職人的HPCプログラミングにおける機械学習技術の活用

Use of machine learning techniques for performance-aware HPC programming

東北大学サイバーサイエンスセンター
滝沢 寛之

我々は、従前では人間の判断が必要とされてきた職人的プログラミング作業（の一部）を、機械学習技術によって代行する可能性を探ってきた。職人的HPCプログラミングにおいて、プログラマが何らかの判断を行うために必要なアプリケーションやコードの特徴を事前に数値化しておくことは一般的に困難である。このため、我々は機械学習モデルの表現学習能力に着目し、それを有効利用することで職人的HPCプログラミングの作業の一部を機械学習で代行することを考えてきた。本稿では、職人的なHPCプログラミングにおける機械学習技術の活用を目指し、疎行列計算カーネルの実装選択およびコンパイラオプション選択に機械学習を適用した研究成果を概説する。また、機械学習技術をHPCプログラミング分野に効果的に適用するための要件について議論する。

1．はじめに

ポストムーア時代を迎えつつある現在、高性能計算（HPC）システムの性能を今後さらに高めていくためには、システムの構成がより一層の大規模化･複雑化をせざるを得ない。ポストムーア時代前夜の現在ですらHPCシステムはすでに大規模で複雑であり、その性能を正確にモデル化することは難しい。このため一般的に、アプリケーションコードに加えた変更がそのアプリケーションの実行性能に与える影響を事前に予測することは困難である。その結果、例えばコンパイラの自動最適化機能による実アプリケーションの性能改善には限界があり、研究レベルでは様々な提案もなされているが、現場レベルでは熟練のプログラマが経験と勘に頼りながら経験的にコードを修正して性能を高めていることが多い。そのような**職人的プログラミング**では、しばしば試行錯誤によって効果的なコード修正が見出されているため、その作業にはかなりの時間と労力を要する。冒頭でも述べた通り将来のシステムの大規模化と複雑化が避けられないことを考えれば、職人的プログラミングに求められる時間や労力も今後さらに増大することが危惧される。少子高齢化が進む我が国ではこの技術的課題の「人海戦術」による解決は困難であり、今後も安定的にHPC技術を活用していくためには、職人的プログラミングの労力軽減が喫緊の課題であるといえる。

一方、そのようなアルゴリズム化が難しく人間の判断に頼らざるを得ない分野で、機械学習技術が近年目覚ましい成果を挙げている。例えば、画像認識分野では、機械学習が人間を大きく上回る認識率を達成できることが報告されている［1］。

このような背景から我々は、従前では人間の判断が必要とされてきた職人的プログラミング作業（の一部）を、機械学習技術によって代行する可能性を探ってきた。その第一歩として、多くの科学技術計算において重要な計算カーネルである疎行列ベクトル積（Sparse

matrix-vector multiplication, SpMV）を対象として考え、複数のSpMV実装が利用可能な場合に、それぞれの疎行列に対して最適な実装を選択する機械学習モデルを構築した［2］。また、そのアプローチは反復解法アルゴリズムにおける前処理選択にも発展的に応用されている［3］。さらには、疎行列以外への応用として、コンパイラオプションの選択問題においても機械学習技術の利用を検討してきた［4］。本稿では、主にSpMV実装選択の研究成果を概説する。

2．画像認識分野における機械学習の成功と表現学習

人工知能（AI）や機械学習といった技術への期待や注目度が高まって久しい。「第3次AIブーム」と呼ばれる現在の盛り上がりは、機械学習モデルの一種であるConvolutional Neural Network（CNN）がImageNetという画像データベースの認識において既存技術を大幅に上回る結果を達成［5］したことに始まるともいわれている。そのエポックメイキング的な文献［5］で、6000万個ものパラメータをもつ大規模CNNの学習のために、当時すでにHPCシステムの重要な構成要素の一つとなっていたGraphics Processing Unit（GPU）を利用し、学習処理の効率的な実装についても議論されていることは特筆に値する。現代の機械学習は大規模HPCシステムの有効利用を前提に発展してきた技術であり、さらには膨大な量のデータにも従来よりもはるかに低コストで手軽にアクセス可能となったことで飛躍的に実用化が進んできた。

ImageNetにおけるCNNの華々しい成功の後、画像認識分野では大きな変化が起こった。従来の画像分類問題においては、画像の特徴を事前に何らかの数値で表現（つまり数値化）し、その値に基づいて画像を分類するアプローチが主流であった。画像の特徴を数値で表現する**特徴量**を定義することを**特徴量選択**と呼ぶ。特徴量選択は、画像分類で好成績を収めるための最も重要で、最も難しい技術的課題であった。一方CNNは、画像分類のために有用な特徴量も画像データから学習によって習得できることが知られている。画像を認識するための規則性だけではなくて、その認識に必要な特徴量まで画像データから習得する能力は、**表現学習**（representation learning）と呼ばれている［6］。表現学習能力は、CNNによる画像分類の実用性を飛躍的に高め、その適用可能な範囲を広げる大きな要因となってきた。

3．表現学習能力の利用による疎行列計算カーネルの実装選択

画像分類において特徴量選択が困難であったのと同様に、職人的HPCプログラミングにおいても、プログラマが何らかの判断を行うために必要なアプリケーションやコードの特徴を事前に数値化しておくことは一般的に困難である。このため、我々はCNNの表現学習能力に着目し、それを有効利用することで職人的HPCプログラミングの作業の一部を機械学習で代行することを考えてきた。その第一歩として、計算対象となる疎行列の特徴に基づいてプログラマが行ってきた判断を、機械学習モデルで代行する方法を研究した。疎行列計算に複数の実装が利用可能なとき、多くの場合、最も高い性能を達成できる実装は入力となる疎行列の特徴によって変化する。このため、疎行列に合わせて計算カーネルの実装を選択する問題を考え、その選択を機械学習で代行する検討を行った。

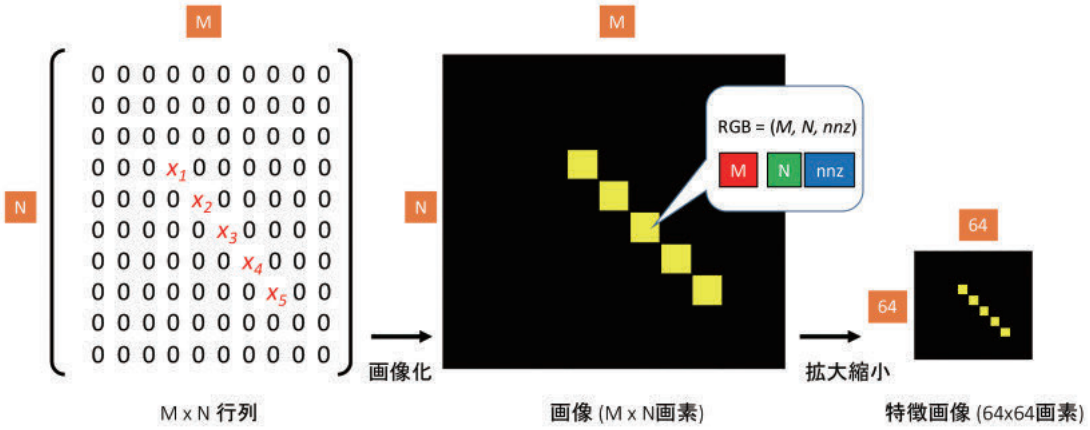

図１　SpMV実装選択における疎行列の特徴画像への変換

画像分類問題への変換

疎行列計算の実装選択問題の関連研究として、疎行列の持つ特徴をいくつかの数値で表し、複数あるSpMV実装の中から最も性能の高いものを機械学習モデルで予測する研究が挙げられる［7］。これに対して、我々はCNNの持つ表現学習能力を利用することを考えてきた。CNNは特に画像認識分野で目覚ましい成功を収めている機械学習モデルであるため、計算カーネル実装選択問題を画像分類問題に変換することができれば、CNNによる高い認識精度の達成を期待できる。このため本研究では、まず疎行列全体を１枚の画像として表現し、行列要素が０だった位置の画素値は（R, G, B）＝（0, 0, 0）とし、非ゼロ要素に対応する位置だけに何らかの色を付ける。以下、疎行列を変換することで生成される画像を**特徴画像**と呼ぶ。疎行列を特徴画像に変換する方法を図１に示す。このようにして疎行列内の非ゼロ要素の分布を画像化することで、計算カーネル実装選択問題を画像分類問題に変換することができる。

疎行列を特徴画像として表現する際の検討課題として、行列サイズの表現方法が挙げられる。ある与えられた疎行列に対する最適な計算カーネル実装は、ほとんどの場合、その疎行列のサイズに強く影響される。例えば、比較的小さな疎行列が計算対象の場合には、データ並列性が低いことやキャッシュの影響などによってCPUで実行したほうが速い傾向にある。一方、大きな疎行列の場合には並列度が高いなどの理由によってGPUで実行したほうが速い傾向にある。すなわち、疎行列サイズに依存して適切なSpMV実装は変化する。しかし、標準的なCNNでは入力層のノード数が事前に決められており、画像全体をそのまま入力することを考えるとすべての入力画像を一定のサイズに正規化することが望ましい。このため、本研究ではCNNへの入力画像を拡大縮小によってある一定サイズにすることを前提とし、元の行列サイズを画素値として画像内に埋め込むことで、行列サイズという元の疎行列が持っていた重要な特徴が画像化された後にも画像の特徴として残るようにしている。

一般的に画像の各画素の色は光の３原色に対応する３つの数値、あるいは透明度も含めた４つの数値で表現されている。このため、特徴画像の各画素に元の疎行列の幅と高さを保持しても、まだ１～２個の数値を各画素に埋め込むことが可能である。この数値として埋め込むべき特徴量は、実装選択したい計算カーネルの種類によって異なる。本研究では予測精度向上に有効な値を経験的に調査し、SpMV実装選択［2］においては、非ゼロ要素数が極めて重要であることから、それを特

徴量の１つとして非ゼロ要素に対応する画素の色に用いている（図１）。また、反復解法アルゴリズムの前処理選択［３］においては疎行列の要素の値そのものが影響を与えることから、行列要素の値を特徴量の１つとして対応する画素の色に用いている。

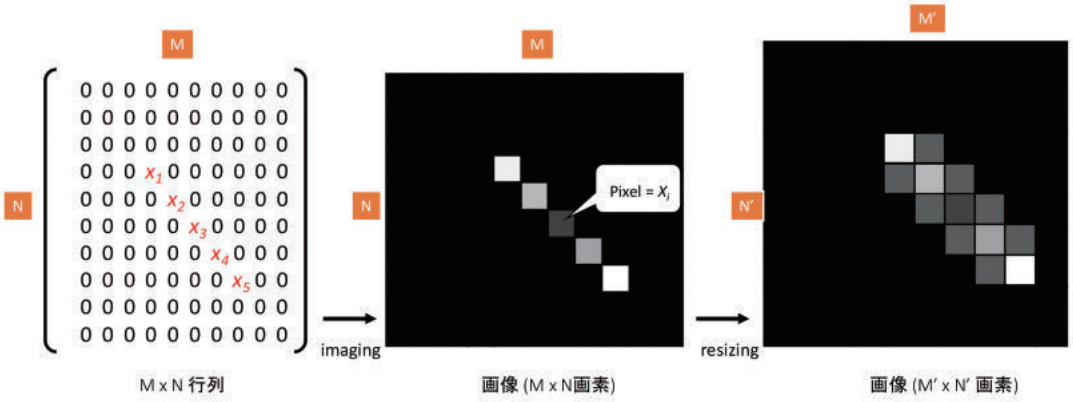

図２　SpMV実装選択における学習データ生成

学習データ生成

上述のように、SpMV実装選択問題の場合、適切な実装は行列サイズと非ゼロ要素分布の両方に依存している。例えば、行列サイズが同じ場合でも、非ゼロ要素の分布が変化することで適切な実装も変化する。同様に、非ゼロ要素分布が同じ場合でも、行列サイズを変える（非ゼロ要素の分布はサイズ変更の前後で相似）ことで適切な実装は変化する。このため、図２のように、すでに得られている疎行列を画像データとして扱い、それを拡大縮小することで新たな疎行列を生成し、CNNの学習に利用可能な学習データを生成することができる。実際のアプリケーションから多種多様な疎行列を集めてくる労力は大きいが、このように画像の拡大縮小によってさまざまな疎行列を人工的に生成するのは容易であり、その結果として容易に学習データ数を十分な数まで増やすことができる。

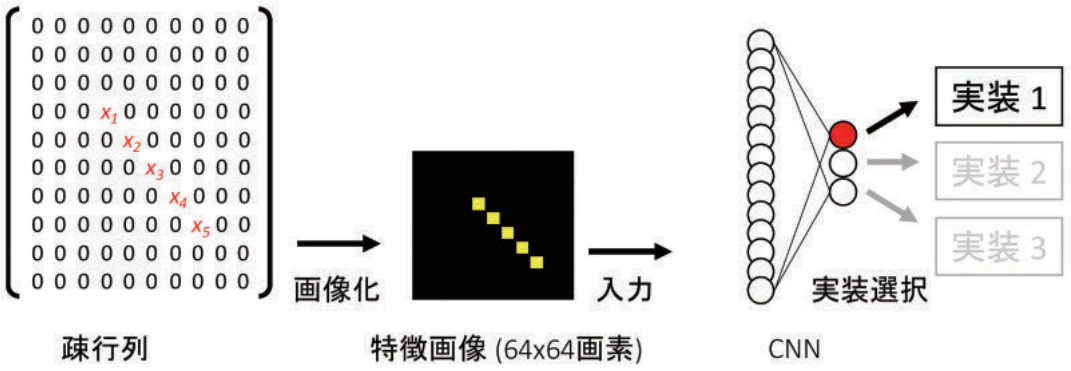

図３　CNNによるSpMV実装選択

評価と考察

本稿では、図３に示すように疎行列を64×64画素の特徴画像へと変換し、CNNでSpMV実装選択することで提案手法の有効性

を議論する。CNNの実装にはCaffe［8］を用いる。CNNのネットワーク構造としてまずはLeNet［9］を基準とし、試行錯誤することでネットワーク構造を経験的に選択した。

本評価では、CPUとしてIntel Core i7-3770、GPUとしてNVIDIA GeForce GT430を搭載するシステムでSpMVを実行するものと仮定する。以下の3つのSpMV実装を考え、当該システム上で最も高い性能を達成できる実装を選択する問題をCNNに学習させる。

CPU_COO：COO形式の行列にSpMVをCPU実行する実装
GPU_CSR：CSR形式の行列にSpMVをGPU実行する実装
GPU_HYB：HYB形式の行列にSpMVをGPU実行する実装

本評価では、フロリダ大学の疎行列集［10］から行サイズと列サイズがともに0以上20,000以下の行列を877個を選択し、その中からランダムに選ばれた670個を学習データ、207をテストデータとして使用する。さらに670個の学習データを拡大縮小して新たな学習データを生成することで、学習データ数を4,811個まで増加させて評価を行う。学習データに含まれる各疎行列に対して、3つの実装のうち最も性能が高い実装を事前に調査し、その実装に対応する出力値が大きくなるようにCNNの学習を行う。

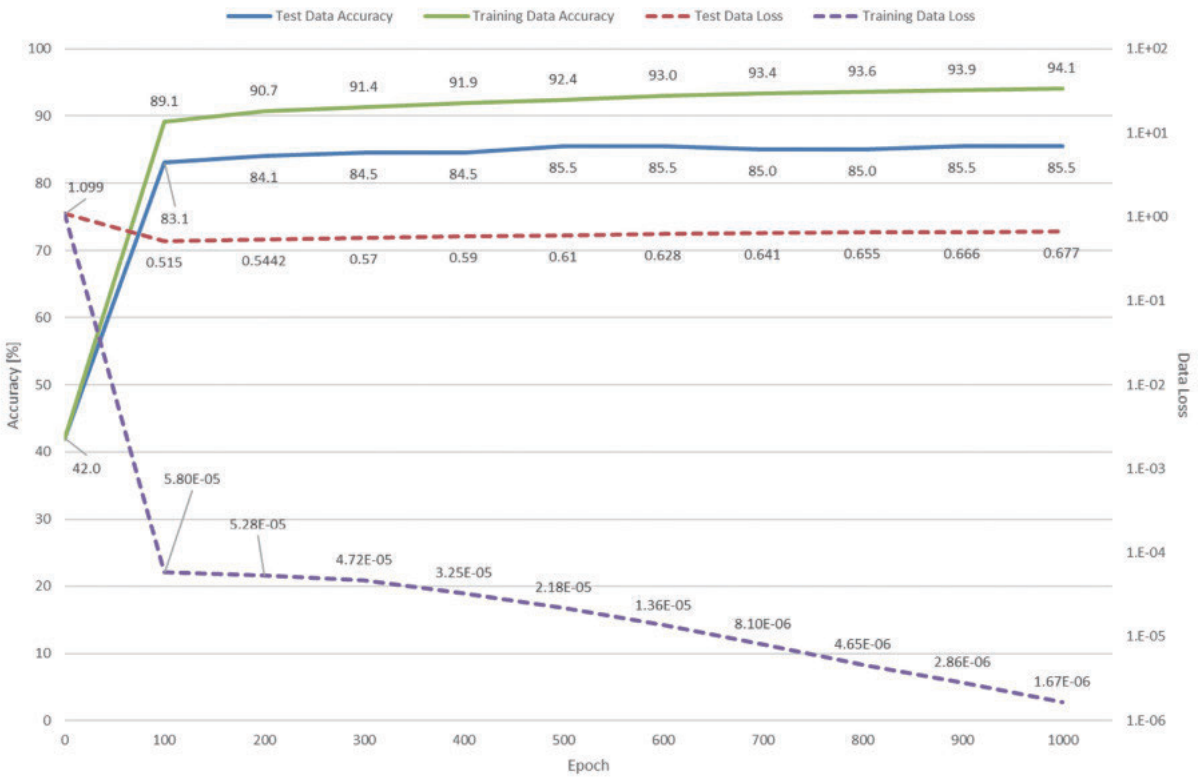

図4　学習による精度向上および誤差減少の様子

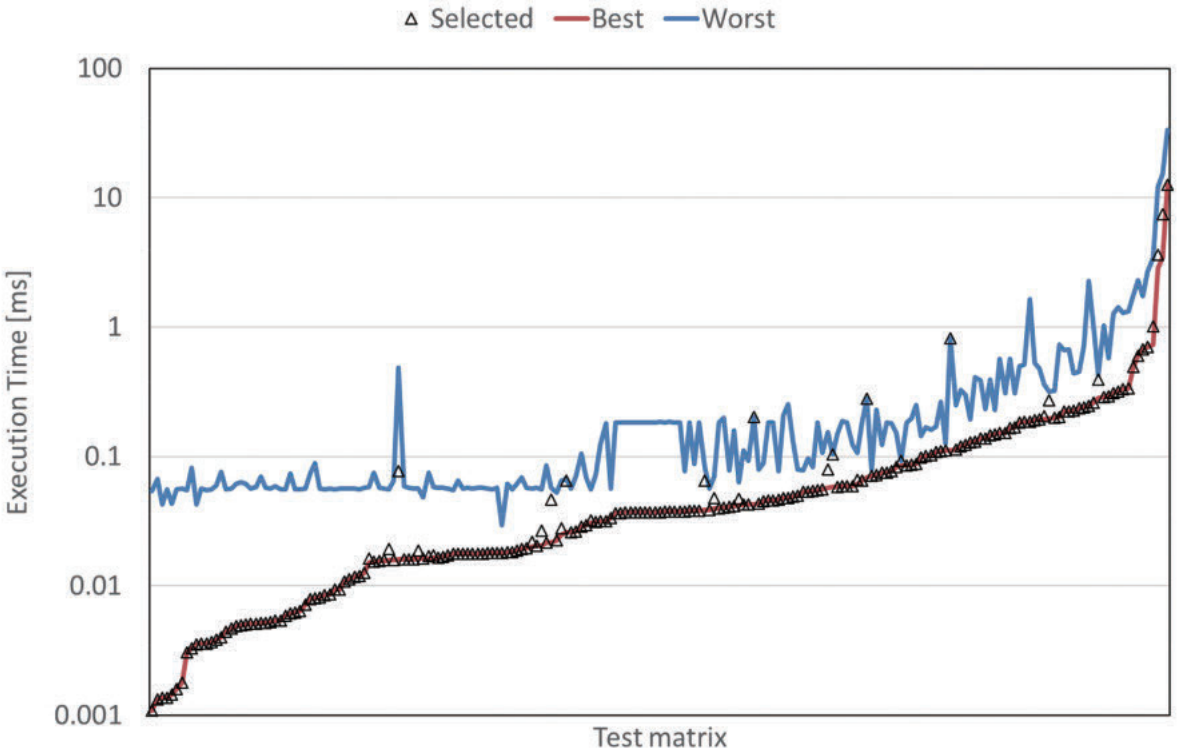

図 5　本手法により選択された実装の実行時間

学習処理が進むにつれて学習データに対する誤差（training loss）が減少する様子、およびそれに伴ってテストデータに対する誤差（test loss）も減少する様子を図 4 に示す。学習データおよびテストデータの疎行列に対して、最良の実装を選択する比率（正答率、Accuracy）も示されている。この結果から、学習が進むにつれてCNNが学習データの各疎行列に対する適切なSpMV実装を選択できるようになっていること、および学習データに含まれない疎行列に対しても適切なSpMV実装を選択できるようになっていく様子を見ることができる。このことから、CNNは特徴画像からSpMV実装を選択するために有用な特徴量を学習により習得し（表現学習）、それに基づいて適切な実装を徐々に選択できるようになることが分かる。

テストデータに含まれる疎行列をサイズに基づいて昇順にソートし、それぞれの疎行列に対して提案手法で選択された実装の実行時間（Selected）を図 5 に示す。同図では、それぞれの疎行列に対する最良の実装（実行時間が最も短くなる実装）および最悪の実装（実行時間が最も長くなる実装）の実行時間も、それぞれBestおよびWorstで示している。この結果から、大半の疎行列に対して提案手法は最良の実装を選択できていること、および最良の実装と最悪の実装との実行時間差が小さいときに選択を誤る傾向があることが分かる。同様の評価実験を繰り返し行い、安定して85％程度の精度で最良の実装を選択できることが明らかになった。図 5 の結果では、85. 1％の疎行列に対して最良の実装を選択できている。

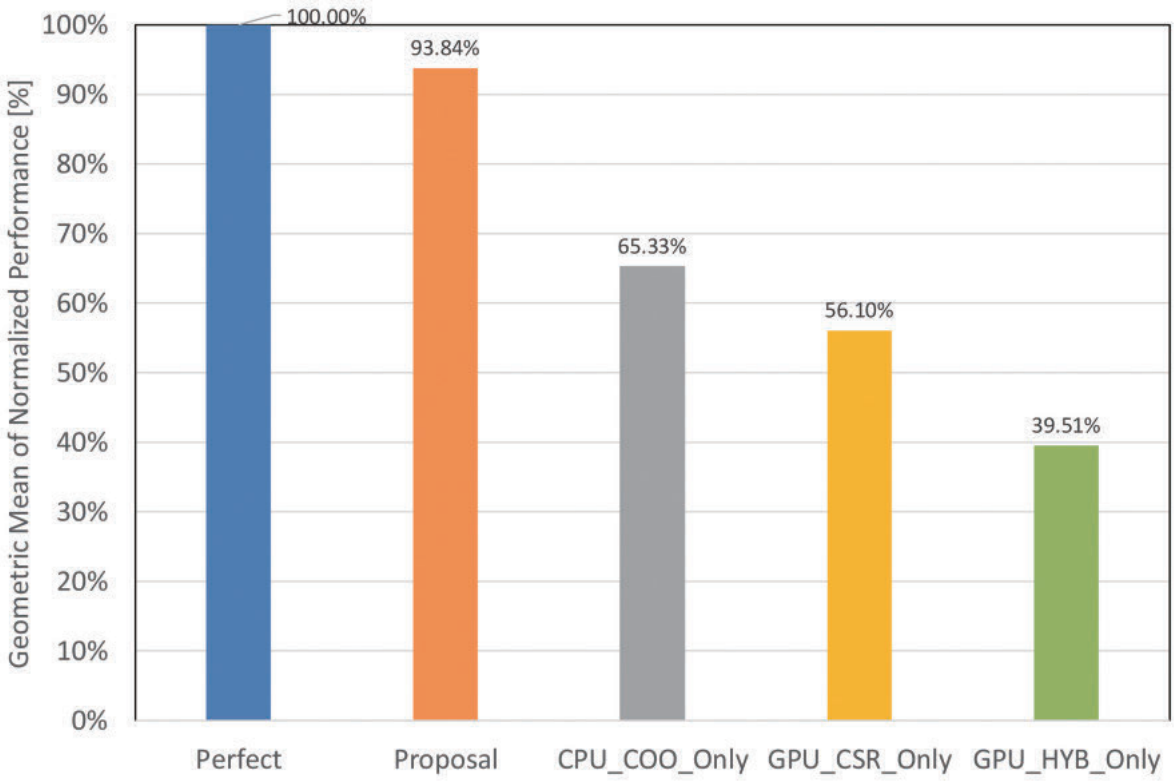

図 6　本手法による平均速度向上率（性能上限で正規化）

次に、本手法による実装選択が常に最良の実装を選択した場合と比較することで、本手法による性能向上率を定量的に議論する。この議論のために以下の指標を用いる。すなわち、CNNによって選択された実装と最良の実装との性能比の幾何平均を用いて、本手法による性能向上率を評価する。

$$G = \left(\prod_{i=1}^{N} Best_i/Selected_i\right)^{1/N}$$

ここでNはテストデータの総数であり、テストデータiの疎行列に対する最良の実装による実行時間および選択された実装の選択時間はそれぞれ$Best_i$および$Selected_i$である。CNNが最良の実装を選択した場合、$Best_i/Selected_i = 1$になり、それ以外の実装を選んだ場合には$Best_i/Selected_i < 1$となる。このため、すべての疎行列に対して最良の実装を選んだ場合にのみG＝１となり、実行時間が最短となる。幾何平均Gが１に近い値を示せば、提案手法によって選択された実装の実行時間は、最短実行時間に近いといえる。幾何平均の値は、選択された実装の平均性能向上率（最良の実装の性能を基準、つまり100％とする）を表している。幾何平均の値を図６にパーセント表記で示す。常に最良の実装を選択する場合（Perfect）と比較して、提案手法は約94％の性能を達成している。常に同じ実装を使い続けた場合と比較すると28～54％程度の性能向上を達成している。このことから、CNNは疎行列の特徴画像を表現学習し、適切な実装を選択できていることが性能向上率の面からも示され、本手法の有効性が示された。

４．職人的HPCプログラミングに機械学習技術を活用するための要件

コンパイラは、内部的に様々なコード最適化を行っている。多くの場合、コンパイラの最適化レベルを指示するコンパイラオプショ

ンフラグを指定することで、コードに対して適切な最適化技法を自動的に適用する。しかし、より詳細な最適化技法を指示することで、さらに高い性能を実現できることもある。ただし、適切なコンパイラオプションを選択することは、適切なコンパイラ最適化処理を指示することと同じであり、熟練のプログラマであっても試行錯誤なしに最適なコンパイラオプションを言い当てることは難しい。

コンパイラオプション選択問題の関連研究として、プログラムの性能プロファイル情報から適切なコンパイルオプションを機械学習モデルにより予測する研究が挙げられる[11]。しかし、実用上の問題として、多様な性能プロファイル情報を得るためには膨大な数のソースコードを収集する必要がある。そこで、我々は性能プロファイル情報に加えて、ソースコード自体のコード構造に関する情報も利用することで、少ない学習データでも機械学習モデルによる予測精度を高める研究も行っている［4］。図 7 に示すようにTree-Based Convolutional Neural Network（TBCNN）と呼ばれるCNNの一種を利用し、コードの抽象構文木（Abstract Syntax Tree）から特徴量ベクトルへの変換を学習により獲得する手法を検討した。すなわち、本手法では、ASTから有用な特徴量への変換を表現学習することを期待している。

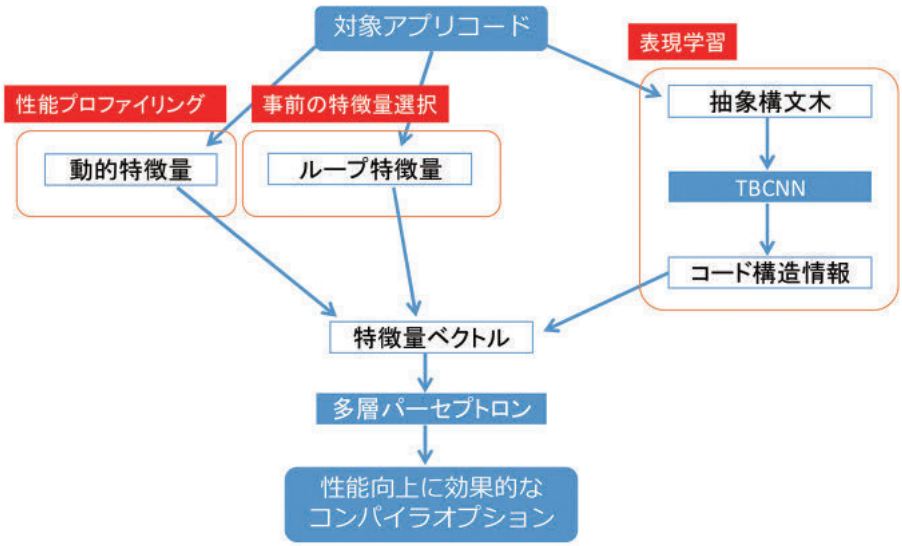

図 7　TBCNNによるコード構造に関する特徴の数値化

評価結果から、TBCNNにより出力される特徴量ベクトルを用いてコンパイラオプション選択を行うことで、性能プロファイルを使う場合とほぼ同等の予測精度を達成できることが示されており、TBCNNの表現学習能力によってコード構造情報の特徴量を学習できることが明らかになった。しかし、類似する構造をもつ学習データ数が少ないコードに関しては、コード構造情報を利用することで予測精度がかえって低下する場合があることも明らかになった。すなわち学習データ数が少なすぎる場合、その限られた学習データから他の特徴（例えばコード構造の特徴）を取り出したとしても、それを学習することで機械学習の予測精度が改善するとは限らないことを示している。

SpMV実装選択では機械学習が効果的に利用されているのに対して、コンパイラオプション選択では多くの課題が残されている。これら 2 つの事例で機械学習の有効性が大き

く異なった理由として、少なくとも以下の2点が挙げられる。

（1）SpMV実装選択問題では、疎行列における非ゼロ要素の分布と行列サイズが適切な実装の選択に大きな影響を与えている。このため、非ゼロ要素の分布形状やサイズの異なる疎行列を人工的に生成して学習データに用いることで、学習データ数を増やすことができる。一方、コンパイラオプション選択問題では、予測精度向上に有効なソースコードを意図的に生成することはできていない。このため、予め用意したソースコードのみを用いて学習を行っており、学習データ数が十分ではない。

（2）SpMV実装選択問題は画像分類問題に変換されており、画像分類問題を解くためのCNNのネットワーク構造も広く研究されている。一方で、TBCNNによってコード構造の特徴を学習する問題は、画像分類問題と比較すると十分に研究されているとは言い難く、成功裏に行うためのネットワーク構造も明らかではない。

前者より、CNNの表現学習能力には有用な特徴量の自動検出を期待できるものの、多数の有用な学習データを効率よく収集するためには、有用な特徴量を事前にある程度把握しておく必要があることがわかる。機械学習には、いわゆるビッグデータと呼ばれるような大量の学習データが必要不可欠である。このため、機械学習モデルをブラックボックスとして扱って盲目的に大量のデータを集めるのではなく、機械学習以前のアプローチと同様に、事前に対象問題を十分に分析することが機械学習の利用においても成功への近道となる可能性が高いと言える。

後者は、機械学習を用いることで熟練のプログラマの経験と勘に頼っていた部分を自動化できる可能性がある一方で、機械学習モデルの設計に熟練者の経験と勘が必要となる懸念を示している。例えば、CNNは層状のネットワーク構造となっており。層の数や各層のユニット数など、様々なパラメータを事前に決める必要がある。そのように学習以前に定義しなければならないパラメータは**ハイパーパラメータ**と呼ばれ、その適切な決定方法は未確立であることから、多くの場合には熟練者が試行錯誤で決定している。そこで我々は現在、ベイズ推定に基づいてハイパーパラメータの調整を実行時間の観点から効率化する研究も行っている［12］。

5．まとめ

本稿では、職人的なHPCプログラミングにおける機械学習技術の活用を目指し、疎行列計算カーネルの実装選択およびコンパイラオプション選択に機械学習を適用した研究成果を紹介した。また、機械学習技術をHPCプログラミング分野に効果的に適用するための要件について議論した。本稿で紹介したSpMV実装選択のように、職人的HPCプログラミングにおける種々の問題をすでに機械学習の利用が成功している問題に変換することにより、その問題を機械学習で解決できる可能性がある。しかし、HPCプログラミング分野で膨大な数の学習データを用意できる問題は稀であり、効率的な収集のためには対象問題を十分に分析する重要性が示された。

職人的HPCプログラミングと同様に、機械学習の利用においても熟練者の経験と勘に頼らなければならない。ただし、すでに数値化されているハイパーパラメータの調整であるため、人的労力ではなく計算コストの問題に置き換えて考えることが可能である。このことが機械学習技術を職人的HPCプログラミングに適用することで得られる、最も重要な利点と考えている。

謝辞

本研究の一部は、次世代領域研究開発事業量子アニーリングアシスト型次世代スーパーコンピューティング基盤の開発、科研費基盤研究（B）16H02822、挑戦的萌芽研究15K12033の支援を受けている。

参考文献

［1］ He et al. "Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification." In: Proceedings of the IEEE international conference on computer vision, pp.1026-1034, 2015.

［2］ Cui et al. "A machine learning-based approach for selecting SpMV kernels and matrix storage formats," IEICE Transactions on Information and Systems, Vol. E101-D, No. 9, pp. 2307-2314, 2018.

［3］ Yamada et al. "Preconditioner auto-tuning using deep learning for sparse iterative algorithms," 2018 Sixth International Symposium on Computing and Networking Workshops, pp. 257-262, 2018.

［4］ Kawarabatake et al. "Use of code structural features for machine learning to predict effective optimizations," 2018 IEEE International Parallel & Distributed Processing Symposium Workshops, pp. 1049-1055, 2018.

［5］ Krizhevsky et al. "ImageNet classification with deep convolutional neural networks," Advances in neural information processing systems, pp. 1097-1105, 2012.

［6］ LeCun et al., Deep Learning, Nature, Vol. 512, No. 7553, pp.436-444, 2015.

［7］ Muralidharan et al. "Nitro: a framework for adaptive code variant tuning," 2014 IEEE International Parallel & Distributed Processing Symposium, pp. 19-23, 2014.

［8］ Yangqing et al. "Caffe: Convolutional architecture for fast feature embedding," Proc. 22nd ACM International Conference on Multimedia, pp.675-678, 2014.

［9］ LeCun et al. "Gradient-based learning applied to document recognition," Proc. IEEE, vol.86, pp.2278-2324, 1998.

［10］ Davis et al. "The university of Florida sparse matrix collection," ACM Transactions on Mathematical Software, Vol. 38, Issue 1, 2011.

［11］ Cavazos et al. "Rapidly selecting good compiler optimizations using performance counters," International Symposium on Code Generation and Optimization, pp. 185-197, 2007.

［12］ Wang et al. "Automatic hyper-parameter tuning of machine learning models under time constraints. IEEE Big Data 2018 Workshop, the Second International Workshop on Automation in Machine Learning and Big Data (AutoML 2018), pp. 1-7, 2018.