グラフ構造によるアルゴリズムの表現を用いた AutoML-Zero の提案

三嶋 隆史[†], 小野 功[†],
東京科学大学情報理工学院[†],

1 はじめに

AutoMLは, 機械学習のモデルを自動で最適化する手法として注目されてきた. これまでのAutoMLに関する研究の多くは, 計算コストを抑えるために人間のデザインに大きく依存した制約付きの空間を探索している. 例えば, ニューラルネットワークの構造探索では, 事前に専門家が用意した経験則的に性能が高くなる層を構成要素として使うことで, 最適化の対象を構成要素の組合わせやハイパーパラメータに限定したり, 重みの更新方法として常に誤差逆伝搬法を用いることで探索空間を制限している[7) 2) 6)].

一方で, Estebanらが言及しているように, 探索空間を限定する既存のAutoMLには, 2つの問題点が存在する[3)]. 1つ目は, 人間がデザインした探索空間にはバイアスがかかってしまい, 人間がまだ発見していないより良いアルゴリズムを見つけられる可能性が減少してしまう点である. 2つ目は, 探索空間を限定する際は, 極めて慎重に行う必要があり[8) 5) 1)], 結果的に研究者に負担が掛かってしまう点である.

Estebanらは, これらのAutoMLにおける課題を解決するため, 人間からの入力を最小限に抑えた機械学習アルゴリズムの探索手法AutoML-Zeroを提案した[3)]. AutoML-Zeroは, Regularized Evolution (RE) を世代交代モデルとして採用した進化計算により, 与えられた機械学習タスクの集合内の各タスクに対する適合度を最大化する機械学習アルゴリズムを探索する. EstebanらのAutoML-Zeroは, 人間の事前知識をほとんど使用しない設計にもかかわらず, AutoML-Zeroは勾配降下法やReLU関数の再発明に成功するなど, 注目すべき成果を示している[3)].

しかしながら, EstebanらのAutoML-Zeroの探索効率には重要な課題が残されている. 例えば, ReLU関数の再発明に成功し, CIFAR-10やMNISTデータセットに対する分類アルゴリズムを発見したSection 4.2の実験では, 膨大な計算リソースが必要とされた[3)]. 具体的には, 1秒間あたり10,000モデルの評価が可能なCPUを搭載したマシンを10,000台使用し, 約5日間, 評価回数にして10^{12}オーダーの計算を実行している. 我々は, 探索効率を低下させ得る問題として, 探索空間の冗長性の問題, 良質な構造を子に継承できない問題, 集団の多様性維持に関する問題に着目した.

本研究の目的は, EstebanらのAutoML-Zeroの探索効率の問題に対処した手法を提案し, 主要な回帰問題や分類問題で既存手法と性能を比較することである. 提案手法では, グラフ構造を用いてアルゴリズムを表現する手法（アルゴリズムグラフ）とその突然変異を導入することで, 探索空間の冗長性の問題と良質な構造を子に継承できない問題に対処する. また, 集団の多様性維持に関する問題に対処するために, Minimal Generation Gap (MGG) [4)] による世代交代モデル, 集団内の同一個体の重複排除, および希少度に基づく生存選択を導入する.

Proposal of AutoML-Zero Using Graph-Based Algorithm Representation

† Ryuji Mishima (mishima.r.ab@m.titech.ac.jp)
† Isao Ono (isao@dis.titech.ac.jp)
School of Computing, Institute of Science Tokyo (†)

2 問題設定

本研究で対象とするAutoML-Zeroは, 与えられた機械学習タスク集合$\mathcal{T}$内の各機械学習タスクに対して, 高い適合度を持つアルゴリズム$a^* \in \mathcal{A}$を探索する問題であり, 以下のように定式化される.

$$a^* = \arg\max_{a \in \mathcal{A}} \frac{1}{|\mathcal{T}|} \sum_{T \in \mathcal{T}} F(a, T) \qquad (1)$$

ここで, $\mathcal{A}$はアルゴリズムの集合, $F(a, T)$はアルゴリズムaのタスクTに対する適合度である.

タスク集合$\mathcal{T}$は, 複数の機械学習タスクによって構成される. 各機械学習タスク$T \in \mathcal{T}$は, 入力ベクトル$\boldsymbol{x}_j$と正解ラベルy_jの順序対の集合であり, 以下のように定義される.

$$T = \left\{ (\boldsymbol{x}_j, y_j) \mid \boldsymbol{x}_j \in \mathbb{R}^d,\ y_j \in \mathbb{R} \right\}$$

ここで, Nおよびdはそれぞれタスクごとに定まるデータの個数とタスクの次元である. また, タスクT内のデータは, 学習用データD_{train}と検証用データD_{valid}に分割される.

アルゴリズムaのタスクTに対する適合度$F(a, T)$は, 学習データD_{train}をアルゴリズムaで学習させた上で, 検証データD_{valid}に対する損失を$[0, 1]$に変換することで計算される. 適合度は, 1に近いほどタスクTに適合している, 言い換えればTの検証データに対する損失が小さいことを意味する. 損失や損失を$[0, 1]$に変換する関数はユーザによって与えられる.

3 既存手法

3.1 アルゴリズムの表現

RE-AutoML-Zeroにおいて機械学習アルゴリズムは, 小さな仮想メモリで動作するプログラムとして表される. 仮想メモリには, スカラー, d次元ベクトル, $d \times d$次元行列を複数個格納できる.ここで, dはタスク集合$\mathcal{T}$に含まれるタスクTの入力ベクトルの次元である. 以降, スカラーを格納する変数を$s0, s1, \cdots$, ベクトルを格納する変数を$v0, v1, \cdots$, 行列を格納する変数を$m0, m1, \cdots$と表す. $s0$, $s1$, $v0$は, それぞれ正解ラベル, アルゴリズムによる予測ラベル, 入力ベクトルを格納する先として使われる特別な変数である. その他の変数は学習対象のパラメータ（学習パラメータ）を格納したり, 計算結果を一時的に保存する用途で用いられる. 変数の個数の上限はスカラー, ベクトル, 行列それぞれに対してユーザが指定する必要がある.

Algorithm 1 既存手法のアルゴリズムの表現

```
1:  function Setup
2:      s3 = 0.01 // 学習率の設定
3:  end function
4:  function Predict
5:      s6 = dot(v6, v0) // 傾きを適用
6:      s1 = s7 + s6 // 切片を適用
7:  end function
8:  function Learn
9:      s4 = s0 − s1 // 予測誤差を計算
10:     s6 = s3 ∗ s4 // 学習率の適用
11:     v3 = s6 ∗ v0 // 傾きの更新分を計算
12:     v6 = v6 + v3 // 傾きの更新
13:     s7 = s7 ∗ s6 // 切片を更新
14: end function
```

アルゴリズムは, Algorithm.1に示したように, Setup, Predict, Learnの3つの関数で表現される. 各関数はEstebanらの論文[3]のTable S1に示されている64個の命令の列で構成される. 命令は, 人間のバイアスを与えすぎないようにするため, 高校数学で学ぶ程度の演算のみを使用し, 機械学習のアルゴリズムの概念や行列の分解等の演算は含まれていない. また, 命令に与える引数は, 基本的には仮想メモリに格納されているスカラー$s1, s2, \cdots$, ベクトル$v1, v2, \cdots$, 行列$m1, m2, \cdots$のいずれかである. 一部例外として, 正規分布による乱数生成の命令等では, μ, σ等の定数が入力されることもある.

3.2 アルゴリズムの評価

タスク集合$\mathcal{T}$内の1つのタスクTに対するアルゴリズムの評価は, Algorithm 2に示した流れで行われる. Algorithm 2の入力は評価対象

Algorithm 2 タスクTに対するアルゴリズムの評価

initialize_memory()
Setup()
for $e = 0, 1, \cdots, N_{\text{epochs}}$ **do**
  for all $(\boldsymbol{x}_j, y_j) \in D_{\text{train}}$ **do**
    $v0 \leftarrow \boldsymbol{x}_j$
    Predict()
    $s1 \leftarrow \text{Normalize}(s1)$
    $s0 \leftarrow y_j$
    Learn()
  end for
end for
$l_{\text{sum}} = 0.0$
for all $(\boldsymbol{x}_j, y_j) \in D_{\text{valid}}$ **do**
  $v0 \leftarrow \boldsymbol{x}_j$
  Predict()
  $s1 \leftarrow \text{Normalize}(s1)$
  $l_{\text{sum}} \leftarrow l_{\text{sum}} + \text{Loss}(y, s1)$
end for
$l_{\text{mean}} \leftarrow l_{\text{sum}} / |D_{\text{valid}}|$
fitness $= \text{Rescale}(l_{\text{mean}})$
return fitness

のアルゴリズム$a = (\text{Setup}, \text{Predict}, \text{Learn})$, タスク$T \in \mathcal{T}$の学習用データ$D_{\text{train}}$および検証用データ$D_{\text{valid}}$であり, 出力は評価対象のアルゴリズムのタスク$T$に対する適合度$F(a, T)$である.

1行目で行ったメモリの初期化以降, 特別な変数$s0$, $s1$, $v0$以外の変数への代入は, 関数以外で行われることがない. そのため, 評価対象のアルゴリズムの各関数では, $s0$, $s1$, $v0$以外の変数に学習パラメータを格納することで, 初期化時から検証時まで値を引き継ぐことができる.

3.3 既存手法の世代交代

Estebanらが提案したRE-AutoML-Zeroでは, N_{pop}個のアルゴリズムをランダムに初期集団として生成した後に, Fig.1のSTEP1からSTEP4に示したRegularized Evolution (RE)を繰り返し行うことで, 最適なアルゴリズムの探索を行う. REでは, STEP1で最も古い個体を削除した後に, STEP2でトーナメント選択を行う. その後, STEP3でトーナメント選択によって選ばれた個体をコピーし, STEP4で一定確率p_{mutate}で突然変異を行う. 集団サイズN_{pop}, トーナメントサイズK, 突然変異確率p_{mutate}はユーザパラメータである.

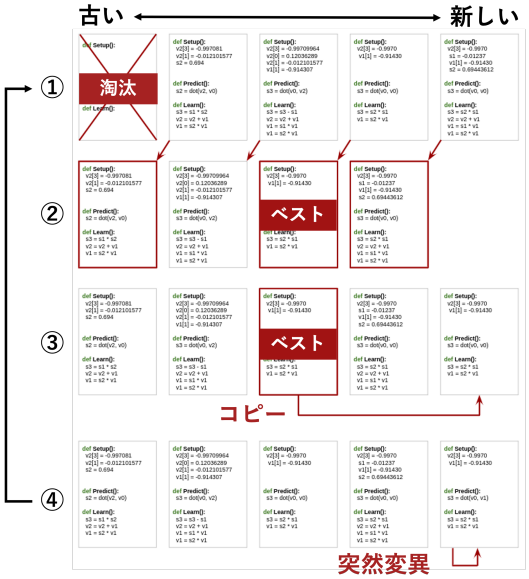

Fig. 1　RE-AutoML-Zero[3] の世代交代モデル

既存手法は, 次の3つの突然変異の手法を用いて子個体を生成する. (1) ランダムに命令を追加または削除する突然変異, (2) 関数のいずれかの命令列全体をランダムに書き換える突然変異, (3) ある命令の入力, 出力, 即値をランダムに変更する突然変異である.

4 既存手法の問題点

Estebanらが提案したRE-AutoML-Zeroには, 探索空間の冗長性に関する問題, 良質な構造を子に継承できない問題, 集団の多様維持に関する問題が存在し, 探索が非効率的になっていると考えられる. 以下で, それぞれの問題点について詳しく説明する.

4.1 探索空間の冗長性に関する問題

既存手法の探索空間は探索空間が冗長であり, 無意味な評価が多く発生し, 探索効率を低下していると考えられる. 既存手法の冗長性には, 変数名や命令の実行順の違いに

よる冗長性と非妥当なアルゴリズムによる冗長性の2つが存在する.

変数名や命令の実行順の違いによる冗長性は, Algorithm.1を例に挙げて説明する. Algorithm.1において, $s6$という変数が$s8$に変わったり, 12行目と13行目が入れ替わっても実行結果は同じである. 既存手法では, 実行結果に違いがなく, 変数名や命令の実行順が異なるアルゴリズムを同一視することができず, 探索空間が冗長になっている.

非妥当なアルゴリズムによる冗長性は, 任意の機械学習アルゴリズムにおいて満たすべき条件を満たさないアルゴリズムが探索空間に含まれていることに起因する. 例えば, 正解ラベルや入力ベクトルを全く利用しないアルゴリズム, 代入しても利用されない変数が存在するアルゴリズムなどがその例である. しかし, 既存手法ではこれらの条件を考慮せず, 全てのアルゴリズムを探索しているため, 探索空間に冗長になっている.
これらの条件は, 機械学習タスクを高い適合度で解くために, タスクの種類によらず普遍的に必要不可欠な性質であるため, 非妥当なアルゴリズムを探索することは非効率的であると考えられる.

4.2 良質な構造を子に継承できない問題

既存手法の突然変異は, 親個体が妥当なアルゴリズムであったとしても, 非妥当なアルゴリズムに変化してしまう. 実際, 予備実験の結果により, 妥当な線形回帰アルゴリズムを突然変異させると, 98%以上の確率で妥当ではないアルゴリズムに変化することが確かめられている.

また, 既存手法では突然変異(2)の関数全体をランダムに変更する破壊的な変化が高頻度で発生するため, 関数中の良質な部分命令列が存在していたとしても, 子個体にその良質な部分命令列が継承ができなくなってしまう可能性が高いと考えられる. 故に, 既存手法の子個体生成方法では, 関数を逐次改善することが難しいと考えられる.

4.3 集団の多様性維持に関する問題

既存手法は, 集団の多様性が維持しにくく, 局所最適解に陥りやすい問題が存在すると考えられる. 既存手法の世代交代モデルであるREは, 佐藤らの論文[4] の考察を踏まえると, 無条件で元の集団の個体が淘汰されている点, 複製選択で強い選択圧が掛かるトーナメント選択が用いられている点, 生存選択で選ばれる個体が淘汰される個体と無関係である点で集団の多様性を失いやすいと考えられる. また, 既存手法では同等の個体や類似個体が考慮されておらず, 希少な構造を持つ個体が他の数の多い個体によって淘汰される可能性が高いと考えられる. これにより, 有望な個体が早期に淘汰され, 局所最適解に収束する問題が生じると考えられる.

5 提案手法

提案手法では, 既存手法の問題点として挙げた探索空間の冗長性, 個体の良質な構造を子に継承できない問題, 集団の多様性維持に関する問題に対処した手法を提案する. RE-AutoML-Zeroの人間の事前知識や介入を最小限にできる利点は残しつつ, 探索効率を向上させることを目指す. 提案手法では, グラフ構造を用いてアルゴリズムを表現する手法（アルゴリズムグラフ）とその突然変異を導入することで, 探索空間の冗長性の問題と良質な構造を子に継承できない問題に対処する. また, 集団の多様性維持に関する問題に対処するために, Minimal Generation Gap (MGG) [4] による世代交代モデル, 集団内の同一個体の重複排除, および希少度に基づく生存選択を導入する.

5.1 アルゴリズムグラフ

本論文では, アルゴリズムグラフを非巡回順序付き有向グラフ$G = (U, E)$として定式化する. ここで, Uはノードの集合, $E \subset U \times \mathbb{N} \times U$は順序エッジの集合である. 順序エッジ$(u_1, n, u_2)$は通常の有向グラフのエッジとは異なり, 始点$u_1$と終点$u_2$の他に, u_1の何番目のエッジであるかを表す順序値$n \in \mathbb{N}$を持つ.

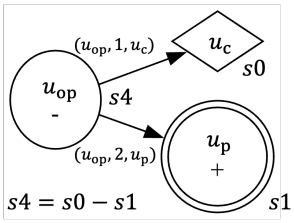

Fig. 2　Algorithm.1における9行目の命令$s4 = s0 - s1$に対応するAGの命令ノード

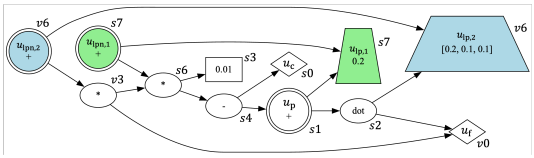

Fig. 3　アフィン回帰と解釈可能なアルゴリズムAlgorithm.1のAG

アルゴリズムグラフ$G = (U, E)$のノード集合Uに属する各ノードには, 以下の種類が存在する.

定数ノード

定数値を表すノードでスカラー, ベクトル, 行列のいずれかの定数値が格納されている.

入力ベクトルノード: u_{f}

学習データや検証データの入力ベクトルを代入するノード. 既存手法のアルゴリズムの表現における$v0$に対応する.

正解ラベルノード: u_{c}

学習データの正解ラベルを代入するノード. 既存手法のアルゴリズムの表現における$s0$に対応する.

LPノード: $u_{\mathrm{lp},i}$

学習時に逐次更新される学習パラメータ (Learning Parameter, LP) を表すノード. 学習時の最初のステップでは, 当該学習パラメータの初期値が格納されており, それ以降のステップでは, 1ステップ前で更新された値が格納されている. 一般に複数存在するため, 順序をつけて$u_{\mathrm{lp},i}$と表す.

命令ノード

命令セット内の命令が割り当てられたノード. 命令の入力に対応するノードを子ノードとして持ち, 命令の出力値が格納されている. n番目の子ノードが第n引数に対応する. また, 出力の利用用途が特殊な命令ノードとして, 以下の予測ノードとNLPノードが存在する.

予測ノード: u_p

出力が予測ラベルとして利用される特殊な命令ノード. 既存手法のアルゴリズムの表現における$s1$に対応する. また, 予測に正解ラベルを利用することはできないため, 予測ノードの子孫ノードに正解ラベルノードを含めることはできない.

NLPノード: $u_{\mathrm{nlp},i}$

出力が次のステップの特定の学習パラメータ (Next Learning Parameter, NLP) の値として利用される特殊な命令ノード. NLPノードは, U内にLPノードと同様の個数存在し, NLPノードとLPノードは1対1で対応するため, 添え字を対応させて$u_{\mathrm{nlp},i}$と表す.

各ノードにはスカラー型, ベクトル型, 行列型のいずれかが割り当てられており, 命令ノードの子ノードを割り当てる際は, 命令の入力の型の整合性が保たれる必要がある.

AGにおける1つの命令ノードとそのノードを始点とするエッジは, 第3.1節で述べた既存手法のアルゴリズムの表現における1行と対応する. 例えば, Fig.2に示した命令ノード$u_{\mathrm{op}} \in U$と2つのエッジ$(u_{\mathrm{op}}, 1, u_{\mathrm{c}})$と$(u_{\mathrm{op}}, 2, u_{\mathrm{p}})$は, Algorithm.1の9行目の命令$s4 = s0 - s1$と対応している.

アフィン回帰と解釈可能なアルゴリズムAlgorithm.1のAGをFig.3に示す. 図において, 命令ノードは円形, 定数ノードは四角形, データノードはひし形, LPノードは台形で表現されている. 特殊な命令ノードである予測ノードとNLPノードは二重丸で示した. また, 対応関係にあるLPノードとNLPノー

ドは同じ色で着色した. 命令ノードには, 割り当てられている命令の演算子, 定数ノードには代入されている定数値, LPノードには学習パラメータの初期値が記載されている. 子ノードを持つノードの場合は, 図の上部にある子ノードを順序値が小さいものとする. また, アルゴリズムグラフでは不要ではあるが, Algorithm.1との対応関係を明確にするために, 各ノードには代入先の変数名を記載している.

5.2 妥当なアルゴリズムグラフ

また, 提案手法では非妥当なアルゴリズムが探索対象とならないように, 以下に示す妥当なアルゴリズムの条件を満たす妥当なアルゴリズムグラフ (Valid Algorithm Graph, VAG) のみを探索対象とする.

1. Learn関数で逐次更新される全ての学習パラメータが以下の条件を満たすこと.
   (a) 1ステップ前の自分自身の値に依存して更新されていること.
   (b) Predict関数で利用されている場合は, 正解ラベル$s0$に依存して更新されていること.
   (c) Predict関数で利用されている場合は, 予測ラベル$s1$に依存して更新されていること.
2. Predict関数で予測ラベル$s1$の算出に, 以下が利用されていること.
   (a) 入力ベクトル$v0$
   (b) 1つ以上のLearn関数で逐次更新される学習パラメータ
3. 全ての変数が最終的に予測ラベル$s1$の算出に寄与すること.

提案手法ではアルゴリズムをグラフ構造で表現しているため, 妥当なアルゴリズムの条件が成立するか否かの判定が容易である. これにより, 非妥当なアルゴリズムが探索空間に含まれることを効果的に防ぐことができる.

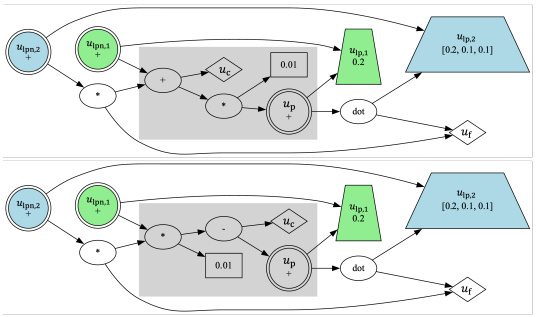

Fig. 4 部分グラフの再構築の突然変異の前後のVAGの様子

5.3 初期個体の生成

提案手法では, VAGのみを探索対象とするために, 予測ノードを起点として, 妥当なアルゴリズムの条件を反映させながら, グラフを段階的に構築していくことでVAGの生成を行う. はじめに, 予測ノードとその子孫ノードに規定の個数になるまで命令ノードを追加する. その後, 予測ノードに必要な依存関係を追加した上で, 空いているはノードにLPノードや定数ノードを追加する. その後, LPノードに対応するNLPノードとその子孫ノードに規定の個数になるまで命令ノードを追加する. そして、最後にNLPノードに必要な依存関係の追加を行い, VAGが生成される。

5.4 子個体の生成

提案手法では, 子個体に良質な構造を継承できるように, 非妥当なアルゴリズムに変わる突然変異を完全に発生しないようにした上で, グラフ構造を用いた突然変異を導入する. 具体的には, 突然変異後がVAGではない場合は, 再度突然変異を行う. また, 以下の8つのグラフ構造を用いた突然変異を導入し, 既存手法の子個体に良質な構造を継承できない問題に対処する. (1) 定数ノードの値の変更, (2) LPノードの初期値の変更, (3) 定数ノードを接続に置き換え, (4) 接続を定数ノードに置き換え, (5) 接続関係の変更, (6) 部分グラフの再構築, (7) NLPノードをルートとする部分グラフの再構築, (8) 予測ノードをルートする部分グラフの再構築. 突然

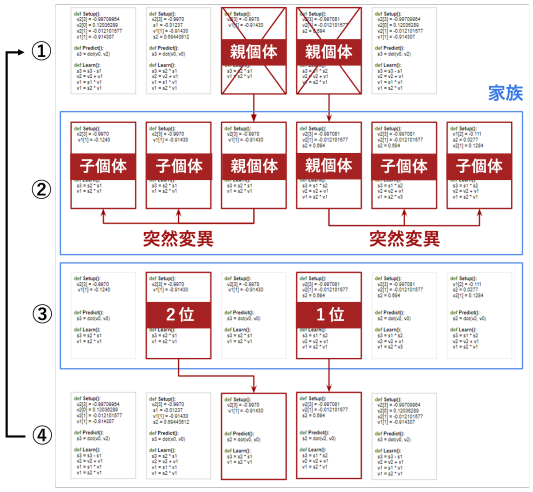

Fig. 5　MGGによる世代交代の流れ

変異を行う際には, それぞれ所与の選択確率に従って, これらの突然変異操作のいずれかを適用する.

ここでは, 既存手法の問題点を解決する上で最も重要な(6)の部分グラフの再構築の突然変異について, Fig.4のVAGを例にあげて説明する. 図において, 灰色で示した領域が変更対象となる部分木である. 部分木の突然変異では, 突然変異前後において, ルートノードの型と葉ノードは変化させず, ルートノードの命令の種類や中間ノードの命令とそのエッジを変更する. 部分グラフの再構築による突然変異は, 関数内の特定変数の計算に必要な部分命令列の変更に対応する. 既存手法では関数全体を書き換える突然変異が高頻度で発生し, 良質な部分命令列が破壊的に変更されていた. 一方, 提案手法における部分グラフの再構築は, 関数内の特定変数の計算に必要な部分のみを変更するため, 関数内の良質な部分命令列を維持した突然変異が可能となると考えられる.

5.5　集団の多様性維持

提案手法では, 集団の多様性を維持しやすくするため, RE-AutoML-Zeroと同様の方法で初期集団を生成した後に, Fig.5に示したMGGによる世代交代を繰り返し行う. 各世代交代では, STEP1で無作為に2つのアルゴリズムp_a, p_bを親個体として取り出し, STEP2でp_aを突然変異させた個体を$N_{\mathrm{children}}/2$個, p_bを突然変異させた個体を$N_{\mathrm{children}}/2$個つくり, 合計でN_{children}個の子個体を生成する. その後のSTEP3,4では, 生成したN_{children}個の子個体と親個体p_a, p_bを合わせた家族に対して生存選択で2つ個体を選択して集団に戻す. 集団サイズN_{pop}, 子個体生成数N_{children}はユーザパラメータである.

世代交代モデルをMGGに置き換えることで, 第4.3節で述べた3つの集団の多様性を低下させる要因に対処できると考えられる. MGGでは, Fig.5のSTEP1で淘汰される候補となった親個体も, Fig.5のSTEP3, 4の生存選択で集団に戻される可能性があるため, 集団内の個体が無条件で淘汰されることはない. また, Fig.5のSTEP1で無作為に複製選択するため, トーナメント選択のような強い選択圧がかかりにくいと考えられる. 加えて, STEP3, 4における生存選択で選ばれる個体が, 淘汰される候補である親個体の家族に限定されているため, 淘汰される個体と無関係な個体が次世代に引き継がれることが少ない.

さらに, 提案手法では同一の個体や類似の個体が集団内に増えることを抑制するために, 集団内の同一個体の重複排除と希少度を考慮した生存選択を行う. 重複排除は, 集団内の個体の重複を検知し, 重複している個体を削除することで, 集団内の多様性を維持することができる. 希少度を考慮した生存選択では, 定数や学習パラメータの値のみが異なるアルゴリズムを類似個体と判定し, 集団内の個体の希少度を計算し, 希少度が高い個体を優先的に選択する. これにより, 集団内の多様性を維持し, 局所最適解に収束する可能性を低減することができると考えられる.

6　実験

6.1　目的

本実験の目的は, 回帰アルゴリズムや分類アルゴリズムの探索問題をローカルPCの少ない計算リソース上で実行し, 提案手法と既存手法の探索性能を比較をすることである. 既存手法であるEstebanらの手

法RE-AutoML-Zero[3] に比べて, 提案手法のMGG-AutoML-Zero+VAGの方が, 所与の評価回数で高い適合度のアルゴリズムが得られることを確認する.

6.2 実験設定

本実験では, 回帰問題と分類問題それぞれで, 線形, アフィン, 非線形の特性を持つ問題を用いる. それぞれの問題に対して, ノイズがない場合とノイズが正規分布$N(0, 0.1)$に従う場合の2種類の問題を用意し, 合計12種類問題で実験を行う.

6.3 評価基準

本実験では, それぞれの問題設定で乱数を変えて10試行の実験を行い, 所与の評価回数までに発見されたアルゴリズムの適合度の平均値を評価基準とする. 適合度は回帰問題の場合であれば全てのタスクの検証データに対して完全に誤差なしで回帰できた場合に1, 分類問題であれば全てのタスクの検証データを正しく分類できた場合に1となる. また, 各実験における打ち切り適合度は0.999, 打ち切り評価回数は線形回帰/分類の問題では20,000回, アフィン回帰/分類の問題では2,000,000回, 非線形回帰/分類の問題では20,000,000回とした.

6.4 実験結果

実験の結果をTable.1に示す. 実験結果より, 全ての問題設定において, 提案手法は既存手法よりも高い適合度のアルゴリズムの発見に成功していることが確認できる.

7 考察

7.1 探索効率

本節では, 提案手法の探索効率がどの程度向上しているかを定量的に評価するために, 第6章の線形回帰の問題設定で, 適合度が1.00になる最適解の発見までにかかる評価回数を比較する実験を行った. 乱数を変えて, 10試行行った時の評価値の改善推移をFig.6に示す. 実験結果より, 提案手法は既存手法に比べて, 少ない評価回数で最適解を発見できていることがわかる. 定量的には,

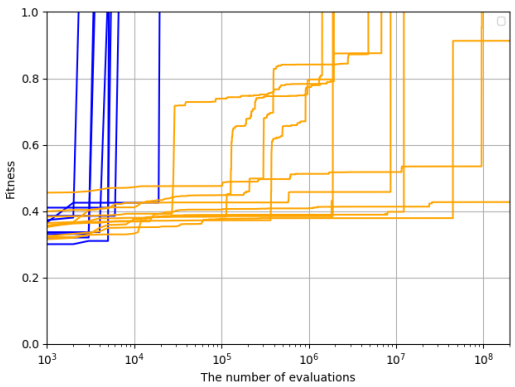

Fig. 6 線形回帰問題における最適解発見まで評価値の推移（青：提案手法, 橙：既存手法）

既存手法が平均5,399,857回の評価回数で最適解を発見したのに対し, 提案手法は平均5,270回の評価回数で最適解を発見しており, 必要な評価回数はおよそ1/1000倍になっていることがわかった.

7.2 発見されたアルゴリズム

第6章の実験で発見された非線形回帰アルゴリズムをAlgorithm.3に示す. このアルゴリズムを解釈すると, 活性化関数がReLUの1層のニューラルネットワークと類似したアルゴリズムであることとがわかる. 特に, Learn関数では, ReLU関数の導関数であるheaviside関数で勾配の計算をした上で, ニューラルネットワークの重みに対する勾配降下法を適用していると考えられる.

8 おわりに

8.1 まとめ

本論文では, Estebanらが提案したRE-AutoML-Zeroの探索効率に対する問題点を指摘した上で, その問題点に対処した手法の提案を行った. 提案手法では, グラフ構造を用いてアルゴリズムを表現する手法（アルゴリズムグラフ）とその突然変異を導入することで, 探索空間の冗長性の問題と良質な構造を子に継承できない問題に対処した. また, 集団の多様性維持に関する問題に対処するために, Minimal Generation Gap

Table 1　既存手法と提案手法の実験結果

問題		既存手法	提案手法
線形回帰	ノイズなし	0.399 ± 0.032	$\mathbf{1.000 \pm 0.000}$
	ノイズあり	0.400 ± 0.036	$\mathbf{0.931 \pm 0.006}$
線形分類	ノイズなし	0.978 ± 0.009	$\mathbf{0.980 \pm 0.009}$
	ノイズあり	0.973 ± 0.008	$\mathbf{0.972 \pm 0.005}$
アフィン回帰	ノイズなし	0.586 ± 0.256	$\mathbf{1.000 \pm 0.000}$
	ノイズあり	0.431 ± 0.159	$\mathbf{0.936 \pm 0.002}$
アフィン分類	ノイズなし	0.863 ± 0.079	$\mathbf{0.983 \pm 0.003}$
	ノイズあり	0.917 ± 0.056	$\mathbf{0.977 \pm 0.004}$
非線形回帰	ノイズなし	0.360 ± 0.059	$\mathbf{0.536 \pm 0.098}$
	ノイズあり	0.364 ± 0.058	$\mathbf{0.519 \pm 0.127}$
非線形分類	ノイズなし	0.876 ± 0.447	$\mathbf{0.961 \pm 0.011}$
	ノイズあり	0.854 ± 0.029	$\mathbf{0.918 \pm 0.016}$

Algorithm 3 提案手法で発見された非線形回帰のアルゴリズム (Setup関数は省略)

function PREDICT
    $v8 = \mathrm{dot}(m0, v0)$
    $v10 = \mathrm{maximum}(v8, v1)$
    $s1 = \mathrm{dot}(v9, v10)$
end function
function LEARN
    $s2 = s0 - s1$
    $v5 = s2 * v9$
    $v7 = \mathrm{heaviside}(v8, 1.0)$
    $v6 = s3 * v7$
    $v3 = v5 * v6$
    $v2 = s2 * v4$
    $m1 = \mathrm{outer}(v3, v0)$
    $v1 = v1 + v2$
    $m0 = m0 + m1$
end function

(MGG) [4] による世代交代モデル, 集団内の同一個体の重複排除, および希少度に基づく生存選択を導入した.

その結果, 回帰問題や分類問題の主要なベンチマークにおいて, 提案手法は既存手法よりも高い適合度のアルゴリズムの発見に成功し, 線形回帰アルゴリズム問題では, 最適解発見までに必要な評価回数が1/1000になることが確かめられた.

8.2　今後の課題

本研究の今後の課題として, ハイパーパラメータのチューニング, 高度な問題設定による大規模実験の実施, VAGの交差の実現の3つが挙げられる. 各突然変異の選択確率等のハイパーパラメータは, 探索の進捗に合わせて動的に設定することで, 探索効率の改善が期待できる. また, 本論文の実験では, 限られた命令セットで機械学習の問題設定の中でも比較的簡単な問題を扱って, 探索性能の確認を行った. その結果, 提案手法は既存手法の1000倍以上の性能を持つことが確認できたので, より大規模な問題に挑戦していきたい. また, 現状は1つのVAGに対する突然変異のみで, 複数の親個体の構造を継承することができない, そのため, VAGの交叉手法を開発し, 複数の親個体の構造を継承できるようにし, より効率的に探索を行うことが課題である.

参考文献

1) Renato Negrinho, Matthew Gormley, Geoffrey J Gordon, Darshan Patil, Nghia Le, and Daniel Ferreira. Towards modular and programmable architecture search. Advances in neural information processing

systems, Vol. 32, pp. 524–532, 2019.

2) Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V Le. Regularized evolution for image classifier architecture search. In Proceedings of the aaai conference on artificial intelligence, Vol. 33, pp. 4780–4789, 2019.

3) Esteban Real, Chen Liang, David So, and Quoc Le. Automl-zero: Evolving machine learning algorithms from scratch. In International Conference on Machine Learning, pp. 8007–8019. PMLR, 2020.

4) Hiroshi Sato. A new generation alternation model of genetic algorithms and its assessment. Transactions of the Japanese Society for Artificial Intelligence, Vol. 12, No. 2, pp. 82–91, 1997.

5) David So, Quoc Le, and Chen Liang. The evolved transformer. In International conference on machine learning, pp. 5877–5886. PMLR, 2019.

6) Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2820–2828, 2019.

7) Barret Zoph and Quoc Le. Neural architecture search with reinforcement learning. 2017.

8) Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V Le. Learning transferable architectures for scalable image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 8697–8710, 2018.