5G-04

ディープラーニングを用いた数値計算ライブラリの最適実装選択の検討

山田賢也† 片桐孝洋‡ 永井亨‡ 荻野正雄‡

名古屋大学工学部電気電子・情報工学科† 名古屋大学情報基盤センター‡

疎行列を対象とする数値計算ライブラリでは，実装選択に関するチューニングパラメタが多く存在するが，その設定の違いによる計算性能の差が非常に大きい．また，最適実装は計算を行う疎行列に依存する．しかし一般的に，各疎行列に対して実行して性能を調べる以外の方法で最適実装を見つけることは容易ではない．そこで本研究では，あらかじめ実行を行い最適実装が分かっている疎行列を訓練データとして利用し，ディープラーニングによる機械学習を行うことで最適実装を予測する手法の提案と評価を行う．このことより，最適実装について実行を行うことなく推定ができるようにし，数値計算ライブラリの使用者が専門家でなくても，許容範囲内の性能で数値計算ライブラリが利用できるようにすることを目標とする．性能評価の結果，提案手法による予測により 62.1%の精度で最適解を推定することができた．

1.はじめに

疎行列を対象とする数値計算ライブラリでは，実装選択に関するチューニングパラメタが多く存在するが，チューニングパラメタに対する最適実装の選択は，実際の実行を行う以外の方法では容易には見つけられない．疎行列は数値計算の様々な分野で取り扱われ，疎行列を対象とする数値計算ライブラリ（疎行列ライブラリ）もまた，多くの分野で利用されている．そこで疎行列ライブラリの最適な実装を，実行を介することなく最適化することができれば，多くの分野における高性能化のコストが大きく削減可能となる．

本研究では，疎行列ベクトル積（SpMV）や反復解法の専門家でなくても最適な実装方式が指定できる数値計算ライブラリの構築を目指し，人工知能による実装選択予測手法の開発を目指す．本手法により，専門家による実装選択と同等，もしくは，許容範囲内の性能で実装が選択できる数値計算ライブラリの構築を目指す．

本論文の構成を以下に示す．2 章で関連研究と性能予測の問題設定の説明を行う．3 章で既存手法と提案手法の説明を行う．4 章で性能評価を行い，最後に 5 章でまとめを行う．

2.関連研究と問題設定

2.1 関連研究

疎行列を対象とする数値計算ライブラリでは，多くの場合 SpMV を用いるが，SpMV は様々な実装が存在する．SpMV の実装選択は，数値計算ライブラリの実装選択の一部分と見なせるため，SpMV の実装選択を対象とする機械学習を用いた自動チューニングの既存研究が存在する．例えば，Cui らの手法[1]では，疎行列を変換して生成した特徴画像とディープラーニングを用いて最適な SpMV 実装選択を試みている．ディープラーニングを用いる理由としては，特徴が自動的に抽出されるという点があげられる．これにより最適実装選択に必要な特徴を定義する必要がなくなり，疎行列を画像化したデータをそのまま学習に用いることができる．ただし，ディープラーニングにおいて画像を扱う場合には，CNN（畳み込みニューラルネット）を用いるのが主流であるが，入力として用いるためには画像サイズを固定する必要がある．そのため，疎行列に関する画像の生成方法もまた研究対象となる．本研究では，対象が SpMV であるか，数値計算ライブラリであるかの違いはあるが，Cui らの手法を既存手法として利用し，提案手法の比較と評価の面で利用する．

2.2 使用する疎行列と数値計算ライブラリ実装

今回の研究で用いた数値計算ライブラリは，疎行列を対象とした数値計算ライブラリ *Xabclib* を利用する．その中でも連立一次方程式の数値解を求める反復法である，GMRES 法を実装した *Xabclib_GMRES* を利用する．*Xabclib_GMRES* には様々な実装選択に関するチューニングパラメタが存在するが，今回は前処理 6 種類，SpMV 実装 4 種類を対象とし，これらの組み合わせである全 24 通りの実装に対して最適実装の予測を行う．

利用する疎行列は，フロリダ疎行列コレクション[2]の正方行列のうち，*Xabclib_GMRES* で計算を行って，実行時間制限 600 秒，リスタート回数制限 1000 回の条件のもと，いずれかの実装で解が収束した約 1000 個の疎行列とする．この際，右辺ベクトルは解となるベクトルの要素が全て 1.0 となるように設定した．対象とする行列は分野を問わず，解が収束したものを全て使用し，次元数の範囲は約 200 から 30 万，非ゼロ要素割合は 56%以下のものである．また，本研究では実行環境として名古屋大学情報基盤センターのスーパーコンピュータである，Fujitsu PRIMEHPC FX100 を使用した。

3.既存手法，提案手法の詳細，および学習設定

3.1 既存手法

前述のとおり Cui らによる手法を既存手法として扱うが，この手法では疎行列の画像変換において固定サイズのグレースケール画像への変換を行っている．この際，行列サイズの情報を色として残している(図 1 左)．

これにより，行列の構造情報と行列サイズの情報を残しながら，固定サイズの画像として扱えるので CNN の入力として使用できるようになる．

3.2 提案手法

本研究では，最適実装を求める対象は SpMV ではなく数値計算ライブラリである．そのため，前処理の実装予測を含んでおり，疎行列の非ゼロ要素の値が消えてしまう既存手法では高い精度が得られないことが推測される．そこで提案手法ではグレースケール画像ではなく，RGB3

Examination on selection of optimal implementation with deep learning for numerical computation libraries

† KENYA YAMADA, Electrical and Electronic Engineering and Information Engineering, School of Engineering Nagoya University

‡ TAKAHIRO KATAGIRI, TORU NAGAI, MASAO OGINO, Information Technology Center, Nagoya University

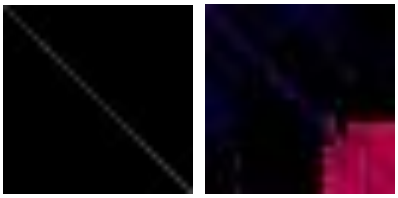

図 1:xingo_afonso_itaipu 特徴画像

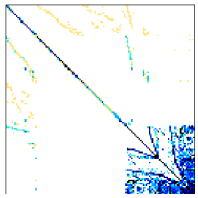

図 2:xingo_afonso_itaipu 行列画像[2]

色からなるカラー画像を用いて特徴画像を生成する．これにより**グレースケール画像では 1 種類しか残すことのできなかった情報を 3 種類まで増やすことが可能**となり，非ゼロ要素の値の情報を残すことができる．

具体的には，行列サイズを青，疎度を緑，非ゼロ要素の値を赤の色として，それぞれ挿入する．ここで赤の値の挿入についてであるが，疎行列によっては非ゼロ要素の値の最大値と最小値の差が大きいために色の変化が出にくい場合があることが判明した．そこで非ゼロ要素の値の最大値と最小値の差が大きい場合は，下記式(1)に従い，値をx_{old}からx_{new}へと変換する。

$$x_{new} = \log_2|x_{old}| \quad (1)$$

これにより非ゼロ要素の値域が減少し、色が出やすいように調整される．図 2 のフロリダ行列コレクションから引用した本来の行列の画像と，図 1 左の特徴画像を比較すると，非ゼロ要素の密度によっては非零要素が消えてしまっている部分があるのがわかる．これは，リサイズの際のサイズ比(今回の例では 13250:32)が非常に大きいため，生じると考えられる．

そこで OpenCV の resize()関数を使うのではなく，元行列を 32×32 等分に分割し，その中に非ゼロ要素が存在すれば特徴画像の対応する座標に画素値を与える方法を考える．これにより非ゼロ要素の密度が小さい場合であっても，その要素を落とすことなく特徴画像に反映させることができる(図 1 右)．ただしこの場合，非ゼロ要素の密度が本来のものより濃くなるという問題点がある．

3.3 学習体系

既存手法，提案手法ともに用いる機械学習モデルはディープラーニングとする．使用したディープラーニングのフレームワークは TensorFlow[4]である．

今回作成した CNN は，畳み込み層とプーリング層のペア 2 組，全結合層 2 層，ソフトマックス層からなる構造である．使用するデータは，3.1 節及び 3.2 節で説明した手法で作成した特徴画像と，実際に実行して得られた最適実装の組を用意した．ただし，3.1 節の手法においては，元行列のサイズによっては resize()関数で変換が行えない場合があり，データ数が減少する．最後に，それらを訓練データとテストデータにデータ数の比が 4:1 となるよう分割した．

学習はミニバッチサイズを 50 とし，ミニバッチサイズ分の学習を 1 ステップとして 200 ステップ行う．1 ステップ毎の精度を測定し，その中で最もテストデータの実装予測の精度が高かった際の CNN モデルを，そのデータセットのおいての学習結果として扱うものとした．これは今回のデータ数を鑑みて過適合の発生が考えられるための処理である．

4.結果と評価

表 1，表 2 にそれぞれ特徴画像 1，特徴画像 2 を用いて性能予測を行った場合の結果をまとめる．ただしこの表における最適正答率は SpMV 及び前処理の予測両方が正解した場合を示す．最適実行時間は，最適パラメタによる実行時間の合計，予測実行時間は各手法による推定パラメタによる実行時間の合計，実行時間誤差はそれらの差を示す．非収束は，推測した実装による *Xabclib_GMRES* での実行が収束しないか，あるいは収束が非常に遅い場合を示す．

表 1，表 2 から，提案手法を用いた場合の方が前処理の正答率が高くなり，全体での最適な実装予測率も高くなることがわかる．提案手法を用いた場合の予測では，62.1%の精度を得られた。

また，最適な実装を選べなかった場合でも収束する実装が選べた場合は，実行時間の差もそれほど大きいわけではない．しかし，収束しない実装を選択している場合は，既存手法で 31%，提案手法で 13%程存在しているため，必ずしもこの実行時間の差のみが誤差の全てを表しているとは言えないことに注意する．

表 1:特徴画像 1(図 1 左)による最適実装予測結果

テストデータ数	最適正答率(%)	SpMV正答率(%)	前処理正答率(%)	非収束実装率(%)
188	42.6	72.3	56.4	31.4
訓練データ数	最適実行時間(s)	予測実行時間(s)(除:非収束分)	最適実行時間(s)(除:非収束分)	実行時間誤差(s)(除:非収束分)
755	2925.474	836.285	831.456	4.828

表 2:特徴画像 2(図 1 右)による最適実装予測結果

テストデータ数	最適正答率(%)	SpMV正答率(%)	前処理正答率(%)	非収束実装率(%)
211	62.1	74.1	78.2	13.3
訓練データ数	最適実行時間(s)	予測実行時間(s)(除:非収束分)	最適実行時間(s)(除:非収束分)	実行時間誤差(s)(除:非収束分)
847	4016.39	3229.959	3200.432	29.527

5.まとめ

評価結果から数値計算ライブラリにおいてディープラーニングを用いた最適実装予測を行う場合に 3 色画像を使用することは有効であると考えられる．また，提案手法で最適実装を選択できなくても，収束している実装を選択できている場合であれば，収束している疎行列のみに対して最適パラメタによる実行時間と本手法による推定パラメタによる実行時間の合計の差は高々1%程度である．この観点で，専門家でなくてもある程度の性能を得られるような実装選択を行えていると考える．

ただし，今回使用している行列の種類が少ない，最適実装の種類の分布が偏っているなど，学習データや，CNN の構造，特徴画像の生成法などの改善点がある．これらの改善を行った場合の再評価は今後の課題である．

謝辞

Xabclib の利用に関し，日立製作所の櫻井隆雄氏に感謝いたします．本研究の一部は，科学技術研究費補助金基盤研究(B)「機械学習技術の活用による職人的プログラミングの知能化」(課題番号:16H0822)による．

参考文献

[1] Hang CUI, Shoichi HIRASAWA, Hiroyuki TAKIZAWA and Hiroaki KOBAYASHI, *A Code Selection Mechanism Using Deep Learning*, 2016 IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip, pp. 385-392.

[2]The SuiteSparse Matrix Collection, https://www.cise.ufl.edu/research/sparse/matrices/, 2017

[3] OpenCV, http://opencv.org/, 2017.

[4]TensorFlow-an open-source software library for Machine Intelligence, https://www.tensorflow.org/, 2017.