

Modelos de Predição de Erro e Consumo Energético em Computação Aproximada

Gregório Koslinski Neto¹, Gislayne Garabini Damasceno¹,
Marco Antônio de R. Zárate¹, Liana Duenha¹

¹Faculdade de Computação – UFMS Campo Grande – MS – Brasil

{gregorio.neto, liana.duenha}@ufms.br

Resumo. A computação aproximada permite a execução de operações com precisão reduzida em cenários tolerantes a erro, com ganhos significativos de energia. No nível de instrução, cada operação aritmética pode ser mantida exata ou substituída por uma contraparte aproximada, e a avaliação de cada variante resultante por simulação é computacionalmente cara. Este trabalho propõe modelos de aprendizado de máquina baseados em árvores (LightGBM, Random Forest e Extra-Trees) para estimar o erro de saída e o consumo de energia das variantes aproximadas, treinados com dados de simulação e de análise estática extraídos via LLVM/MLIR. Avaliados em cinco aplicações do AxBench, os modelos alcançaram $R^2 \geq 0,955$ para erro e $R^2 \geq 0,986$ para energia, com correlação de Spearman superior a $\rho = 0,966$, mantendo a ordenação das variantes em relação à simulação.

1. Introdução

A computação aproximada tem sido explorada em domínios intrinsecamente tolerantes a erro, nos quais relaxar os requisitos de precisão permite um equilíbrio favorável entre a qualidade da saída e a eficiência energética [Yazdanbakhsh et al. 2017]. Nesse cenário, a avaliação empírica de cada variante de aproximação por simulação é custosa, motivando modelos que estimem seus efeitos sem executá-la.

Este trabalho constrói o conjunto de dados e os preditores de erro e energia. Cada operação elegível é caracterizada por *features* estáticas extraídas via infraestrutura LLVM/MLIR, como vivacidade de variáveis, profundidade e estrutura de aninhamento de laços, e por *features* dinâmicas obtidas por simulação. A validação foi conduzida em ambiente de simulação RISC-V com instruções aproximadas sobre o benchmark Ax-Bench [Yazdanbakhsh et al. 2017].

2. Metodologia

O conjunto de dados de treinamento é formado por *features* estáticas e dinâmicas. As estáticas são extraídas do código-fonte via infraestrutura LLVM/MLIR, que identifica propriedades estruturais e semânticas das instruções elegíveis que influenciam a propagação de erro e a qualidade da saída, por técnicas de fluxo de dados de vivacidade [Aho et al. 2006], propagação de erro [Barros et al. 2021] e análise ciente de laços, que incorpora a estrutura iterativa, dominante no tempo de execução. As dinâmicas, obtidas por simulação no simulador SPIXE, descrevem o comportamento das aplicações em tempo de execução. As *features* são organizadas em grupos: codificação binária das linhas com instruções aproximadas; contagens dinâmicas de operações aproximadas por

tipo; propriedades de região via MLIR (usos de variáveis, dependência de entrada externa, erro estático esperado, estrutura de laços e variáveis globais); distância ao retorno da função em quatro níveis; criticidade e dependência; e profundidade na árvore de expressão, indicando quão profundamente o erro pode se propagar.

As configurações de aproximação foram geradas por combinações binárias das operações aritméticas elegíveis. Os quatro benchmarks menores, com espaços de 512 a 8.192 variantes, foram enumerados completamente; para Black-Scholes a enumeração é inviável, sendo o conjunto de dados uma amostra estruturada de 35.119 configurações ($\approx 0,05\%$ do espaço), cobrindo todas as configurações de linha única, quase todas as de duas e três linhas e a enumeração de um subespaço de 15 linhas. Cada variante foi simulada para coleta das métricas de erro específicas (MARE, *miss rate*, RMSE) e do consumo energético, com até 234,5 horas de simulação (Black-Scholes).

A avaliação considerou regressores de *ensemble* baseados em árvores: o LightGBM [Ke et al. 2017], modelo principal por sua eficiência com as *features* binárias de alta dimensionalidade; a Random Forest [Breiman 2001], linha de base; e a Extra-Trees [Geurts et al. 2006], alternativa para conjuntos menores. Todos usaram `n_estimators=3000`; o LightGBM foi ajustado com taxa de aprendizagem reduzida, regularização e *early stopping*, com divisão 80/20 estratificada e demais configurações padrão do scikit-learn [scikit-learn developers 2024].

A estratégia de predição foi adaptada à distribuição de erro: aplicações com distribuição bimodal e pico de saturação (Black-Scholes e Inversek2j) empregaram *pipeline* de dois estágios, com classificador binário que decide se a variante colapsa em falha e regressão do erro apenas das variantes não falhas, enquanto FFT, Jmeint e K-Means, unimodais, usaram regressão direta. Um preditor é treinado por aplicação.

Três análises avaliaram quantas *features* são necessárias para predição acurada: correlação de Pearson, importância por permutação [Breiman 2001] e ablação, que re-treina o preditor em subconjuntos progressivamente menores. A correlação em Black-Scholes revelou redundância limitada (13 de 351 pares, 3,7%, com $|\rho| > 0,9$). O ranking de importância combina métricas estruturais do MLIR e descritores dinâmicos da região aproximada, com `uses_global_sum` presente nos cinco benchmarks e contadores de operações aproximadas em quatro de cinco. A ablação indica que o catálogo pode ser reduzido à união das 20 *features* superiores por benchmark sem perda mensurável de acurácia.

3. Resultados e Discussão

A Tabela 1 resume o desempenho preditivo para erro e energia no teste retido: Jmeint apresentou predição quase perfeita, com comportamento altamente regular, enquanto Inversek2j exibiu o maior MAE apesar do elevado R^2 , refletindo maior variabilidade intrínseca. Entre as arquiteturas avaliadas, o LightGBM apresentou a melhor média de R^2 para o erro (0,966, contra 0,908 e 0,918 das alternativas), média competitiva de 0,995 para energia e treinamento aproximadamente $10\times$ mais rápido, sendo selecionado como preditor principal dos experimentos.

Quantificando a simulação requerida, o re-treinamento sobre frações amostradas dos espaços mostrou que com apenas 25% das configurações os preditores de Black-

Scholes alcançam $R^2 = 0,87/0,99$ (erro/energia), reduzindo a geração do conjunto de dados de 234,5 para menos de 59 horas; com 50%, aproximam-se da acurácia completa, e os classificadores de falha excedem 99% de acurácia com apenas 2% do espaço. Na análise de exclusão de uma linha (*leave-one-line-out*), a extrapolação a linhas nunca aproximadas é viável nos espaços mais ricos (mediana de R^2 entre 0,90 e 1,00 em Black-Scholes e Jmeint), mas não nos menores.

Tabela 1. Desempenho preditivo dos modelos LightGBM no teste retido (20%).

Aplicação	Métrica	R^2 erro	MAE	R^2 energia	MAPE (%)
Black-Scholes	MARE	0,962	0,033	0,998	0,058
FFT	MARE	0,957	0,040	0,996	0,292
Inversek2j	MARE	0,956	0,242	0,986	2,434
Jmeint	miss rate	0,999	0,002	1,000	0,000
K-Means	RMSE	0,955	0,024	0,995	0,040

4. Conclusão

Apresentamos preditores de aprendizado de máquina para o erro de saída e o consumo energético de variantes aproximadas em nível de instrução em RISC-V, substituindo a simulação por predição. Três arquiteturas de *ensemble* foram avaliadas, com o LightGBM selecionado como preditor principal por sua generalização superior e treinamento aproximadamente $10\times$ mais rápido; Black-Scholes e Inversek2j exigiram *pipeline* de dois estágios; os demais, estratégia direta. Todos alcançaram $R^2 \geq 0,955$ para erro e entre 0,986 e 1,000 para energia, com MAPE abaixo de 2,5%. Como trabalhos futuros, destacam-se a robustez para casos extremos e o treinamento a partir de pequenos subconjuntos amostrados, eliminando a geração exaustiva de conjuntos de dados.

Referências

- Aho, A. V., Lam, M. S., Sethi, R., and Ullman, J. D. (2006). *Compilers: Principles, Techniques, and Tools*. Pearson Education, Boston, 2nd edition.
- Barros, G., Akemaru, R., Bezerra, E., Oliveira, R., and Mazzine, M. (2021). The synthesis of computational imprecision: Impact of compilation techniques on the approximate computing paradigm. *ACM Computing Surveys (CSUR)*, 54(8):1–35.
- Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.
- Geurts, P., Ernst, D., and Wehenkel, L. (2006). Extremely randomized trees. *Machine learning*, 63(1):3–42.
- Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T.-Y. (2017). Lightgbm: A highly efficient gradient boosting decision tree. *Advances in neural information processing systems*, 30.
- scikit-learn developers (2024). scikit-learn: Machine learning in Python. <https://scikit-learn.org/>. Version 1.6.1. Accessed: June 2026.
- Yazdanbakhsh, A., Mahajan, D., Esmaeilzadeh, H., and Lotfi-Kamran, P. (2017). Axbench: A multiplatform benchmark suite for approximate computing. *IEEE Design and Test*, 34(2):60–68.