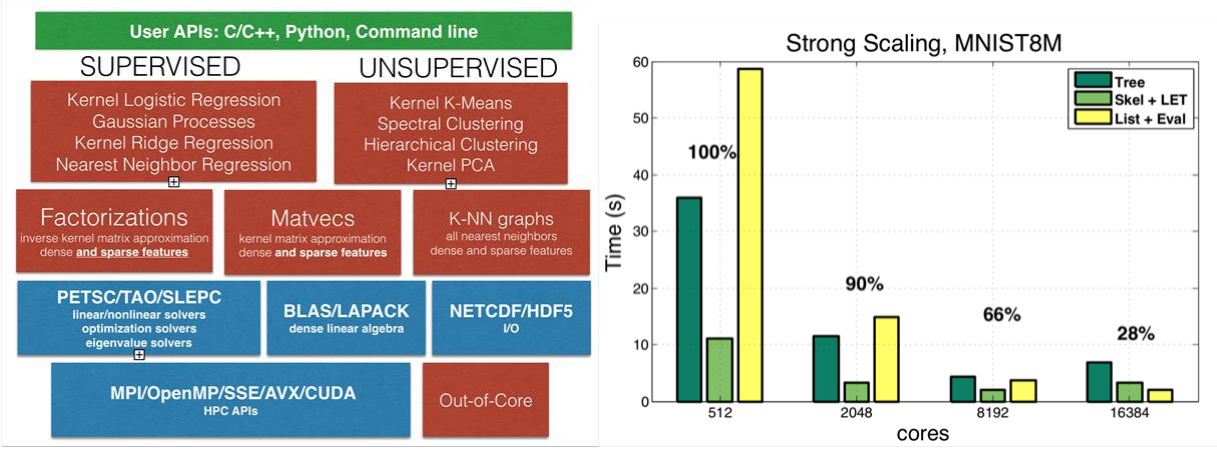
HPC Algorithms for Scalable Machine Learning and Data Analytics

George Biros
Parallel Algorithms for Data Analytics and Simulation Group
Oden Institute, The University of Texas at Austin

Increasingly, scientists and engineers are incorporating data analytics tasks in their workflows. Examples, of such tasks include uncertainty quantification, inverse problems, design and control, classification, anomaly detection, rare-event detection, data-assimilation, model reduction, pattern discovery, graph analytics, compression, visualization and dimensionality reduction to name a few. Many high-fidelity simulations in science and engineering require High Performance Computing (HPC) resources. Therefore, to increase the coherence between simulation and data analytics we need software that scales to large datasets and can support current and future HPC architectures.

Many groups in industry, government labs, and academia are working towards such HPC machine learning and data analytics software. However, the need to process in-situ large scientific and engineering datasets is not fully met with current software, and sometimes significant down-sampling is required in order to use existing tools. Also, popular ML libraries like Torch and TensorFlow are designed around industrial applications (natural language processing, audio, images, and video) and not scientific applications that have more unstructured data.

In my group, we have been focusing on algorithms and software for HPC machine learning algorithms with a focus on nearest neighbor graphs and kernel methods. (See figure below.)



***Left**: Work on HPC algorithms for machine learning in the PADAS group at UT. Red boxes indicate components developed in our group and support supervised and unsupervised learning. Different modules for kernel approximations, randomized nearest-neighbors, factorization of kernel matrices, kernel PCA, and toward supervised and unsupervised learning. **Right:** Scalability of kernel regression on a dataset with 8 million points in 768 dimensions. The run took place on 16,384 of TACC's Stampede I system.*

This research is funded by NSF awards CCF-1817048 and CCF-1725743.

Examples of kernel methods include support vector machines, logistic regression, Gaussian process approximation, density estimation, and clustering. These methods are in turn used in numerous applications in signal analysis, text analysis, and bioinformatics, and science and engineering. A key bottleneck in kernel methods is that they scale quadratically with the problem size. The problem size is typically defined as the number of examples (or training points)

presented to the machine learning algorithm. Since training points can number in the millions, various approximation schemes are necessary for scalability to large-scale problems. Software for kernel methods has to meet several conflicting goals: it should use of state-of-the art algorithms (which may be complicated and difficult to parallelize), achieve both shared memory and distributed memory scalability so that large-scale problems can be solved, be robust and easy to use (no excessive parameter tuning), achieve high-performance so that the underling hardware resources are efficiently utilized, and offer a sufficiently general suite of algorithms. Most of the existing machine packages either avoid kernel methods due to their complexity (related to the need to approximate dense matrices), or when they include kernel methods they lack accuracy (due to crude approximations) and scalability---particularly on distributed memory architectures. We have developed several C++ libraries that provide scalable implementations for a large class of problems. With support by NSF we have developed scalable software for kernel methods: ASKIT (library for kernel density estimation and kernel regression), RKDT (nearest neighbor algorithms), GOFMM (geometry oblivious hierarchical matrices that can be used to ``invert'' kernel matrices), and KLR (a library for kernel logistic regression). These packages can be found in https://padas.oden.utexas.edu. These algorithms have been ported on several XSEDE platforms at TACC and have been scaled to billions of points. Scalability is achieved by using a combination of MPI, OpenMP and GPU acceleration.

Outstanding challenges / requirements

- Performance portability on upcoming heterogeneous architectures.
- Utilizing variable precision algorithms.
- Support of development and deployment of HPDA algorithms and software that supports a large variety of structured and unstructured data.
- Support for easy to use interfaces, e.g., using Python or other high-level languages.
- Simpler interfaces to leadership resources.
- Long term publicly visible data repositories.
- Reproducibility of complex workflows with dependencies to ever changing libraries and large data.