PEAK: Parallel EM Algorithm using Kd-tree

Laleh Aghababaie Beni, Aparna Chandramowlishwaran(Advisor)

University of California, Irvine

HPC Factory

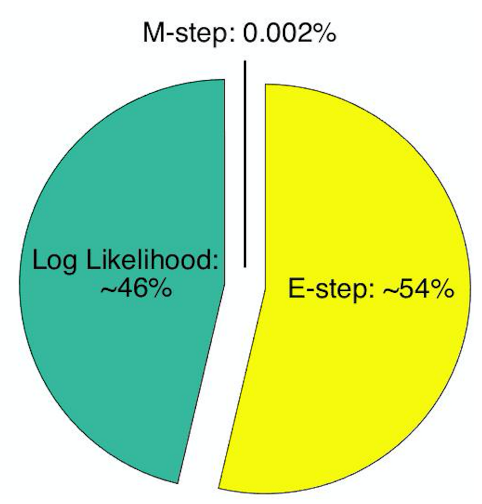
Motivation & Contributions

Why Expectation Maximization?

- **One of the top ten algorithms** having the most impact on data mining
- Popular iterative algorithm for learning mixture models
- Apps: computer vision, machine learning, astronomy, and signal processing

How did we improve it?

- ~99% of total time of EM is spent in two stages namely, E-step and Log-likelihood
- We present a tree-based approximation algorithm for **both** of these stages
- We reduce the complexity from *O(KN)* to *O(K log N)* (N: Number of data points, K: Number of clusters)
- We present the first extensive performance study that includes various optimizations and parallelization



Three stages of EM

EM algorithm

Gaussian Mixture Models (GMM)

$$p(x|\theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

- Samples distribution in many applications
- EM is a popular algorithm for fitting GMM parameters

Expectation Maximization (EM)

- Initialize parameters
- Repeat until convergence of Log-likelihood
  - **E-step**
  - **M-step**
  - **Log-likelihood**

EM using kd-tree

- We use kd-tree based algorithm to approximate and discard regions of space to reduce the asymptotic complexity of EM
- Traversing the kd-tree for EM includes 4 operations:
  - **BaseCase:** Direct point-to-point distance computation
  - **Prune:** Check to see if the node can be approximated
  - **CentroidCase:** Centroid approximation of the pruned node
  - **VisitOrder**: Order of tree traversal

TreeTraversal($\mathcal{N}_r$, q)

Input: reference node $\mathcal{N}_r$, query point q, rules closure $\mathcal{R}$.
  if $\mathcal{R}$.Prune($\mathcal{N}_r$, q) **then**
    return R.CentroidCase($\mathcal{N}_r$, q)
  if $\mathcal{N}_r$ is leaf **then**
    return $\mathcal{R}$.BaseCase($\mathcal{N}_r$, q)
else
    $\mathcal{C}_r = \mathcal{R}$.VisitOrder($\mathcal{N}_r$, q)
    for all $\mathcal{N}_{rc} \in \mathcal{C}_r$ **do**
      TreeTraversal($\mathcal{N}_{rc}$, q)

PEAK

We present a new *high-performance* parallel algorithm on multicore systems for EM:

- Tree-based approximation algorithm for E-step (previous work)
- Tree-based approximation algorithm for computing the Log-likelihood (our contribution)
- Optimizations and multicore parallelization (our contribution)

E-step Algorithm

- Initializing the parameters of Gaussians
- **BaseCase:** Computation only for points in the leaf nodes
- **Prune**: Check the below criteria: (*r* represents the responsibility)

$$(r_{i,max} - r_{i,min}) < \beta r_{i,total} \;\; (i = 1, .., K)$$

- **CentroidCase**: E-step computation for center of hyper-rectangle

Log-likelihood Algorithm

- Initializing the Log-likelihood
- **BaseCase**: Computing the Log-likelihood only for points in the leaf
- **Prune**: Check the below criteria:

$$log \sum_{i=1}^{K} \pi_i \mathcal{N}(x_{max}|\theta_i) - log \sum_{i=1}^{K} \pi_i \mathcal{N}(x_{min}|\theta_i) < \alpha |log(\sum_{i=1}^{K} \pi_i \mathcal{N}(x_{mean}|\theta_i))| \;\; [\alpha = 0.1]$$

- **CentroidCase**: Log-likelihood computation for center of hyper-rectangle

Parallelization

- Parallelizing the post-order tree traversal for both stages
- Using Cilk work-stealing scheduler
- Task-level parallelism (*cilk_spawn*)

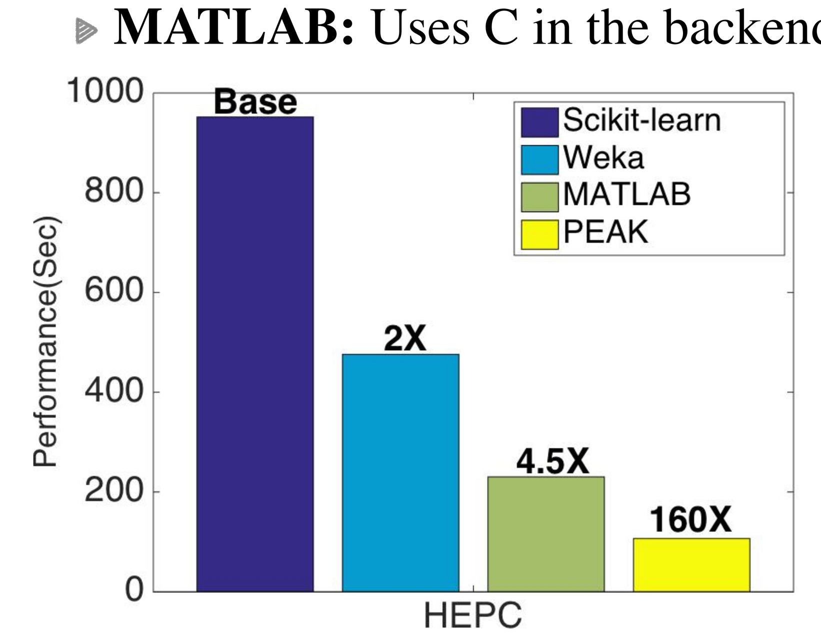
Optimization

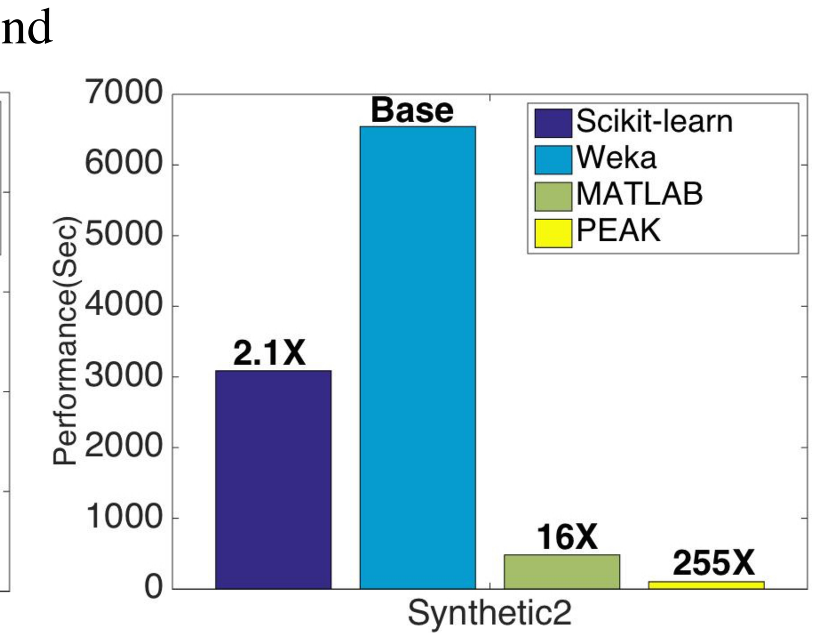
- Compiler optimizations
  - Loop fusion, inlining
- Numerical optimizations
  - Cholesky decomposition
  - Forward substitution

Comparison with other Libraries

We compare our algorithm against three state-of-the-art libraries:

- **Weka:** Waikato environment for data mining written in Java
- **Scikit-learn:** Python module built on top of numPy and sciPy
- **MATLAB:** Uses C in the backend



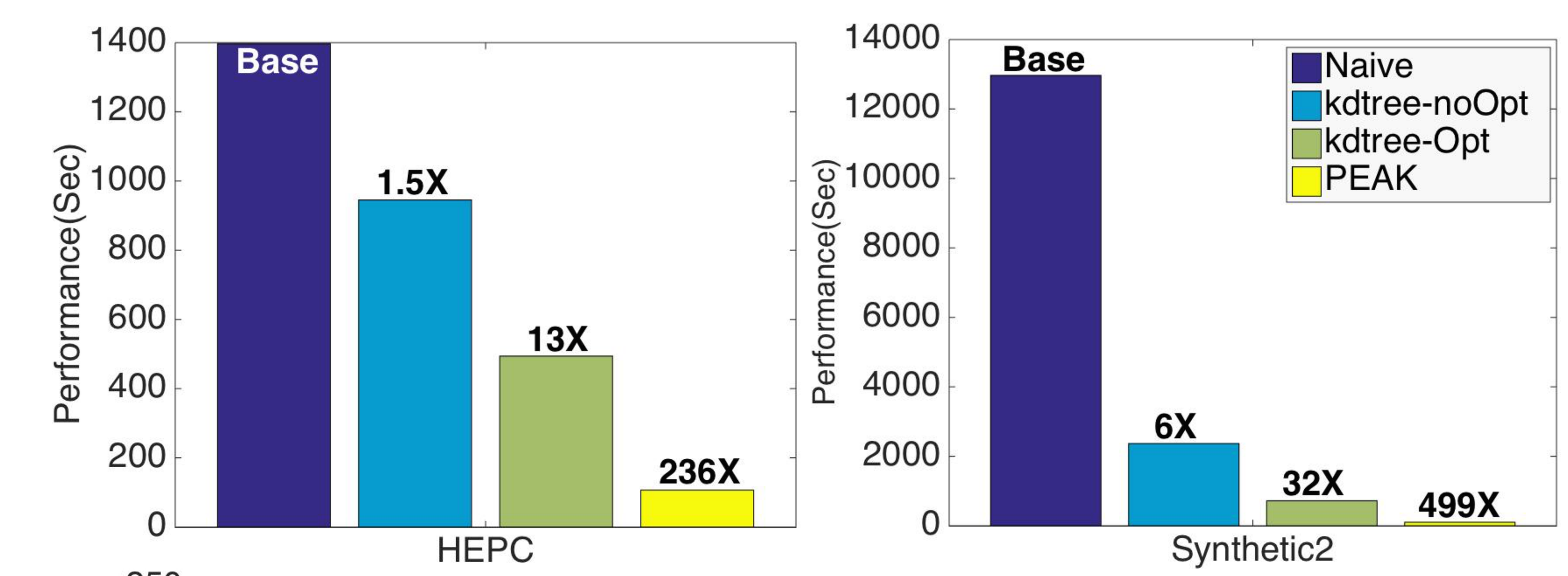


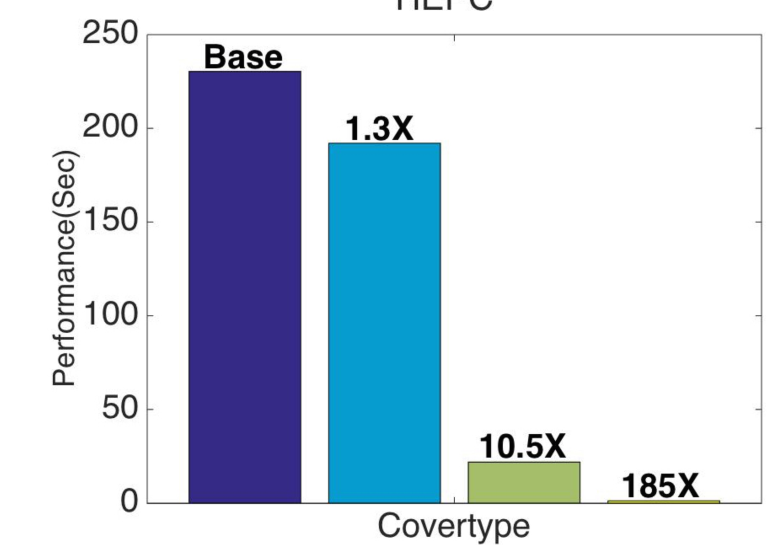
Performance Results

Summary of benchmark datasets

Datasets	*N*	*dim*	*clusters*
SkinSeg	245057	3	2
Covertype	581012	5	15
HEPC	2075259	4	8
Synthetic2	1000000	2	64
Synthetic1	1000000	4	4

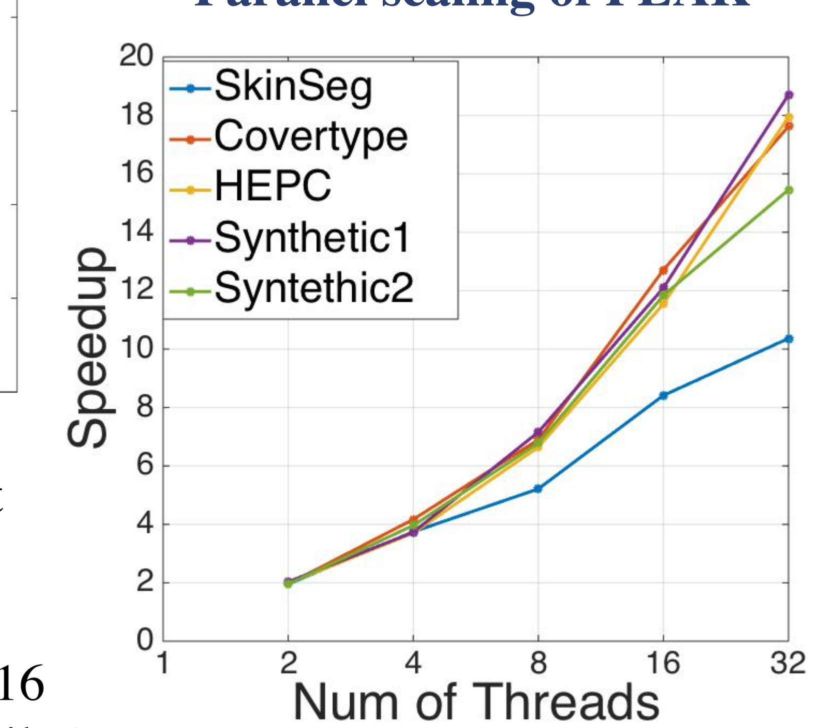
Speedup over the naive baseline code for three datasets





***System spec:** Dual-socket Intel Xeon E5-2630 v3 processor (Haswell). Each socket has 8 cores, total of 16 cores (with Intel C++ compiler).

Parallel scaling of PEAK



Conclusion & Future Work

We introduced a parallel EM algorithm using the same tree for all the stages:

- Our result shows up to **500x** speedups on real world and synthetic datasets.
- We will extend this idea to larger classes of similar machine learning algorithms such as nearest neighbors, density estimation, and range search.
- We are currently building a N-body DSL and code generator that will provide a high-level interface with high performance on target platforms for domain scientists.

References

[1] A. W. Moore. Very fast EM-based mixture model clustering using multiresolution KD-trees. Advances in Neural information processing systems, pages 543–549, 1999.

[2] S.-K. Ng and G. J. McLachlan. Speeding up the EM algorithm for mixture model-based segmentation of magnetic resonance images. Pattern Recognition, 37(8):1573–1589, 2004.

[3] N. Kumar, S. Satoor, and I. Buck. Fast parallel expectation maximization for Gaussian mixture models on GPUs using CUDA. In High Performance Computing and Communications, 2009. HPCC'09.