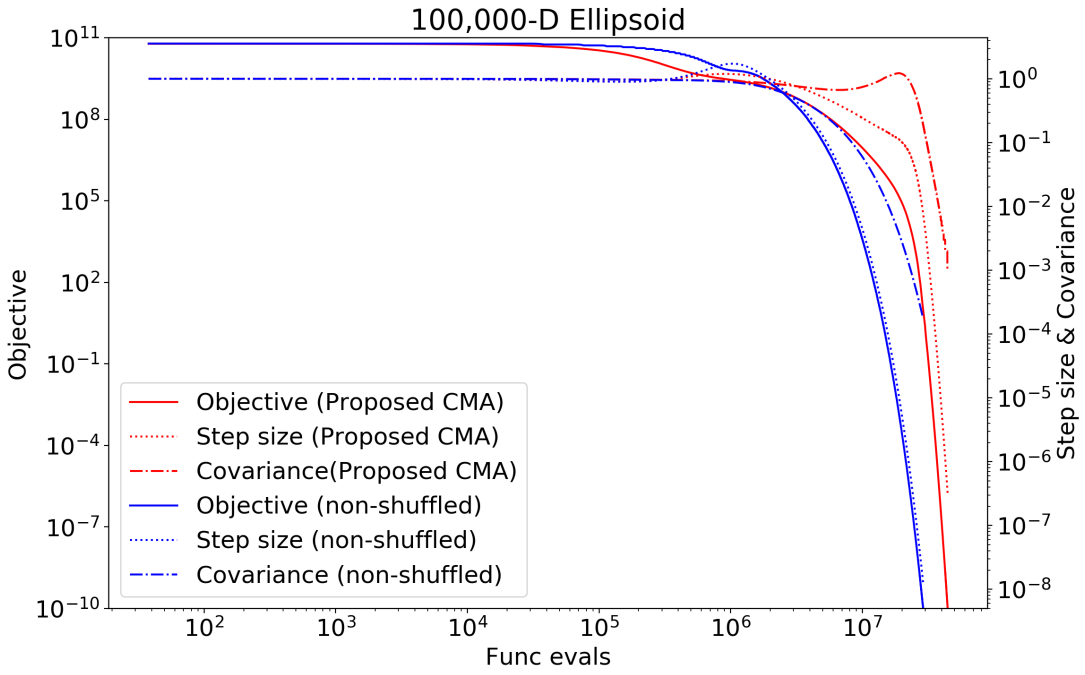

図 4.15: 100,000 次元の Ellipsoid 関数における次元の制限方法についての比較

表 4.2: 計算機環境

CPU	Intel Xeon E5-2680 v4 2.40GHz
RAM	256GB
GPU	NVIDIA Quadro P6000
GPU 環境	Cuda9.0 Cudnn7.2 CuPy4.3.0
OS	Ubuntu 16.04.3 LTS

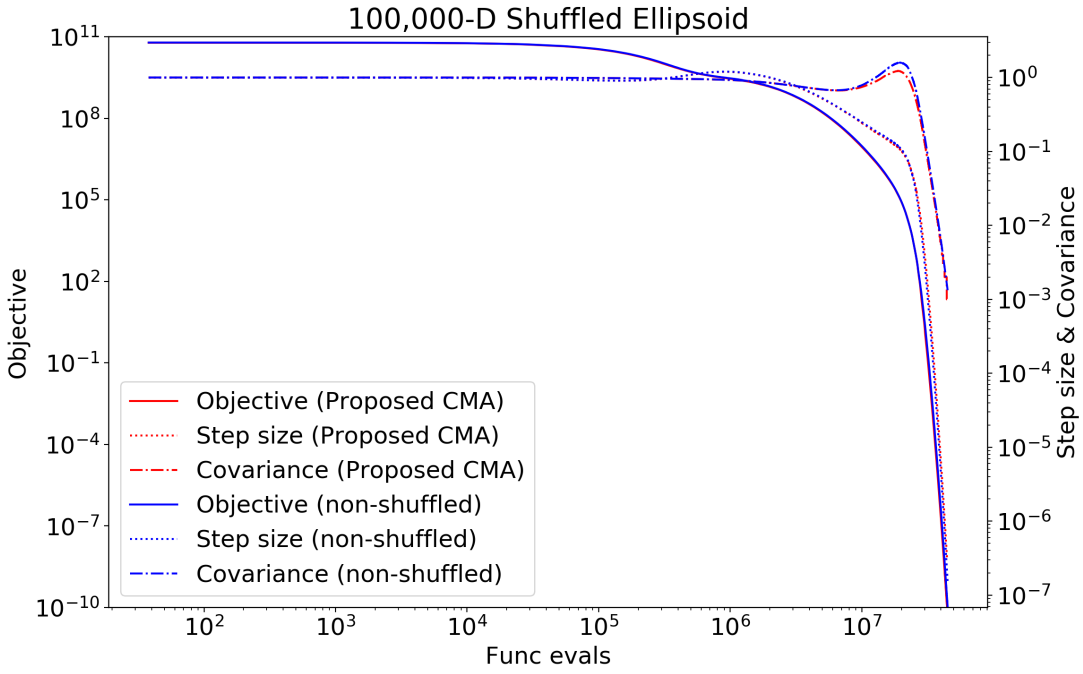

図 4.16: 100,000 次元の Ellipsoid 関数における次元の制限方法についての比較

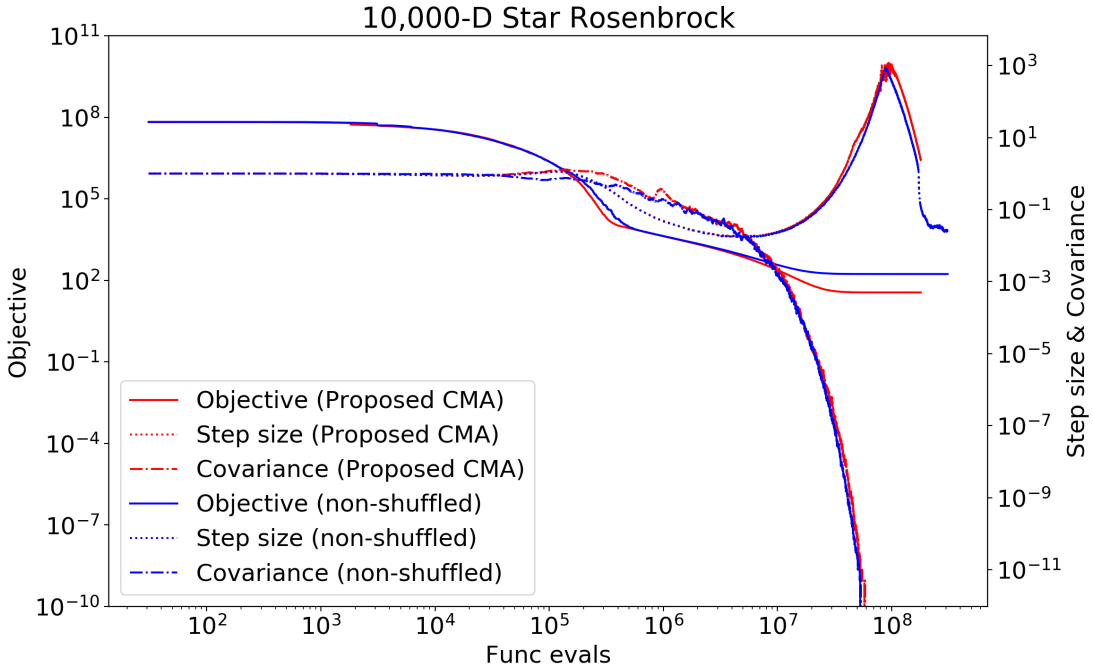

図 4.17: 10,000 次元の Star 型 Rosenbrock 関数における次元の制限方法についての比較

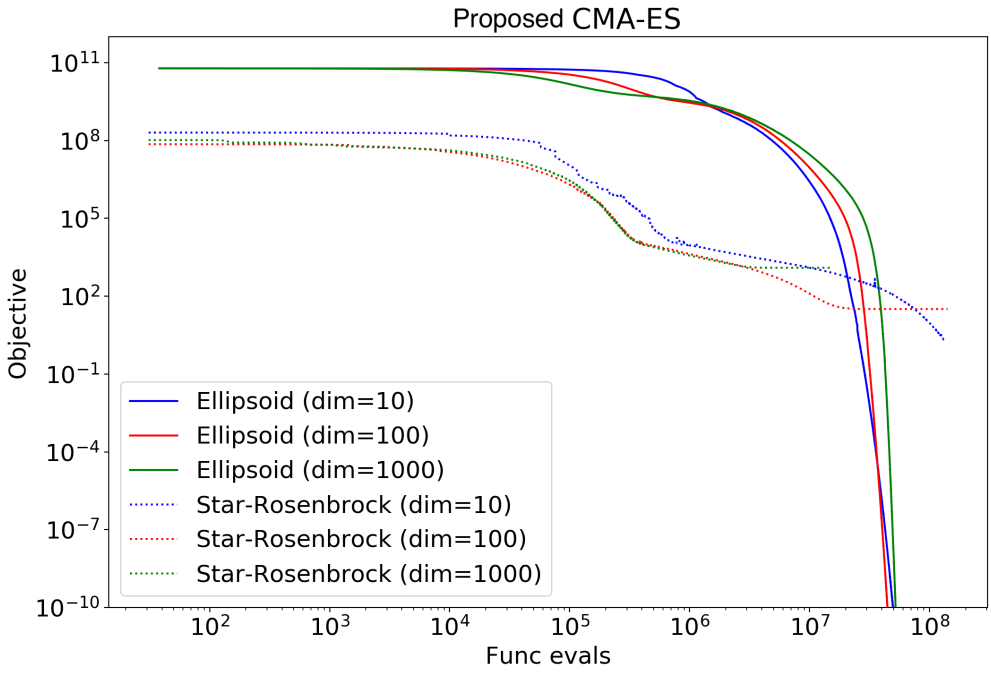

図 4.18: 制限次元数の違いによる評価回数毎の目的関数の値の推移の比較

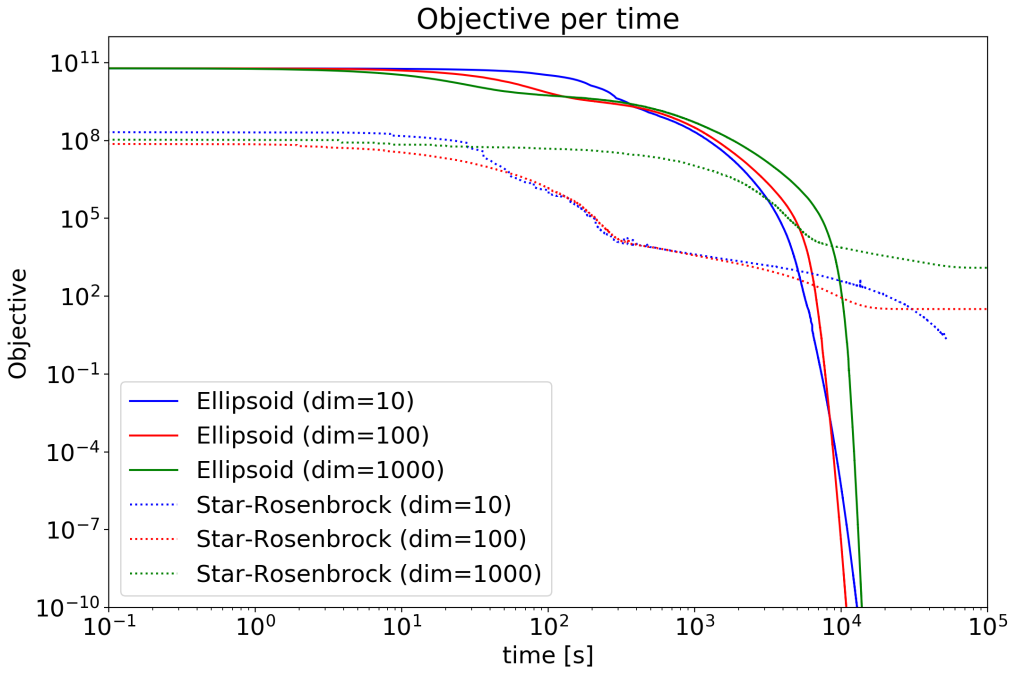

図 4.19: 制限次元数の違いによる時間毎の目的関数の値の推移の比較

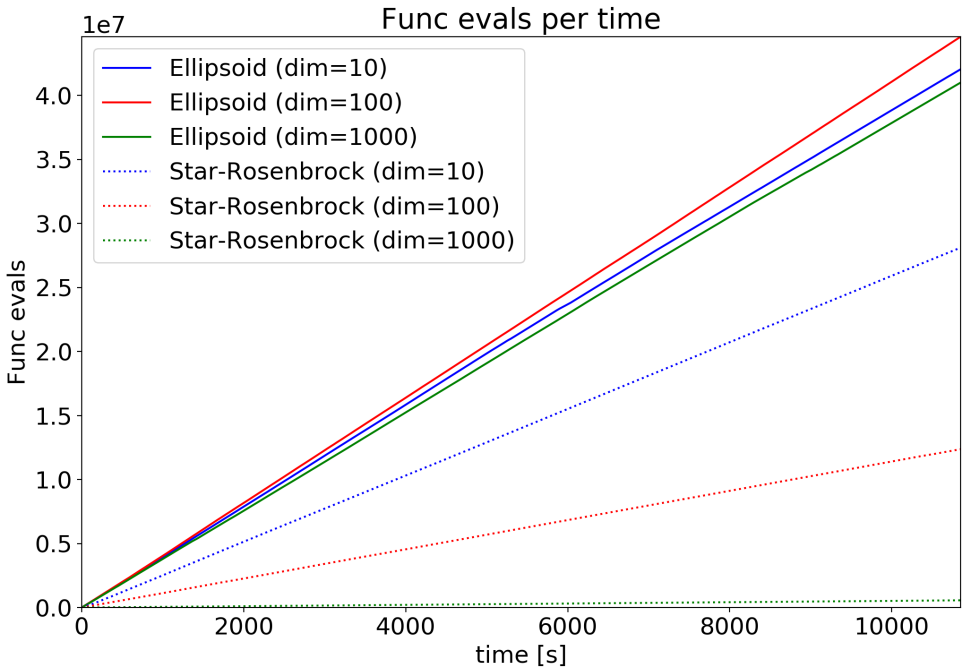

図 4.20: 制限次元数の違いによる時間毎の評価回数の推移の比較

第5章　CMA-ESのニューラルネットワークへの適用

5.1　一層のニューラルネットワークを用いたCMA-ESとSGDとの比較

本節では，CMA-ES のニューラルネットワークの学習における性能を確認するため，一層の非常に浅いニューラルネットワークに適用し，実験を行う．実験に用いた CMA-ES について，第 4 章で提案した確率的 CMA-ES はニューラルネットワーク への学習の適用において，学習の途中でステップサイズが発散し学習が行われなかったため，本実験では，すべて sep-CMA-ES を用いた．

5.1.1　MNISTを用いた実験

本実験では, CMA-ES の集団の大きさを 1000 とし, mini-batch の大きさは 512 とした．また, SGD については, 学習率を 0.001 および 0.01, バッチサイズを 256 とした．Epoch 数は共通に 1000 回とした．使用するニューラルネットワーク は入力層 784 次元, 出力層 10 次元の 2 層ニューラルネットワーク とし, 出力層の活性化関数は softmax 関数, 損失関数には負の対数尤度を用いた．

実験の結果を表 5.1 および図 5.1 に示した．F 値と収束 Epoch は, 1000 回の Epoch 中における最大値とそのときの Epoch を表記した．学習率が 0.001 の SGD については, 1000Epoch 中に収束しなかったため, 収束 Epoch を記さなかった．CMA-ES は SGD および ADAM には及ばなかったものの, ほぼ同等の性能を残した．Epoch を見ると, 最適化に用いたデータ数あたりの収束速度は ADAM が最も速く, 2 番目に

表 5.1: CMA-ES と SGD との比較 (MNIST)

手法	F1-measure	収束 Epoch
sep-CMA-ES	0.918	32
ADAM	0.924	19
SGD (lr=0.001)	0.907	-
SGD (lr=0.01)	0.922	749

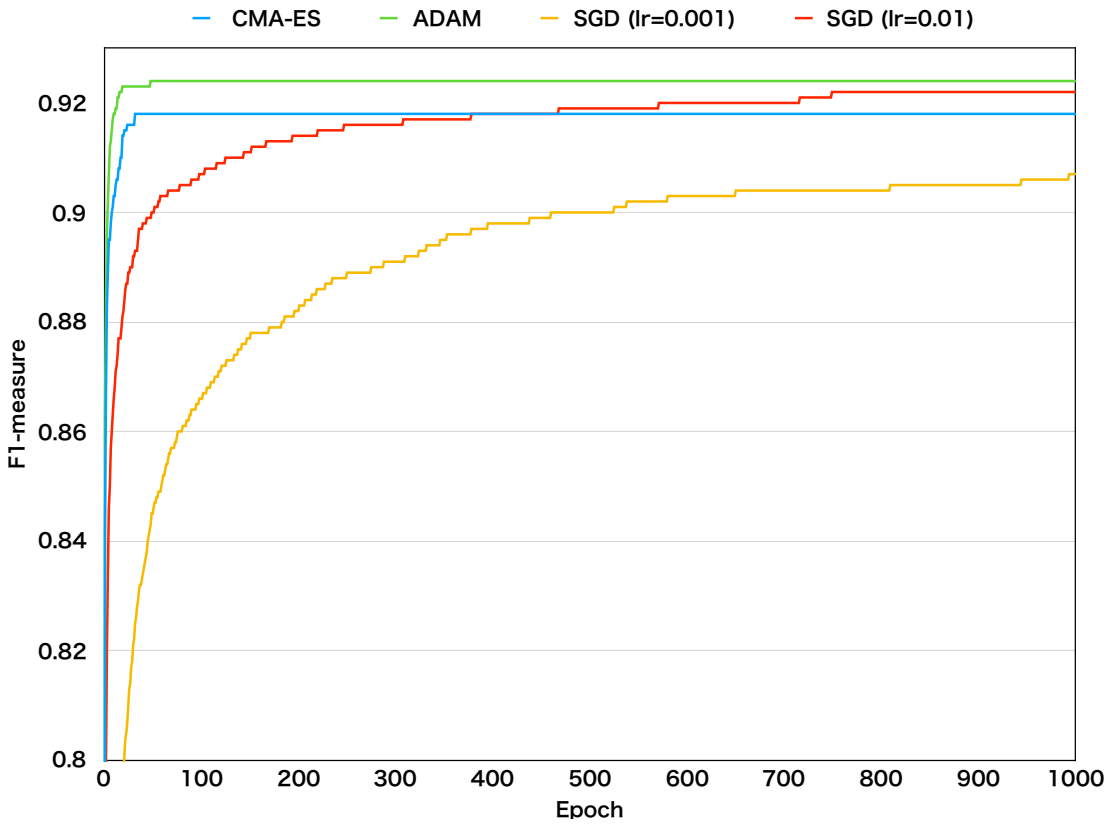

図 5.1: CMA-ES と SGD との比較 (MNIST)

CMA-ES となっており, 比較して SGD は収束がかなり遅くなった. 実際には, CMA-ES は 1Epoch 中に 5000 個のニューラルネットワーク の最適化を行っているため, 実行時間あたりでの収束速度では ADAM と SGD が大きく上回った.

表 5.2: CMA-ES と SGD との比較 (CIFAR-10)

手法	F1-measure	収束 Epoch
sep-CMA-ES	0.317	33
ADAM	0.299	63
SGD (lr=0.001)	0.312	223
SGD (lr=0.01)	0.311	64

5.1.2 CIFAR-10 を用いた実験

本実験では, CMA-ES の集団の大きさを 4000 とし, mini-batch の大きさは 256 とした. また, SGD については, 学習率を 0.001 および 0.01, バッチサイズを 64 とした. その他は, 4.2.1 と同様の設定を用いた.

実験の結果を表 5.2 および図 5.2 に示した. 4.2.1 と同様に F 値と収束 Epoch を示した. この実験においては, CMA-ES が ADAM と SGD を上回る性能を発揮した. また, SGD が ADAM を大きく上回る結果となった. 収束 Epoch 数についても CMA-ES が最も速く, ついで ADAM, SGD となった.

5.2 CMA-ES と SGD の組み合わせ

5.2.1 MNIST

本実験では, CMA-ES の集団の大きさを 1000 とし, mini-batch の大きさは 256 とした. SGD については, いずれの場合も同じ条件とし, 学習率を 0.01, バッチサイズを 64 に設定した. Epoch 数は共通に 1000 とした. 使用するニューラルネットワークは, 10 層のニューラルネットワーク で中間層のユニット数はいずれも 200 で活性化関数は relu 関数を用いた. その他は 4.2 と同様の設定を用いた.

実験の結果を表 5.3 および図 5.3 に示した. 提案手法は SGD と組み合わせたとき, SGD のみを用いたものを上回っており, ADAM のみを用いたものと同じ結果を得た. 一方で ADAM と組み合わせた場合は, ADAM のみを用いたものを下回る結果と

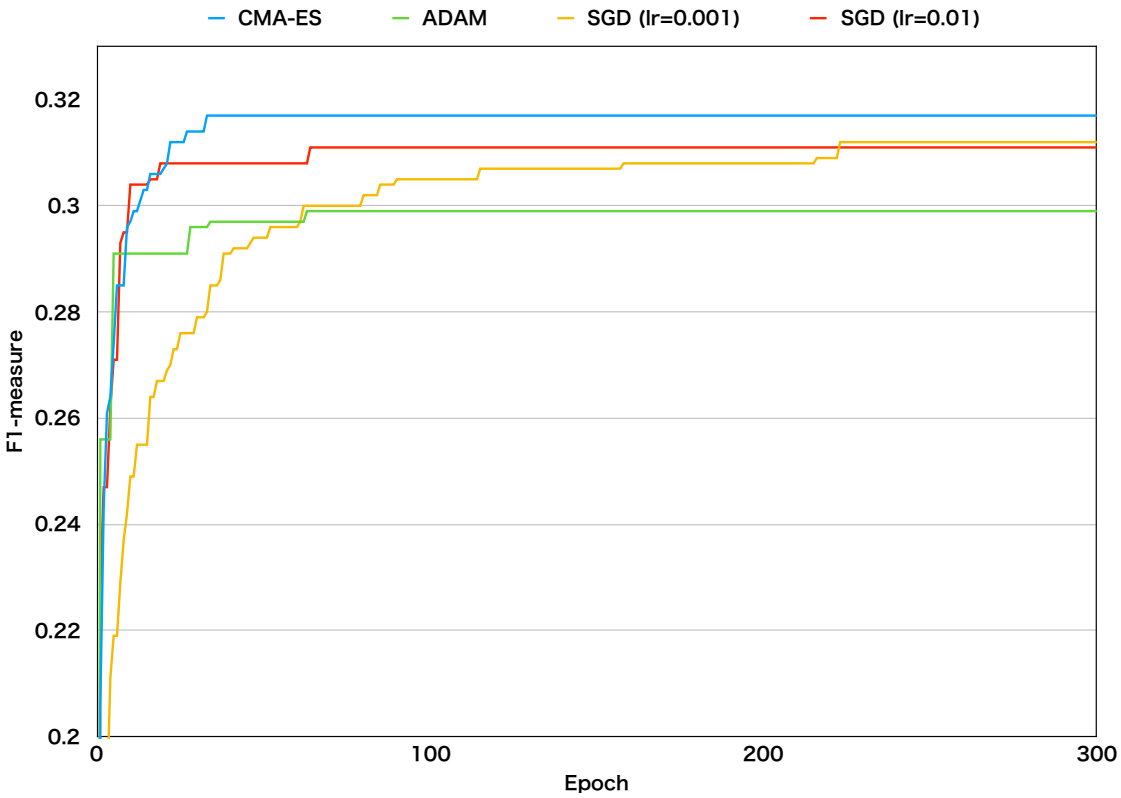

図 5.2: CMA-ES と SGD との比較 (CIFAR-10)

表 5.3: 提案手法と SGD との比較 (MNIST)

手法	F1-measure	収束 Epoch
提案手法 (SGD)	0.982	10
提案手法 (ADAM)	0.980	18
ADAM	0.982	82
SGD (lr=0.001)	0.963	118
SGD (lr=0.01)	0.970	24

なった. Epoch 数を比べると提案手法は最も速く収束しているが, CMA-ES の 1 バッチにつき SGD により 1Epoch 分最適化を行っているため, 単純な比較はできない.

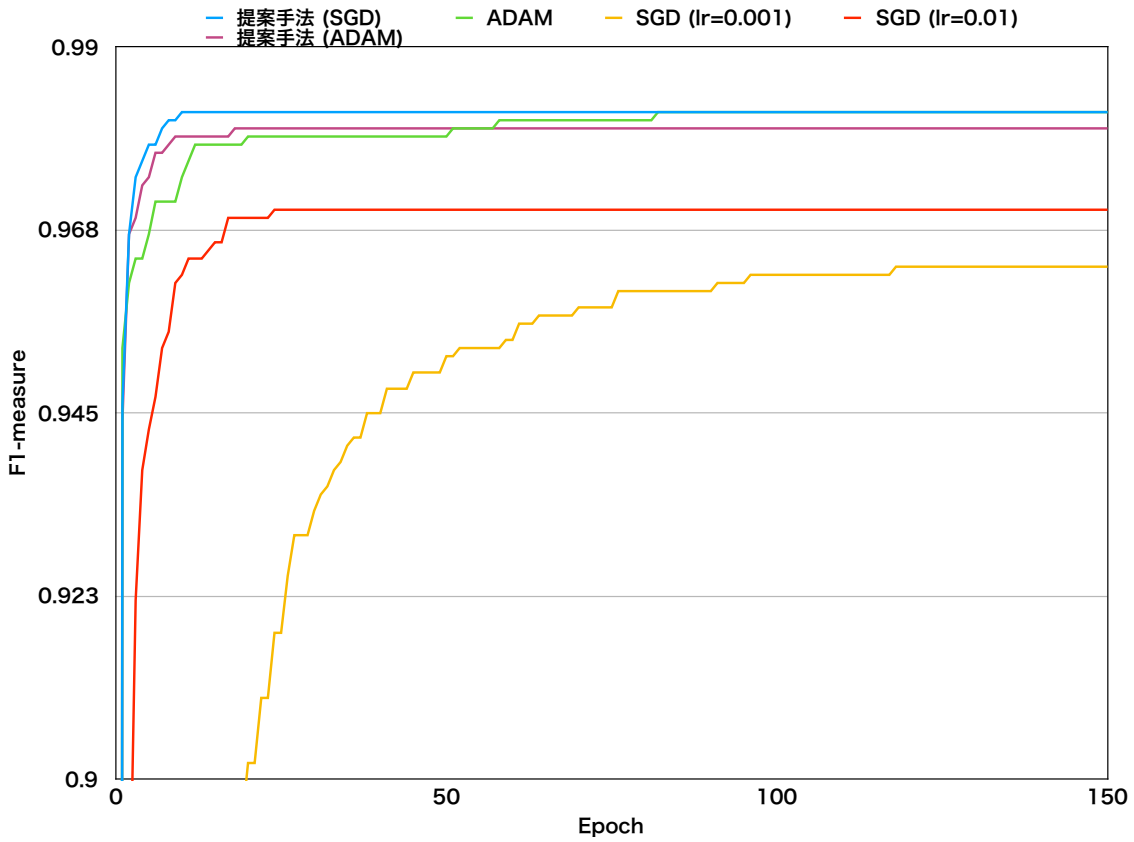

図 5.3: 提案手法と SGD との比較 (MNIST)

5.2.2 CIFAR-10

本実験では, CMA-ES の集団の大きさを 800 とし, mini-batch の大きさは 256 とした. その他は 4.3.1 と同様の設定を用いた.

実験の結果を表 5.4 および図 5.4 に示した. 提案手法は, 収束 Epoch は他の手法よりも少ないものの, F 値が他を大きく下回る結果となった. また, 一層のときと同様に SGD が ADAM を上回る結果となった.

表 5.4: 提案手法と SGD との比較 (CIFAR-10)

手法	F1-measure	収束 Epoch
提案手法 (SGD)	0.327	8
ADAM	0.402	27
SGD (lr=0.001)	0.436	164
SGD (lr=0.01)	0.442	40

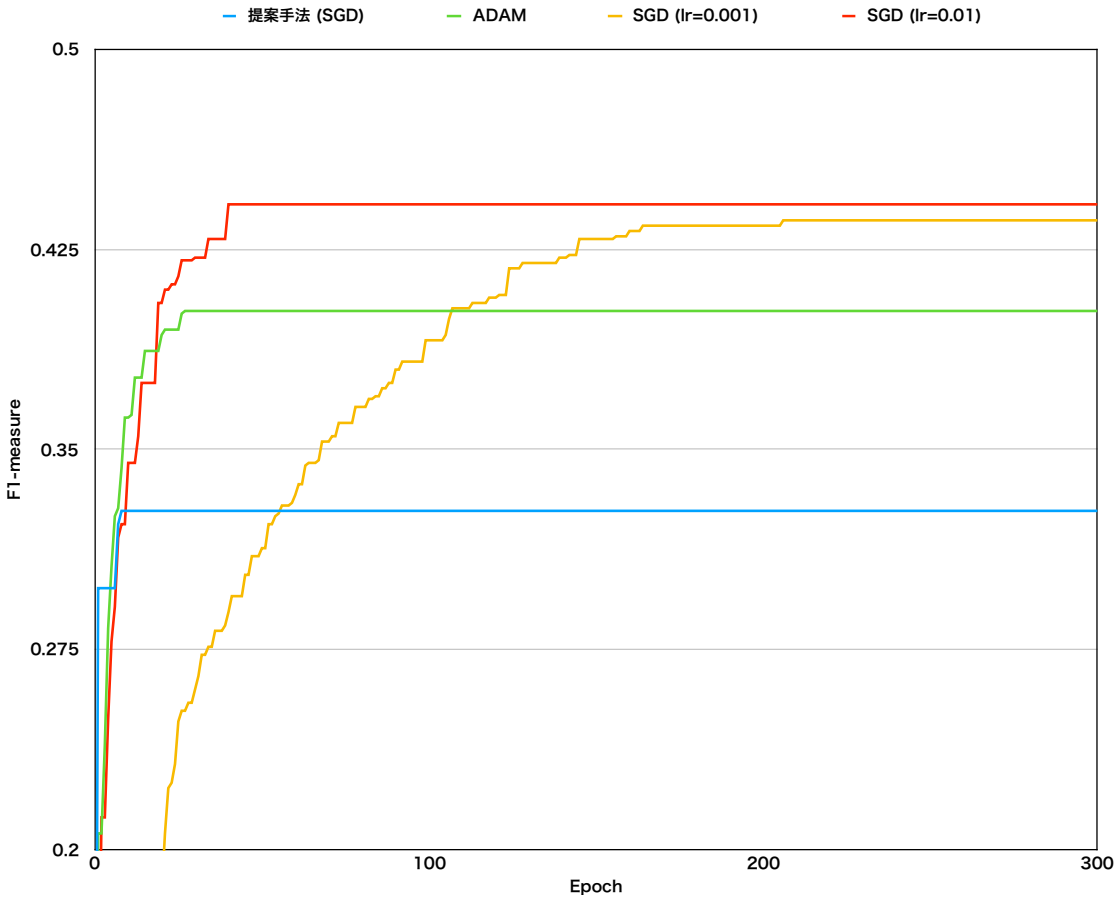

図 5.4: 提案手法と SGD との比較 (CIFAR-10)

5.3 考察

4.2.1 および 4.2.2 から，一層のニューラルネットワーク において，MNIST では ADAM が最もよく，ついで SGD，CMA-ES であったのが，CIFAR-10 では真逆に

CMA-ES が最もよく, ついで SGD, ADAM となる結果が得られた. このことから, CIFAR-10 は MNIST と比較して, 局所解が多数存在し, CMA-ES は複数のネットワークを集団として保持しながら探索することでそれらの局所解に陥りにくいという仮説が考えられる. ADAM が SGD を下回ったのは, 局所解が多数存在したために, 比較的良い局所解の近傍で学習率が収束してしまったことによるものであると考えられる.

4.3.1 から, 10 層のニューラルネットワーク において, 提案手法を SGD と組み合わせた場合は SGD を上回る結果が得られ, ADAM と組み合わせた場合は ADAM を下回る結果が得られた. このことから, SGD は, 多層化されたニューラルネットワーク においては, 出力層から遠い層の最適化が十分に行えず, CMA-ES と組み合わせたことでこれを解消できたと考えられる. 一方で, ADAM は最適化の度合いに応じて徐々に学習率を小さくする手法であることから, 学習率が ADAM の最適化の過程で小さくなってしまい, CMA-ES による重み行列の分布の変化に対応できなかったと考えられる.

4.3.2 については, 提案手法は良い結果を得られなかったが, 4.2.2 より CMA-ES 自体は SGD を上回っていることから, 最適化する対象の次元数に対して, 集団の大きさが十分に足りていなかったことが原因であると考えられる 4.2.2 において, 10240 次元の最適化に対して 4000 個のニューラルネットワーク を用いたのに対して, 4.3.2 では 204800 次元の最適化に対して, メモリの制約上, 800 個のニューラルネットワーク しか用いることができなかったため, 十分な最適化が行えなかったと推察される.

5.4 ラベルノイズに対する頑健性の検証

進化戦略による最適化の特徴として，一般に頑健性が高いと言われているが [25], それについて検証が行われることは非常に少なく，特にニューラルネットワーク の最適化にあたっては先行研究がない.

そこで，本稿では，学習データのラベルに対して人為的にノイズを付与し，それを用いた最適化を行うことで，CMA-ES による最適化の頑健性を検証し，SGD に

表 5.5: CMA-ES と SGD との F 値による比較 (MNIST)

ラベルノイズ [%]	CMA-ES	ADAM	SGD
0	0.918	0.924	0.922
5	0.918	0.915	0.913
10	0.918	0.910	0.910
15	0.917	0.906	0.903
20	0.916	0.899	0.900

より最適化を行った場合との比較を行う.

5.4.1 実験

実世界におけるデータにはラベルノイズが発生している可能性が高いと考えられるが，どの程度のノイズがあるかを知ることが難しいため，検証においては，ベンチマーク用のデータセットである MNIST を用いた．MNIST のデータを 49500 個の学習データと 5500 個のテストデータに分け，学習データのラベルを，一定の割合でランダムに書き換えることで，ラベルノイズを人為的に発生させた．発生させるノイズは 0%から 20%まで 5%刻みで変化させた.

検証には入力層と出力層が直結した 2 層のニューラルネットワーク を用いた．非常に浅いニューラルネットワーク であるが，CMA-ES のメモリ制約上の都合からこれを使用した．比較対象として，CMA-ES と SGD の他に，SGD の学習率を自動的に調整するアルゴリズムである ADAM [26] も用いた．各手法のパラメタ設定として，バッチサイズは全て 256 を用い，学習率は ADAM は初期設定値，SGD については 0.01 とした．評価には F 値を用いた.

実験の結果を表 5.5 及び図 5.5 に示した．表 5.5 を見ると，ノイズが 0%のときは ADAM 及び SGD が優れているが，ノイズが大きくなるに従って ADAM 及び SGD が大きく性能を下げたのに対して，CMA-ES は小さい下げ幅に止まった．ラベルノイズが 20%の場合においては，他の手法が F 値で 0.9 を下回ったのに対して，CMA-ES は 0.9 以上を維持しており，頑健性の高さを示した.

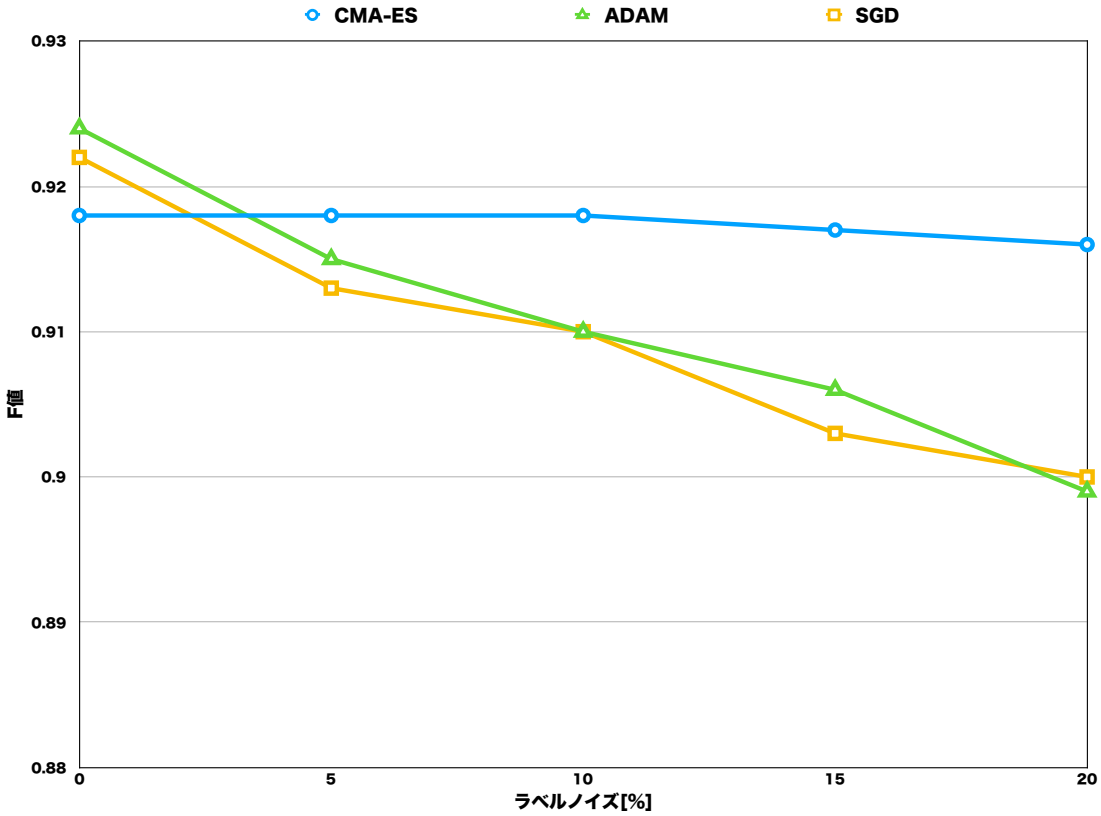

図 5.5: CMA-ES と SGD との比較 (MNIST)

5.4.2 考察

SGD が，ラベルノイズの割合が上がるに連れて急激に性能が低下したことについて，SGD が各学習データのコストに対して勾配を計算し，重み行列の値を更新していることに原因があると考えられる．ミニバッチ学習法を用いることで，1 つの学習データが与える影響を小さくすることができるが，その影響が必ず勾配と重み行列の更新値へ直接及ぶからである．

一方で，CMA-ES においては，各学習データが更新に直接与える影響は，各ニューラルネットワーク の順位を決定する際のコストの算出のみである．確率分布のパラメタの更新は，その順位に基づいて，各ニューラルネットワーク の重み行列の値か

ら行われるため，ラベルノイズを含んだ学習データが順位の上下に影響を与えない限りは，最適化への影響は生じないと言える．これにより，CMA-ES はラベルノイズが上昇しても F 値が大きく下がらなかったと考えられる．

第6章　終わりに

本論文では，(1) 高次元悪条件問題に対するランダムに次元を制限した CMA-ES の提案，と (2) ブラックボックス最適化の実問題への応用としての CMA-ES のニューラルネットワークへの適用，の二つに取り組んだ．(1) では，高次元悪条件最適化問題において，ランダムに次元を制限することで，見かけの条件数を低減し，さらにステップサイズのベクトル化を可能とした CMA-ES を提案することで，探索の速度の向上及び最適解の探索性能の向上を実現した．また (2) では，ブラックボックス最適化の実問題への応用として，CMA-ES をニューラルネットワークに適用することで，CMA-ES を用いた学習のラベルノイズへの頑健性を確認すると共に，SGD と組み合わせたアルゴリズムを提案することで勾配消失問題の低減を試みた．

今後の課題及び展望としては，(1) については，次元を制限することによる見かけの条件数の期待値を数理的あるいは解析的に求めることで，全体の次元数に対して最適な制限する次元数についての検討を行う必要がある．また，変数間依存性が強い関数については，依存関係にある変数の組が制限された次元のに含まれなければならないため，分散共分散行列の値などから依存関係を推測するような手法を検討する必要がある．(2) については，(1) で提案した手法がニューラルネットワークの学習に適用した場合に，ステップサイズが発散してしまい良い結果を得られなかった原因とその解消方法について検討する必要がある．また，今回はニューラルネットワークの学習についてのみ CMA-ES を適用したが，ニューラルネットワークの構造について，CMA-ES を適用することで重み間の共起成分を考慮した構造の最適化を行う手法を検討している．

参考文献

[1] Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. 37(1):55–57.

[2] 阪井 節子 and 高濱 徹行. 最適化手法における関数評価回数の削減手法-ポテンシャルモデルに基づく比較推定法の提案-.

[3] Nikolaus Hansen. The CMA Evolution Strategy: A Tutorial.

[4] Raymond Ros and Nikolaus Hansen. A Simple Modification in CMA-ES Achieving Linear Time and Space Complexity. In Günter Rudolph, Thomas Jansen, Nicola Beume, Simon Lucas, and Carlo Poloni, editors, *Parallel Problem Solving from Nature – PPSN X*, volume 5199, pages 296–305. Springer Berlin Heidelberg.

[5] Ilya Loshchilov. A computationally efficient limited memory CMA-ES for large scale optimization. In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation - GECCO '14*, pages 397–404. ACM Press.

[6] Ilya Loshchilov. LM-CMA: An Alternative to L-BFGS for Large-Scale Black Box Optimization. 25(1):143–171.

[7] Youhei Akimoto, Anne Auger, and Nikolaus Hansen. Comparison-based natural gradient optimization in high dimension. In *Proceedings of the 2014 Conference*

on Genetic and Evolutionary Computation - GECCO '14, pages 373–380. ACM Press.

[8] Youhei Akimoto and Nikolaus Hansen. Projection-Based Restricted Covariance Matrix Adaptation for High Dimension. In *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference - GECCO '16*, pages 197–204. ACM Press.

[9] A. Auger and N. Hansen. A Restart CMA Evolution Strategy With Increasing Population Size. In *2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1769–1776. IEEE.

[10] Kouhei Nishida and Youhei Akimoto. PSA-CMA-ES: CMA-ES with population size adaptation. In *Proceedings of the Genetic and Evolutionary Computation Conference on - GECCO '18*, pages 865–872. ACM Press.

[11] Stanford Vision Lab. ImageNet, 2015.

[12] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. pages 1–23, 2016.

[13] 秋本 洋平. Evolution Strategies による連続最適化 CMA-ES の設計原理と理論的基盤.

[14] Nikolaus Hansen and Anne Auger. Principled Design of Continuous Stochastic Search: From Theory to Practice. In Yossi Borenstein and Alberto Moraglio,

editors, *Theory and Principled Methods for the Design of Metaheuristics*, pages 145–180. Springer Berlin Heidelberg.

[15] Andreas Ostermeier, Andreas Gawelczyk, and Nikolaus Hansen. Step-size adaptation based on non-local use of selection information. In Yuval Davidor, Hans-Paul Schwefel, and Reinhard Männer, editors, *Parallel Problem Solving from Nature — PPSN III*, volume 866, pages 189–198. Springer Berlin Heidelberg.

[16] Nikolaus Hansen and Andreas Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. 9(2):159–195.

[17] Nikolaus Hansen and Stefan Kern. Evaluating the CMA Evolution Strategy on Multimodal Test Functions. In Xin Yao, Edmund K. Burke, José A. Lozano, Jim Smith, Juan Julián Merelo-Guervós, John A. Bullinaria, Jonathan E. Rowe, Peter Tiňo, Ata Kabán, and Hans-Paul Schwefel, editors, *Parallel Problem Solving from Nature - PPSN VIII*, volume 3242, pages 282–291. Springer Berlin Heidelberg.

[18] Gregory Morse and Kenneth O. Stanley. Simple Evolutionary Optimization Can Rival Stochastic Gradient Descent in Neural Networks. *Proceedings of the 2016 on Genetic and Evolutionary Computation Conference - GECCO '16*, (Gecco):477–484, 2016.

[19] Xingwen Zhang, Jeff Clune, and Kenneth O. Stanley. On the Relationship Between the OpenAI Evolution Strategy and Stochastic Gradient Descent. 2017.

[20] G D Magoulas, V P Plagianakos, and M N Vrahatis. Hybrid methods using evolutionary algorithms for on-line training. *Ijcnn'01: International Joint Conference on Neural Networks, Vols 1-4, Proceedings*, 3(1):2218–2223, 2001.

[21] Giorgio Patrini, Alessandro Rozza, Aditya Menon, Richard Nock, and Lizhen Qu. Making Deep Neural Networks Robust to Label Noise: a Loss Correction Approach. sep 2016.

[22] Aritra Ghosh, Himanshu Kumar, and P. S. Sastry. Robust Loss Functions under Label Noise for Deep Neural Networks. pages 1919–1925, 2017.

[23] Jacob Goldberger and Ehud Ben-Reuven. Training Deep Neural Networks using a Noise Adaptation Layer. *Iclr 2017*, (2014):1–9, 2017.

[24] Shigenobu Kobayashi. The frontiers of real-coded genetic algorithms. 24(1):147–162.

[25] Hitoshi Iba. 進化計算と深層学習. *Ohmsha*, 2017.

[26] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. dec 2014.

発表文献

査読なし国内会議

1. 清水洸希，小宮山純平，豊田正史，進化戦略を用いた Neural Network の重み最適化，第 10 回データ工学と情報マネジメントに関するフォーラム (DEIM2018)
2. 清水洸希，小宮山純平，豊田正史，CMA-ES を用いたニューラルネットワークの重み行列の最適化における頑健性の検証，2018 年度 人工知能学会全国大会 (JSAI2018)
3. 清水洸希，小宮山純平，豊田正史，高次元悪条件最適化問題のための確率的次元選択 CMA-ES, 第 11 回データ工学と情報マネジメントに関するフォーラム (DEIM2019) (採録予定)

研究会

1. 清水洸希，豊田正史，分散共分散行列の確率的適応に基づく CMA-ES を用いた高次元悪スケール性関数の最適化，進化計算シンポジウム 2018