Vol.2022-ARC-248 No.1
Vol.2022-SLDM-198 No.1
Vol.2022-EMB-59 No.1
2022/3/10

分岐命令の選択的近似による決定木アンサンブルの高速化

釜堀 恵輔[†1,a)]　高前田 伸也[†1,b)]

概要：軽量な組込みプロセッサで効率的に機械学習処理を行うためには，エネルギーやリソース使用量を節約できる高速化機構が望まれている．本論文では，機械学習モデルの精度といったサービスの品質を確保しつつ，プログラマによるアノテーションに基づき，軽量なプロセッサ上で決定木アンサンブルなどの機械学習アプリケーションの性能を改善する近似計算手法を提案する．本手法は，決定木中の重要度などに基づき選択された分岐命令について，分岐予測ミスの際にロールバックを行わずに，誤った分岐パスをそのまま継続実行することで，性能向上を目指す．提案手法はエッジデバイス上の機械学習アプリケーションを高速化するための有望なアプローチとなると考えられる．

1. はじめに

機械学習アプリケーションなどのデータ集約的なワークロードの増加に伴い，コンピュータに求められる性能は年々飛躍的に向上している．エッジコンピューティングは，データの処理をデータの生成源の近くで行うアプローチであり，エッジデバイスとデータセンター間のネットワーク転送量を削減できるという利点がある．そのため，低遅延で動くことが必要な機械学習アプリケーションで広く利用されている．しかしながら，多くの場合プロセッサの性能，メモリ容量，回路面積など，利用可能な計算資源に厳しい制約がある．特に，回路規模が小さい超低消費電力のエッジデバイスでは，汎用プロセッサと機械学習用のドメイン特化型アクセラレータの両方を用意することは困難であるため，小さなプロセッサ上で機械学習ワークロードを処理することが望まれる．

本研究では，決定木ベースのアルゴリズムなどの機械学習アプリケーションの処理を汎用マイクロコントローラ上で高速化するために，分岐命令の選択的近似手法を提案する．これは分岐予測ミスに伴うペナルティをなくして処理速度を向上させる手法である．一般的なインオーダパイプライン型プロセッサアーキテクチャでは，パイプラインの後段で分岐命令の方向 (分岐先に進むか否か) を決定するため，分岐予測器によって予測した分岐方向が間違っていた場合にロールバックが発生し，性能劣化につながる可能性がある．提案手法は，機械学習モデルの予測精度などのアプリケーションの品質がある程度低下することを許容する代わりに，処理速度向上を目指す近似計算手法である．これはプログラマがアノテーションしたコンパイル時情報に基づいて，分岐命令の一部を近似的なものに置き換えることでロールバックによる処理速度低下を防ぐ．さらに，本手法では，近似分岐命令によってアプリケーションの品質が許容範囲を超えて悪くなることを防ぐために，実行時の動的な統計量をもとに近似分岐命令の挙動を制御する．

本論文では，提案の選択的近似分岐命令の決定木アンサンブルへの適用方法を示す．今後，提案した選択的近似分岐命令を RISC-V[8] の拡張として実装し，決定木アンサンブルアルゴリズムなどの機械学習アプリケーションを対象に性能を評価する実験を行う予定である．提案手法は，リソースに制約のあるエッジデバイス上の機械学習アプリケーションの高速化に貢献することが期待される．

2. 選択的近似分岐

ソフトウェアレベルで付与されたアノテーションを元に分岐命令を選択的に近似する手法を提案する．提案手法の核をなすのは近似分岐命令である．従来の分岐命令では，分岐の方向はパイプラインの後段で決定されるため，プロセッサが実際の分岐方向を計算する前に分岐予測器を用いて分岐方向が予測され，それによって次に実行する命令が決定される．分岐予測器の出力と実際の分岐先が一致しない場合，プロセッサは誤って実行した命令をロールバックしてから正しい方向の命令を実行する．しかし，近似分岐命令に対しては，プロセッサはたとえ予測器が間違っている場合でも，ロールバックせずに誤った方向の命令の実行を続ける．

[†1] 東京大学
The University of Tokyo, Bunkyo, Tokyo 113–8656, Japan
[a)] k-kamahori@g.ecc.u-tokyo.ac.jp
[b)] shinya@is.s.u-tokyo.ac.jp

Vol.2022-ARC-248 No.1
Vol.2022-SLDM-198 No.1
Vol.2022-EMB-59 No.1
2022/3/10

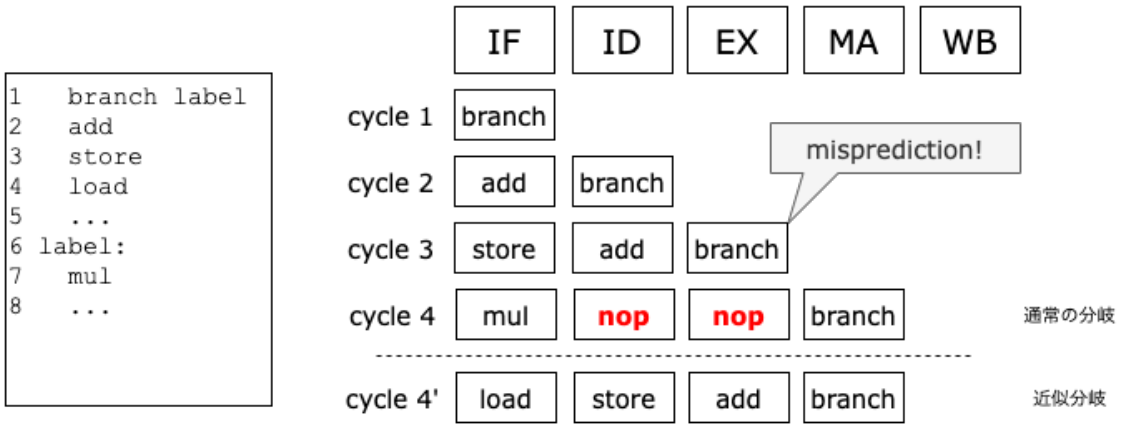

図 1 選択的近似分岐命令の動作例．通常の分岐命令では誤って実行した命令をロールバックし正しい方向の実行を行うが，近似分岐命令では誤った方向での実行をそのまま続ける．

図 1 にパイプラインプロセッサにおける近似分岐命令の動作例を示した．図の左側にプログラムの命令列の例を，右側にそのプログラムを動作させたときの各サイクルにおけるパイプラインの状態を示している．命令 1 が分岐命令となっており，条件に基づいて次に実行する命令が次の行の命令となるか，ラベルで指定した 6 行目以降となるかが決まる．ここで，正しい分岐方向は 6 行目以降だが，プロセッサの分岐予測器は 2 行目以降に進むと予測した場合を考える．一般的なプロセッサにおいては，分岐のどちらに進むかは EX ステージで判定される．そのため図の右側に示したように，通常の分岐命令は 3 サイクル目に分岐が誤っていたことが判明するまでは予測に従って実行し，それ以降は誤ってパイプラインに投入した命令をロールバックし正しい方向で実行を続ける (サイクル 4 で示した)．一方，近似分岐命令では分岐が誤っていたことが判明した後も，命令をロールバックすることはなく誤った方向の実行を継続する (サイクル 4' で示した)．

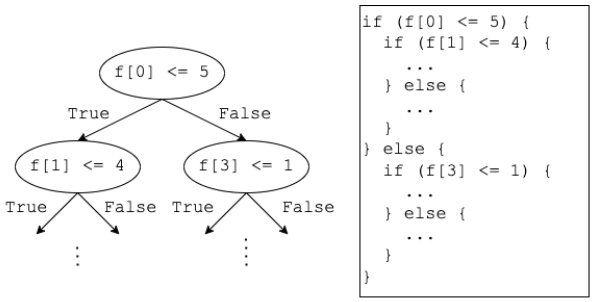

図 2 決定木アルゴリズムの模式図
各ステップで入力データの特徴量と事前に学習された閾値を比較し，それに基づき分岐する

この手法は，決定木アンサンブルなどの機械学習ワークロードを高速化するために有益な技術として期待される．決定木に関するの処理の多くは，図 2 で示したように入力として与えられるデータの特徴量とモデル学習時に定められた閾値を比較し，それに基づき処理の方向を分岐させる命令である．そのため，分岐の方向を分岐予測器で予測することが難しく，分岐の正しい経路を実行するために多くのロールバックが発生し，プロセッサのスループット (IPC) を劇的に低下させることになる．しかし，決定木アンサンブルのような統計的アルゴリズムでは，正しい予測値を出力するためには必ずしも完全な精度でプログラムを実行する必要がない場合が多く，近似計算アプローチによる最適化が可能であると考えられる．そこで本手法では，分岐予測ミスに伴うロールバックを確率的に解消することで，性能を向上させることを目指す．

以下で，決定木アンサンブルに関する前提知識と，提案手法の詳細を説明する．

2.1 決定木アンサンブル

アンサンブル学習とは，複数の異なる機械学習モデル (弱学習器) を用意し，それらの多数決を取ることによって予測精度を向上させる方法である．決定木アンサンブルは，図 2 で示したような決定木を複数組み合わせる手法であり，幅広い分野で応用される一方，プロセッサで実行する上では先ほど述べたような分岐命令に伴うオーバーヘッドが問題となりうる．しかしながら，アンサンブル学習の性質上，全ての弱学習器が正しい結果を出力する必要がない．そのため，本論文で提案する近似計算手法が有用になると考えられる．

2.2 近似分岐命令の選択方法

近似分岐は，ソースコードにおける if 文や while 文のアノテーションとしてソフトウェアレベルで指定されることを想定している．すなわち，近似させることが可能な分岐命令を選択するのはプログラマである．

近似分岐は，ハードウェアが動的に選ぶのではなくソフ

トウェア上で指定される．これはプログラマにとって面倒なことかもしれないが，以下の理由からソフトウェア上で指定することを提案する．

- 品質評価指標はアプリケーションにより異なり，ハードウェアで動的に近似可能な分岐命令を選択するのは困難である．
- ソースコードにアノテーションを提供することは，プログラマにとって新しいことではない．例えば，高性能計算アプリケーションなどではプログラマが手動で並列処理などを指定することが多い．
- 機械学習アプリケーションを作成する際には，フレームワークを用いてアプリケーションを書くことが多くなっており，そのようなフレームワークを用意すれば，プログラマは近似分岐を指定する必要がなくなることが期待される．

以上の理由から，近似分岐命令はソフトウェアレベルで選択するように設計している．

2.3 品質の確保

近似計算の問題点は，機械学習モデルの精度などアプリケーションの品質が許容範囲を超えて悪くなる可能性があることである．その問題を回避するため，提案手法では，近似分岐が誤った方向に実行された割合，すなわち，近似分岐命令の合計実行回数に対する近似分岐命令が本来の意図とは異なる方向で実行された回数の割合を計算し，その上限をプログラマが指定する設計とする．

この問題を回避するための別の方法として，各々の近似分岐命令に対して期待される精度を指定する方法も考えられる，しかしながら，以下の理由から最適とは言えないと考えられる．

- 一部の分岐命令は 1 回しか使われないので，それぞれについて最小精度を指定することには意味がない．
- 近似分岐を使うたびに最小限の精度比を指定するのは，プログラマにとって非常に面倒なことである．
- そのような手法は各命令に精度比の情報を持たせる必要があるが，短い命令語を持つという RISC アーキテクチャの利点が損なわれる．
- プログラムは任意に大きくなるため，近似分岐命令ごとに精度情報を保存することは現実的ではなく，一般的な分岐予測器のようにアドレスビットのサブセットのみを使用して精度を管理することはこの場合意味がない．

したがって，ここでは，近似実行の総合的な精度のみをカウントする．

近似分岐の精度とアプリケーションの品質は必ずしも一致しないが，本技術が様々なドメインで利用されることを見据え，上記の指標がプロセッサの判断基準として最も適していると考える．

2.4 分岐予測

分岐予測器は，近似分岐においても予測結果が正しかったかどうかを確認し，それをもとに学習する．近似分岐命令が正しく実行されたかどうかを判断することは，計算の無駄遣いになるという指摘も想定される．しかし，近似が正確であったと判断することの利点としては動的分岐予測器の学習に使用することができることと，それによって前の節で説明したようなプロセッサの出力の品質が許容範囲にあることをを保証する機構を提供できるということが挙げられる．

3. 評価手法

前章で説明した設計を RISC-V のカスタム命令としてに実装する予定である．MARSS-RISCV シミュレータ [1] は，パイプライン方式の RV32GC ISA をサポートするサイクルレベルのソフトウェアシミュレータであり，インオーダとアウトオブオーダの両方を提供している．軽量なエッジデバイスへの応用を目指すという本研究の目的から，MARSS-RISCV のインオーダ版を利用し，近似分岐命令をサポートするように拡張する．

決定木アンサンブルに基づいた分類モデルを用いて，提案手法の性能を評価する予定である．決定木ベースのアルゴリズムの学習には，scikit-learn[5] を使用し，キャッシュのサイズや分岐予測アルゴリズムが異なる様々なアーキテクチャの下で，IPC，分岐近似精度の下限，決定木アルゴリズムの予測精度の関係を測定する予定である．

4. 関連研究

Nongpoh らは，分岐命令を近似することでプログラムを投機的に実行することを提案した [4]．しかし，彼らの方法は，実行前に既知の品質評価指標でプログラムをプロファイリングする必要があり，実世界での利用にはあまり適していないと考える．一方，我々の手法は，ソフトウェア的な手法で分岐命令に近似を適用するものである．このため，プログラマに余分な作業が発生するが，前章で述べたように，性能向上につながるのであればこのデメリットは無視できる程度である．

また，中村らは，近似計算における近似度を反復的に変更する手法 Stochastic Iterative Approximation を提案した [3]．この手法において近似的なプログラムと正確に計算するプログラムの切り替えのために近似分岐命令を用いているが，もとのプログラム中に存在する分岐命令を近似的に実行しているわけではないという点において本研究とは異なる．

Esmaeilzadeh らは従来のパイプラインに不正確ながら低電圧のパイプラインを組み合わせたアーキテクチャを提案し [2]，San Miguel らはロード命令の近似を提案した [6]．これらの研究は近似分岐に関するものではないが，将来的

に本研究の提案手法との組み合わせも可能であると考えられる．

5. 結論

本研究では，ある程度の品質低下を許容する代わりに，エッジデバイス上の機械学習アプリケーションの性能を向上させるアプローチである選択的近似分岐命令を提案した．今後，シミュレータ上で提案手法を実装し，その有用性を検証していく予定である．

謝辞 本研究の一部は，JSPS 科研費 19H04075 および 18H05288 の支援により行われたものである．

参考文献

[1] Bellard, F., Kothari, G., Sarnaik, P., Yüksek, G. (2019). MARSS-RISCV: Micro-Architectural System Simulator for RISC-V [Source code]. https://github.com/bucaps/marss-riscv

[2] Esmaeilzadeh, H., Sampson, A., Ceze, L., Burger, D. (2012, March). Architecture support for disciplined approximate programming. In Proceedings of the seventeenth international conference on Architectural Support for Programming Languages and Operating Systems (pp. 301-312).

[3] Nakamura, T., Tomida, K., Kouno, S., Irie, H., Sakai, S. (2021, October). Stochastic Iterative Approximation: Software/hardware techniques for adjusting aggressiveness of approximation. In 2021 IEEE 39th International Conference on Computer Design (ICCD) (pp. 74-82). IEEE.

[4] Nongpoh, B., Ray, R., Das, M., Banerjee, A. (2019). Enhancing Speculative Execution With Selective Approximate Computing. ACM Transactions on Design Automation of Electronic Systems (TODAES), 24(2), 1-29.

[5] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., ... Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. the Journal of machine Learning research, 12, 2825-2830.

[6] San Miguel, J., Badr, M., Jerger, N. E. (2014, December). Load value approximation. In 2014 47th Annual IEEE/ACM International Symposium on Microarchitecture (pp. 127-139). IEEE.

[7] ultraembedded (2014). RISC-V Core [Source code]. https://github.com/ultraembedded/riscv

[8] Waterman, A., Lee, Y., Patterson, D. A., Asanovi, K. (2014). The risc-v instruction set manual. volume 1: User-level isa, version 2.0. California Univ Berkeley Dept of Electrical Engineering and Computer Sciences.