High-Performance Machine Learning Primitives
High Performance Computing Kernels in N-body Problems

Chenhan D. Yu

Sep 18, 2017
The 5th BLIS retreat!!!
Austin, TX
This year ...

I am on the job market.

Both academia and industry positions are very welcome!
$O(2mnk)$ FLOPS, $O(mn+mk+kn)$ MOPS, ~ 95% PEAK, if $k$ is large enough ($k > 1*KC$)

\[
C = A \times B
\]

bottleneck $O(mn)$

$O(mk)$

$O(kn)$
N-body Operators [A. Gray, ’03]

Describing the “interactions ×” between data points

N-by-N interactions

N query points

N reference points in a k-dimensional feature space

\[ \begin{align*}
C_{ij} & \quad \text{bottleneck } O(mn) \\
A_i & \quad O(mk) \\
B_j & \quad O(kn)
\end{align*} \]

\[ A \times B = C \]
Learning = Less Outputs

Instead, we need some kind of reduction of C. e.g. select r columns (nearest neighbors) [SC’15]
Spatial Reduction

Instead, we need some kind of reduction of $C$. 
e.g. pool each 3-by-3 block (convolution + pooling layer)
Significants

**ML tasks**
- Supervised Regression / Classification
- Clustering
- Dimensionality Reduction
- Neural Networks

**Primitives**
- Nearest Neighbors
- Kernel Methods
- K-mean Partitioning
- Matrix Compression
  - PCA / CUR / Nystrom
- Convolutional Networks

Copyright © 2017, The University of Texas at Austin
These memory reduction schemes require a more flexible interface than GEMM

\[ C_{ij} = \sum_k A_{ik} \times B_{kj} \]

\[ \theta(C)_{ij} = \text{\textcircled{\(.)}} \text{\textcircled{\(\bigoplus\)}} \alpha(A)_{ik} \text{\textcircled{\(\bigotimes\)}} \beta(B)_{kj} \]

GEMM-like generalization [GraphBLAS]
N-body Computation Primitives

\[ \theta(C)_{ij} = \bigcirc \mathcal{K}(\bigoplus \alpha(A)_{ik} \bigotimes \beta(B)_{kj}) \]

packing routines to extract (reorder) A, B, C into matrices

semi-ring operators \(<TA, TB> \rightarrow <TV>\)

kernel function \(<TV> \rightarrow <TC>\)

reduction operator \(<TC, TC> \rightarrow <TC>\)
BLIS (Framework + Kernel)

\[ \theta(C)_i = \bigcirc \mathcal{K} \left( \bigoplus \alpha(A)_{ik} \bigotimes \beta(B)_{kj} \right) \]

packing routines to extract (reorder) A, B, C into matrices

microkernel

semi-ring operators \(<\text{TA}, \text{TB}> \rightarrow <\text{TV}>\)

kernel function \(<\text{TV}> \rightarrow <\text{TC}>\)

reduction operator \(<\text{TC}, \text{TC}> \rightarrow <\text{TC}>\)

Preserve the BLIS structure (the Goto algorithm)
Worry About Optimization?

\[
\theta(C)_{ij} = \bigcirc \mathcal{K} \bigoplus \alpha(A)_{ik} \bigotimes \beta(B)_{kj}
\]

packing routines to extract (reorder) A, B, C into matrices

BLIS microkernel

semi-ring operators \(<TA, TB> \rightarrow <TV>\)

reuse registers C

kernel function \(<TV> \rightarrow <TC>\)

reduction operator \(<TC, TC> \rightarrow <TC>\)

Reduce storage and slow memory complexity by \(O(mk+kn)\)

Reduce storage and slow memory complexity by \(O(mn)\)

Reduce loads/stores from \(O(mc^*nc)\) to \(O(mc)\)
Gesture Recognition
Classification

Cover rectangles with palm
Kernel Density Estimation

Training

\[ K(x_i^T x_j) \times w = u \]

Evaluation

\[ K(x^T x_j) \times = u \]
Portable Performance*

Intel Xeon Phi

*NVIDIA Kepler

Arm s820

Still $O(kN^2)$ does not scale when $N$ is large!
Approximation [SC’15,’17, KDD’15, IPDPS’15-’17, SISC]

Time (seconds)

N

GEMM
Compress
GOFMM

low-rank compression

O(N^2)
brute force

O(N log N)
compression

O(N) evaluation

O(N) solver

Copyright © 2017, The University of Texas at Austin
The Largest Problem?

For example, I systematically discover low-rank and sparse matrix structures such that I can invert a 32M-by-32M kernel matrix in 10 seconds but not 3 years.

*Note: Direct MATVEC on a 32Mx32M matrix takes 120 minutes using 3,072 Haswell cores. Cholesky factorization takes 2.8 years to complete.*
More Primitives

k-Nearest Neighbors (Sandy-Bridge)

- MKL+STL
- GSKNN
- Intel -O3
- GNU -O3

CONV2D (Qualcomm S820)

- Vanilla QSML (SGEMM)
- HMLP conv2d
- TensorFlow (Eigen)

Gflops vs. Dimension (k)
Thank You!