*CS595: Proposal of Course Project. Due Oct 4, 2018*

**Application of Machine Learning in Selecting Sparse Linear Solvers**

***Wes Ludwig, Max Oellien***

**Abstract**

Solving sparse linear systems is a vital and resource-demanding task in scientific computing, and the selection of system-solving algorithms is daunting. Due to the variability of linear systems associated with even a single PDE-based simulation, solver selection is a difficult, necessary task for specialists to ensure computational performance. Selection heuristics can be produced and tuned through the use of Machine Learning (ML) techniques, generating solver-selection algorithms to increase the efficiency of large computation tasks. Here we propose to extend the existing application of Alternating Decision Trees in Selecting Sparse Linear Solvers by Bhowmick, Eijkhout, et al to a novel set of linear systems to further demonstrate the efficacy of their method.

1. **Introduction**

Solving large sparse systems of linear equations is a fundamental problem in scientific computing and often demands a majority of computing time. Intelligent selection of solution algorithms is vital for ensuring computability and maximizing efficiency. It is therefore vital that tools be constructed to generalize and optimize this process to allow for greater computational power for less specialized scientists. By applying the Probably Approximately Correct theory of machine learning, a series of ‘weak learners’, correlations between system descriptors and solver performance data, can be combined to form a ‘strong learner’, capable of producing hypotheses with a high probability and small generalization error.

This process, called boosting, is used in this case to generate a hypothesis - a rule for classifying instances - in the form of an Alternating Decision Tree (ADT). The resulting decision tree can be called as a function on a set of linear system characteristics, generating a negative or nonnegative score for a given solver’s suitability for the linear system. Linear system characteristics include formatting information, structural information, various norms, spectral statistics, and variability measures. Some of these characteristics are themselves computationally expensive to determine, indicating the importance of selecting a robust, representative training distribution to make the ML process time-efficient.

1. **Current Status**

The scientific community surrounding this problem has generated a robust body of work and already showed the viability and advantage of the method (Bhowmick et al, 2006, 2010). The team has developed software tools where necessary, most importantly the creation of a ‘Standard and Software for Numerical Metadata (NMD)’ (Eijkhout and Fuentes, 2009), allowing for the generation of files containing linear system metadata. The authors perform dataset partitioning, classification, and boosting using the MLJava library; and have taken their example linear systems from PETSc example code and the M3D plasma simulation; and take their linear system solvers from the PETSc solver library.

In each example case, the set of linear systems is partitioned into training and test sets, and both sets have their system properties determined by the NMD AnaMod software. A database is created including entries for the possible solver-parameter-preconditioner combinations, and the performance of solvers on linear systems. Classification is performed iteratively in the boosting algorithm, and correlations between classifiers and performance are weighted and saved. Those weighted correlations are stored in the ADT which can then be used to classify novel linear systems. This process has been shown to be time-advantageous for even relatively small numbers of simulations. Classifier construction for PETSc ex27 required 60 linear systems to be solved, while each simulation generates 27 linear systems, in that case the classifier computing expense was amortized by only three simulation instances. In this case the average execution time saved by the classifier was 33% over the default GMRES-30-ILU.

1. **Proposed Work**

We propose to replicate the method laid out in the 2010 paper for a novel set of linear systems. We would set out to label a subset of linear systems for training and testing using NMD AnaMod, solve training systems with PETSc solvers to develop performance characteristic labels, and implement boosting in MLJava to generate a classifier ADT which could be used to decrease computing time for the full set of linear systems. Though we have not chosen an example set of linear systems, the later tutorials and matrix examples of the PETSc library present numerous possibilities.

1. **Expected Accomplishments**

We are modest in our current expectations as we have not been successful soliciting information from the study authors as of yet. Having example code would greatly increase our ability to perform our task, especially considering that the vital tool NMD AnaMod no longer has accessible documentation. Without assistance we could likely use another NMD labelling software and the remaining work would be largely similar, though without example code the remaining work would be much more difficult. A vital expectation would be to create a small functioning system with a limited number of linear systems and solver options to limit computing time and to show proof-of-concept. This test system could then be expanded to the full scale of a simulation example set. Without aid, this small-scale proof-of-concept may be the limit of our expectations, while with information from the authors we could expect to create a functioning classifier from the full dataset with demonstrably improved performance to the default solver.

1. **References**
2. Bhowmick, S., Eijkhout, V., Freund, Y., Fuentes, E., & Keyes, D. (2006). Application of Machine Learning in Selecting Sparse Linear Solvers. Retrieved from <http://pages.tacc.utexas.edu/~eijkhout/Articles/2006-bhowmick.pdf>
3. Bhowmick, S., Eijkhout, V., Freund, Y., Fuentes, E., & Keyes, D. (2010). Application of Alternating Decision Trees in Selecting Sparse Linear Solvers. Software Automatic Tuning, 153-173. doi:10.1007/978-1-4419-6935-4\_10. Retrieved from

<https://link.springer.com/chapter/10.1007%2F978-1-4419-6935-4_10>

1. Bhowmick, S., Toth, B., & Raghavan, P. (2009). Towards Low-Cost, High-Accuracy Classifiers for Linear Solver Selection. Lecture Notes in Computer Science Computational Science – ICCS 2009, 5544, 463-472. doi:10.1007/978-3-642-01970-8\_45. Retrieved from <https://link.springer.com/chapter/10.1007/978-3-642-01970-8_45>
2. Dongarra, J., & Eijkhout, V. (2003). Self-Adapting Numerical Software and Automatic Tuning of Heuristics. Lecture Notes in Computer Science Computational Science — ICCS 2003, 759-767. doi:10.1007/3-540-44864-0\_78

Retrieved from <https://link.springer.com/chapter/10.1007/3-540-44864-0_78>

1. Eijkhout, V., & Fuentes, E. (2009). A Standard and Software for Numerical Metadata. ACM Transactions on Mathematical Software, 35(4), 1-20. doi:10.1145/1462173.1462174. Retrieved from

<https://pdfs.semanticscholar.org/1d0b/051b34ce43c77bdf07eb5db7f460894946f8.pdf>

1. Eijkhout, V., & Fuentes, E. (2010). Machine Learning for Multi-stage Selection of Numerical Methods. New Advances in Machine Learning. doi:10.5772/9376. Retrieved from

<http://pages.tacc.utexas.edu/~eijkhout/Articles/2010-intech.pdf>

1. Eijkhout, V. (2008, August 19). NMD: Numerical Metadata library Reference Manual. Retrieved from <https://www.icl.utk.edu/files/publications/2007/icl-utk-183-2007.pdf>