P1415R0: SG19 Machine Learning Layered List

Date: 2019-01-21 (Pre-KONA mailing): 10 AM ET

Project: ISO JTC1/SC22/WG21: Programming Language C++

Audience: SG19, WG21

Authors: Michael Wong (Codeplay),
Vincent Reverdy (University of Illinois at Urbana-Champaign, Paris Observatory),
Ritwik Dubey (Exegy)
Richard Dosselmann (University of Regina)
Eugenio Bargiacchi (Vrije Universiteit Brussel)

Contributors
Jordi Inglada (CNES, French Space Agency)

Emails: michael@codeplay.com
vreverdy@illinois.edu
ritwik.exe@gmail.com
dosselmr@cs.uregina.ca
svalorzen@gmail.com

Reply to: michael@codeplay.com

Introduction

P1360 [P1360] establishes the initial charter for the formation of a Machine Learning SG for C++. This paper proposes a list of Machine Learning (ML) and Data Science (DS)[1] topics for C++, in a layered format that shows dependency rather than prioritization. It nonetheless indicates a possible order in which SG19 will tackle these issues.

Motivation

Since SG19's inception in SAN, introduced in P1360, SG19 has had 2 monthly telecon meetings on the 2nd Friday of each month. During that time, we have discussed various topics suitable for C++ (See SG19 meeting minutes).

This paper will start the discussion on a layered list of features. We do not call it a priority list because our various priorities may be different. This is more of a dependency list.

Layered Feature list

We will separate this list into features of Layers 0, 1, 2, 3, etc. The aim is to focus the group's effort, but it does not preclude features that are a particular favourite of the members from moving ahead. The understanding here is that some features are dependent on other features that have not yet been resolved.

Layer 0

- Fundamental arrays, matrices, vectors, tensors, linear algebra
    - There is already a design proposed for Linear Algebra from SG14 [Pxxxx], as well as a history document [Pxxx]. That design is based on a confluence of ideas from games and high-performance computing. They are also using a layered approach. There are requirements that Machine Learning would need to build on top of that and the group has been discussing ideas that would add to the Linear Algebra proposal, such as how to add convolutions or eigenvectors/eigenvalues for principal component analysis (PCA).
    - In addition to multidimensional numeric arrays, arrays with named columns (i.e. data frames) are also needed for building data science tools. Existing examples in C++ are Root's DataFrame class [root] and xframe [xframe].

[1] We use the ML and DS terms in order to cover a wide range of algorithms and statistical models used to extract knowledge from data, generates inferences, and make predictions. The scope of this paper is not limited to Deep Learning (DL) techniques for which linear algebra and automatic differentiation cover most of the needs. Other ML and statistical approaches have different needs in terms of data structures and algorithms.

- Graph and tree data structures
    - Adjacency matrix (static) and list (dynamic) implementations, property graph, bipartite graphs (graphs are used in artificial neural networks, game maps/worlds, mathematical combinatorics problems, graph search problems, representing networks and communications).
    - Depth- and breadth- first searches, directed, undirected, and bidirectional edges, weighted edges, insert/delete edges, insert/delete nodes, edge iterators, node iterators, get num nodes, get num edges, get node indegree, get node outdegree, are nodes adjacent.
    - The BGL [Boost Graph Library] is an excellent example of the features necessary for graph data structures and cbbowen/graph [cbbowen/graph] is a solid port of this that is STL-like.
    - General *k*-ary trees (trees are used in artificial neural networks, decision trees, expressing hierarchical relationships between concepts, operating system directory structures, compiler parsing, binary space partitioning).
    - Pre-, in-, and post-order traversals, node iterators, get num nodes, weighted edges, insert/delete children/parents, compute depth, merge trees.
    - Tree.hh is an example of a generic tree
- Probability utilities
    - Add moments to distributions (i.e. pdf, mean, var, stdev, etc.). Discrete probability sampling with no copying from existing vector distributions. Simple statistics generation from data.
    - Maybe have a look at BOOST Accumulators [BOOSTACC]

Layer 1

- Facilitate better support for interchange of in memory information/data between packages, as for example the Apache Arrow project [Arrow]
- Basic graphing (in terms of drawing)
    - There is already a group SG13 which is designing incidental, not professional level graphics for C++, partly based on the now defunct graphics proposal.
- Optimization, quantization, parallelism, batching computations of vector, matrix, tensors
- Lazy evaluation execution graphs/workflows
- Support for kernel fusion on training and inference
- Support of accelerator dispatch to inference engines, GPUs, FPGAs, MPSoC, Tensor Processing Units, Coarse Grain Reconfigurable Arrays, many of the newer ML boards from Xilinx, Google, ARM, Wave Computing, Nvidia
- Abstraction layer to hide underlying accelerator platform (e.g. CUDA vs RocM for NVIDIA vs AMD GPUs). This will help ML application in C++ decouple from underlying accelerator platform
    - This is already an ongoing effort in SG1 to support some form of heterogeneous computing. Current effort include using affinity, and executors. Existing

implementations of HPX[hpx], SYCL[Expression] can already use modern C++ to dispatch to offnode.
- Do we need a graph extraction pass on top of C++?
- Support interoperability with data formats from Python and R packages (on this matter, having a look to Apache Arrow as a memory layout for tabular data may be useful)

Layer 2

- Packaging to allow adding computation/data manipulation/scaling packages + dependency
- Support portability to various hardware embedded inference engines, and up down convert of different FP sizes between training and inference
- Support of exchange formats (ONNX, NNEF)
- Although not specific to ML, C++ is lacking a (de facto) standard scientific library similar to GNU Scientific Library (GSL), which, being C, can't use templates or any other form of type system features. Standard C++ provides the Special Math library, but it is limited to special functions. Differentiation, integration, interpolation, minimization, statistics (mean, var, stdev, median, etc.), etc. are needed for data science, with basic statistics being common in many projects.
- Other C++ ML/Data Analysis libs: Shark-ML [shark], MLpack, dlib, root [root]
    - Although there exist several high quality C++ ML libs, the equivalent for statistical modelling and inference is lacking.
- Integration Xla, tvm, tensor-rt, glow

Layer 3

- <tentative, open for discussion> Data visualization tool to aid developers in visualizing datasets, rapid prototyping, and debug ML algorithms.
- Tool to experiment with reinforcement learning (RL), with OpenAI Gym as inspiration. Few C++ libraries are out there for RL. Mlpack, AI-Toolbox are a few examples of RL frameworks in C++, while others haven't gained much popularity yet. RL finds great usability in robotics and embedded systems. A C++ RL framework would find utility among robotics and embedded system software developers, many of whom still program in C/C++.
- Support for artificial neural networks (ANN) and DL (both a major focus of modern AI)

Conclusion

This paper proposes a layered approach to developing Machine Learning facilities, establishing focus and expanding on that in [P1360] for further refinement and discussion.

Acknowledgements

Vincent Reverdy's work has been made possible thanks to NSF Awards CCF-1647432 and SI2-

SSE-1642411.

Michael Wong's work is thanks to Codeplay Software Ltd, ISOCPP Foundation, Khronos, and Standards Council of Canada.

References

[AI-Toolbox] https://github.com/Svalorzen/AI-Toolbox

[AMD] https://gpuopen.com/rocm-tensorflow-1-8-release/

[Armadillo] http://arma.sourceforge.net/

[Arrow] https://arrow.apache.org/

[cbbowen/graph] https://github.com/cbbowen/graph/

[BOOSTACC] https://www.boost.org/doc/libs/1_55_0/doc/html/accumulators.html

[Boost Graph Library] https://www.boost.org/doc/libs/1_69_0/libs/graph/doc/index.html

[Catapult] https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/CNN20Whitepaper.pdf

[CuBLAS] https://developer.nvidia.com/cublas

[CuDNN] https://developer.nvidia.com/cudnn

[Eigen] http://eigen.tuxfamily.org

[Expression] https://www.codeplay.com/portal/05-22-17-implementing-opencl-support-for-eigen-using-sycl-and-computecpp

[GSL] https://www.gnu.org/software/gsl/

[hpx] http://stellar.cct.lsu.edu/projects/hpx/

[Inference] https://developer.arm.com/products/processors/machine-learning/arm-nn

[Maili] https://developer.arm.com/technologies/machine-learning-on-arm/developer-material/software-for-machine-learning-on-arm

[Mlpack] https://www.mlpack.org/

[Nervana] https://spectrum.ieee.org/tech-talk/computing/software/nervana-systems-puts-deep-learning-ai-in-the-cloud

[P1360] Towards Machine Learning for C++: Study Group 19: http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2018/p1360r0.pdf

[Pixel] https://www.blog.google/products/pixel/pixel-visual-core-image-processing-and-machine-learning-pixel-2/

[Rasperry]https://medium.com/tensorflow/tensorflow-1-9-officially-supports-the-raspberry-pi-b91669b0aa0

[ROCm] https://rocm.github.io/ROCmInstall.html

[root] https://root.cern.ch

[Shark] http://image.diku.dk/shark/

[SYCL-DNN] https://github.com/codeplaysoftware/SYCL-DNN

[SYCL-ML] https://github.com/codeplaysoftware/SYCL-ML

[syclBLAS] https://github.com/codeplaysoftware/sycl-blas

[TensorFlow] https://www.tensorflow.org/

[TPU] https://cloud.google.com/blog/products/gcp/google-supercharges-machine-learning-tasks-with-custom-chip

[Tree.hh] http://tree.phi-sci.com/

[xframe] https://github.com/QuantStack/xframe