Efficient Evidence Accumulation Clustering for large datasets / big data

Diogo Alexandre Oliveira Silva
ALFAL / ENGEL

Academia da Força Aérea / Instituto Superior Técnico

November, 2015

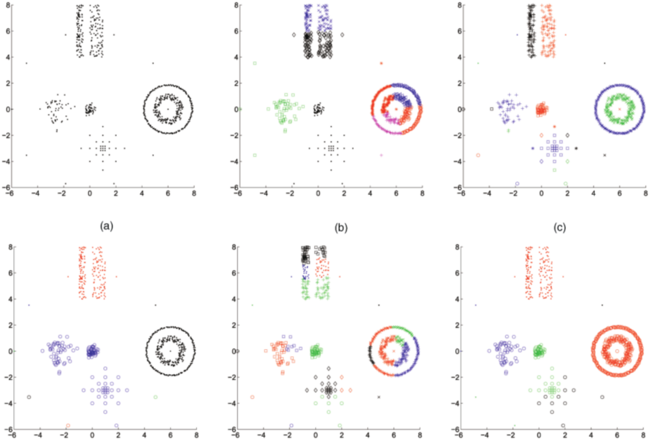
Contents

What is clustering?

"The goal of data clustering, also known as cluster analysis, is to discover the natural grouping(s) of a set of patterns, points, or objects." [1]

[1] A. K. Jain, "Data clustering: 50 years beyond K-means," Pattern Recognition Letters, vol. 31

What is clustering?

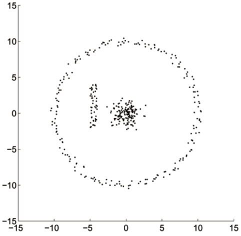


Source: A. N. L. Fred and A. K. Jain, “Combining multiple clusterings using evidence accumulation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 27

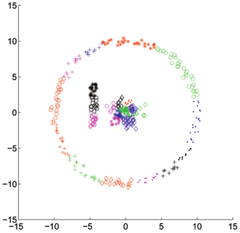
Evidence Accumulation Clustering

- State-of-the-art
- Robust
- Ensemble method

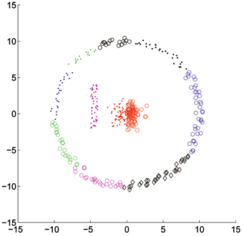
EAC: Production



(a)

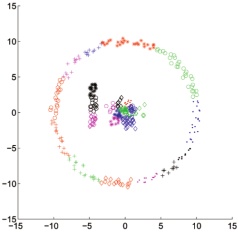


(b)

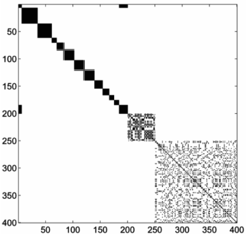


(c)

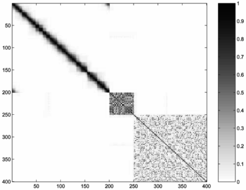
EAC: Combination



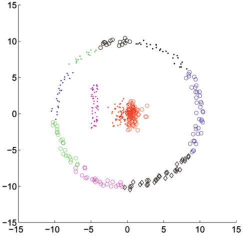
(a)



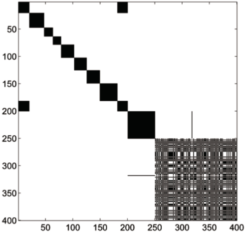
(c)



(e)

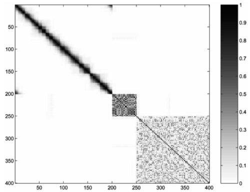


(b)

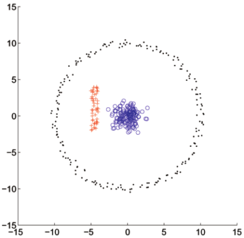


(d)

EAC: Recovery



(a)

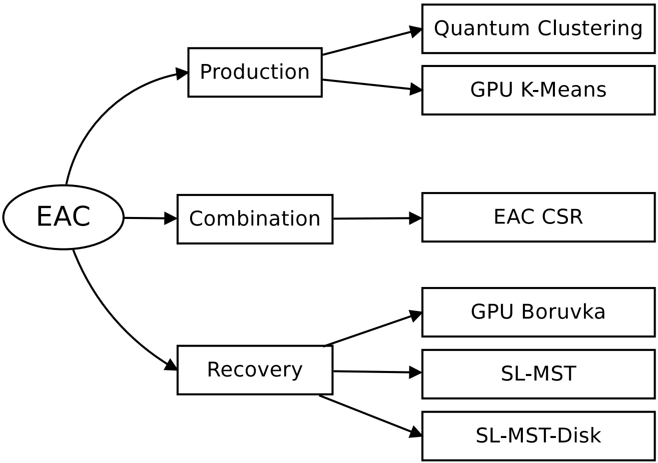


(b)

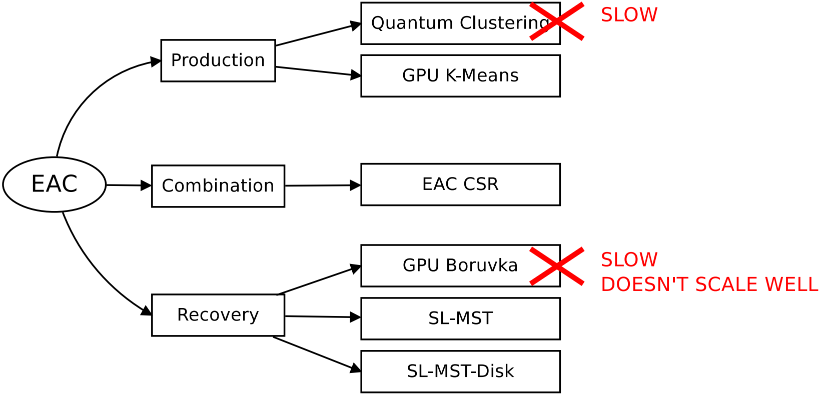
Goals

- Devise strategies to reduce computation and memory complexities of EAC.
- Validation of Big Data EAC on real data.
- Application of Evidence Accumulation Clustering to Big Data.
- Application of EAC to real-world large datasets.

Proposed solution: overview



Proposed solution: final



Optimizing production of ensemble

- K-Means is simple and has been used with EAC before with success.
- K-Means:
  - **labelling**: find the closest centroid to each data pattern
  - **update**: new centroids are the mean of the assigned patterns
- Number of clusters from interval $[K_{min}, K_{max}]$
- Optimization through parallelization in GPU.

GPGPU K-Means

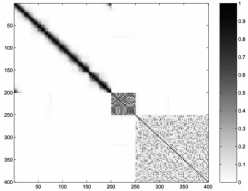
- CUDA GPGPU
  - Hundreds of processors
  - *Single instruction multiple thread* approach
  - Shared memory between threads

GPU K–Means

- GPU
  - Hundreds of processors
  - *Single instruction multiple thread* approach
  - Shared memory between threads
- Parallel K–Means
  1. Host sends data and centroids to GPU
  2. GPU threads compute the closest centroid to data patterns.
  3. The label and distance are stored and transfered to host.
  4. Host computes new centroids and sends them back to host.
  5. Repeat steps 2–4 until stopping criteria is met.

Optimizing combination

- Biggest challenge: $O(n^2)$ memory complexity
  - $200\,000$ dataset $\longrightarrow$ 149 GB
  - $1\,000\,000$ dataset $\longrightarrow$ 3725 GB
- Co-association matrix is:
  - symmetric (49.5% reduction)
  - sparse (association density as low as 1%)
- **exploiting sparsity**

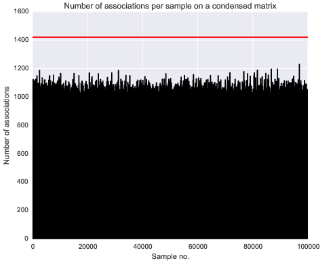


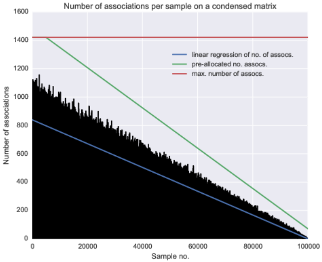
Optimizing combination

- Library solutions are either very slow or occupy too much memory
- Lack of literature on fast building
- Compromise between speed and memory:

EAC CSR

Optimizing combination





Optimizing final clustering

- Single-Link (SL) has been used before with success
- SL is a hierarchical agglomerative algorithm
- SL works on a pair-wise proximity matrix between all patterns
- SL is equivalent to a Minimum Spanning Tree (MST)
- Disk based MST variant to address very large co-association matrices

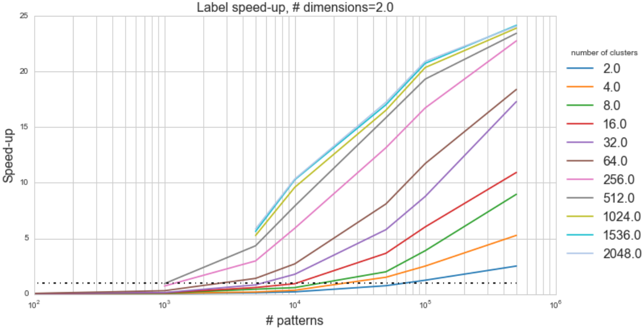
Validation

dataset	Difference between accuracies of implementations True number of clusters	Lifetime criteria
breast_cancer	4.948755e-06	2.825769e-06
ionosphere	1.652422e-06	1.452991e-06
iris	3.333333e-06	3.333333e-06
isolet	1.038861e-07	4.084904e-07
optdigits	3.795449e-06	1.480513e-06
pima	3.333333e-06	3.333333e-06
pima_norm	4.166667e-07	4.166667e-07
wine_norm	1.123596e-07	1.910112e-06

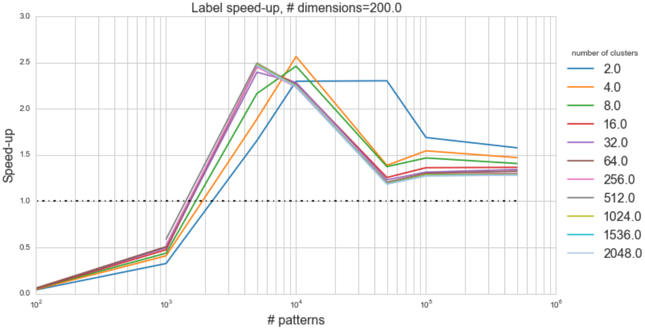
Speed-up over original version

dataset	Production	Combination	Recovery
breast_cancer	50.43974	7.544247	15.83316
ionosphere	21.86286	11.30883	19.97219
iris	19.76525	14.49562	28.50479
isolet	7.010007	6.183124	206.2837
optdigits	17.30209	10.2096	53.02636
pima	50.65624	141.4828	13.93502
pima_norm	54.25415	132.8632	14.355
wine_norm	22.92404	14.56994	25.27709

GPU K-Means


Label speed-up, # dimensions=2.0
Speed-up
patterns
number of clusters
2.0
4.0
8.0
16.0
32.0
64.0
256.0
512.0
1024.0
1536.0
2048.0

GPU K-Means


Label speed-up, # dimensions=200.0
Speed-up
patterns
3.0
2.5
2.0
1.5
1.0
0.5
0.0
10^2
10^3
10^4
10^5
10^6
number of clusters
2.0
4.0
8.0
16.0
32.0
64.0
256.0
512.0
1024.0
1536.0
2048.0

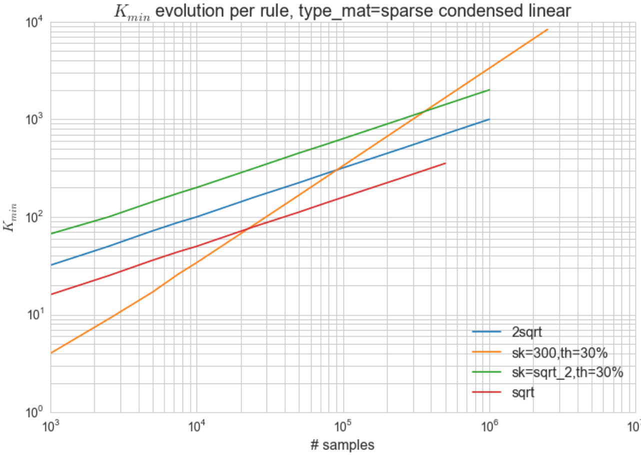
K_{min} rules

Rule	K_{min}	K_{max}
sqrt	$\frac{\sqrt{n}}{2}$	$\sqrt{n}$
2sqrt	$\sqrt{n}$	$2\sqrt{n}$
sk=sqrt2	$sk = \frac{\sqrt{n}}{2}$	$1.3K_{min}$
sk=300	$sk = 300$	$1.3K_{min}$

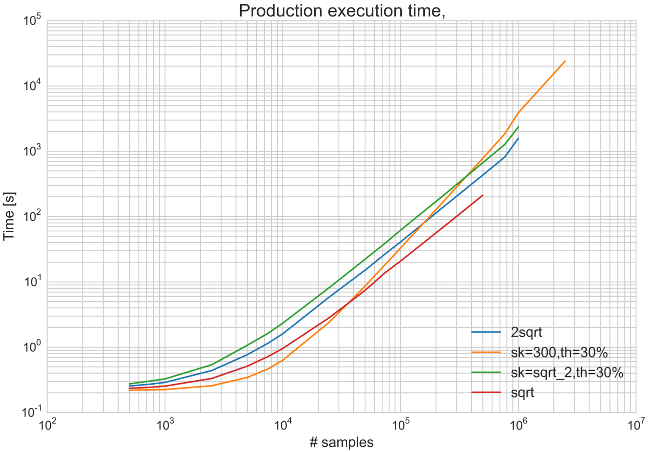
n - the number of samples

sk - the samples per cluster

K_{min} evolution

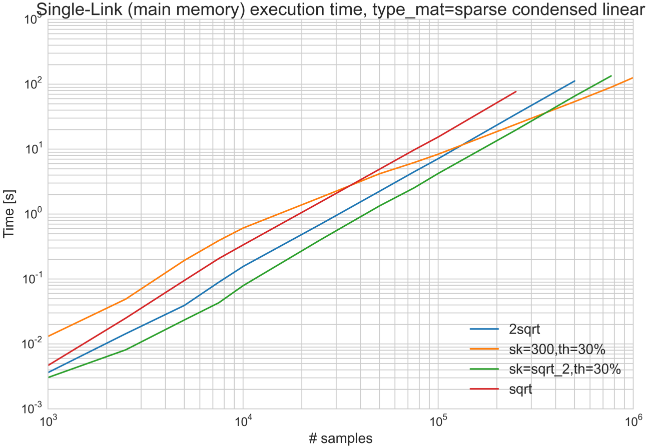


Production time

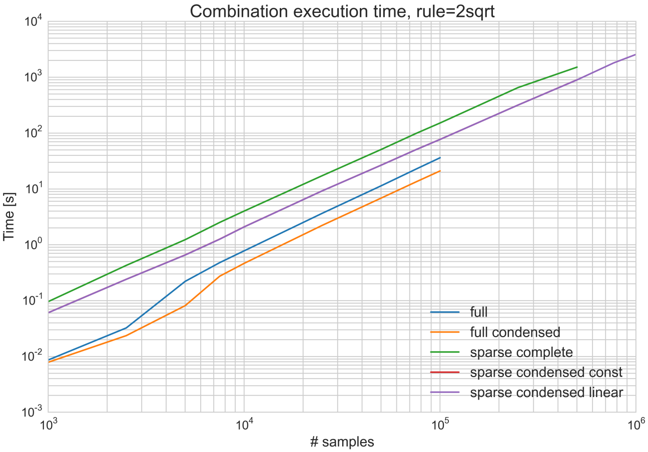


Combination time per rule

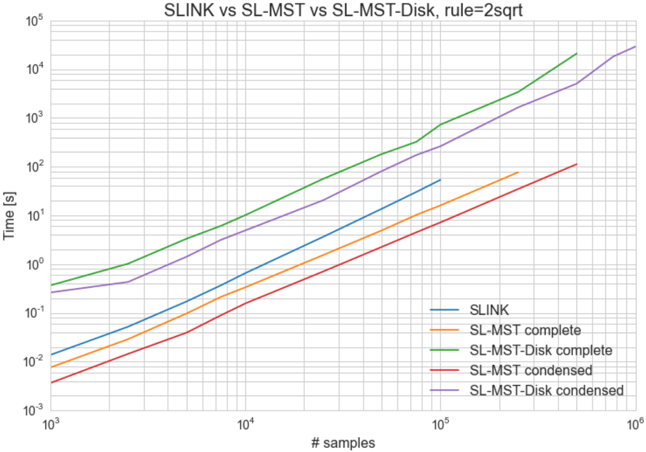
Single-Linkage times per rule



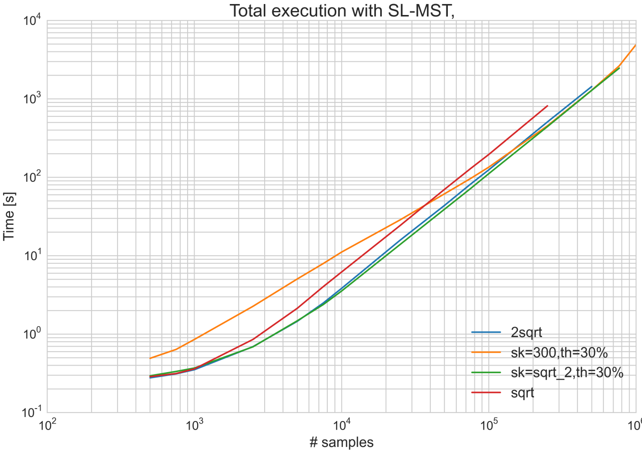
Combination time per matrix format



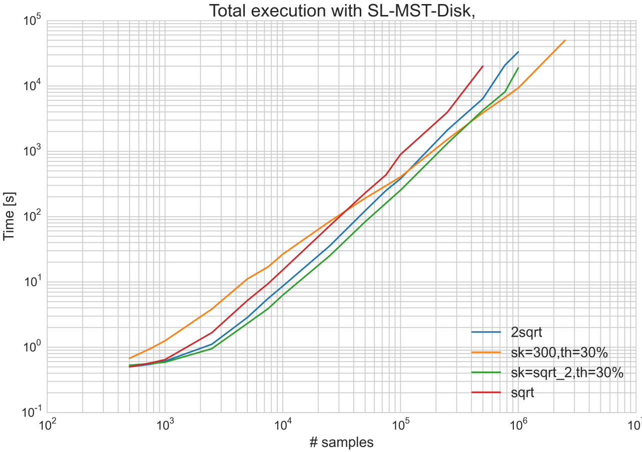
Single-Linkage times per matrix format



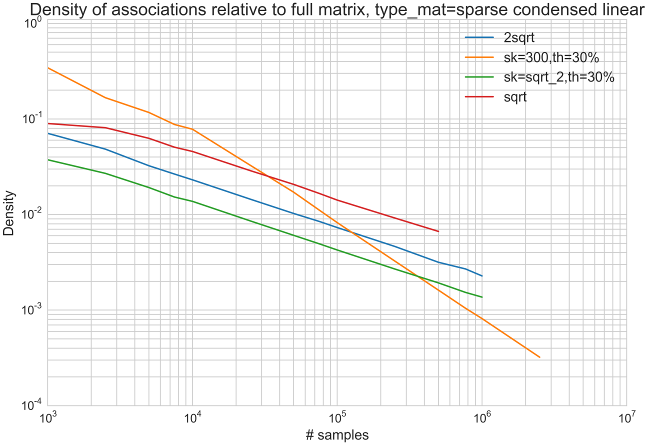
Total time: main memory



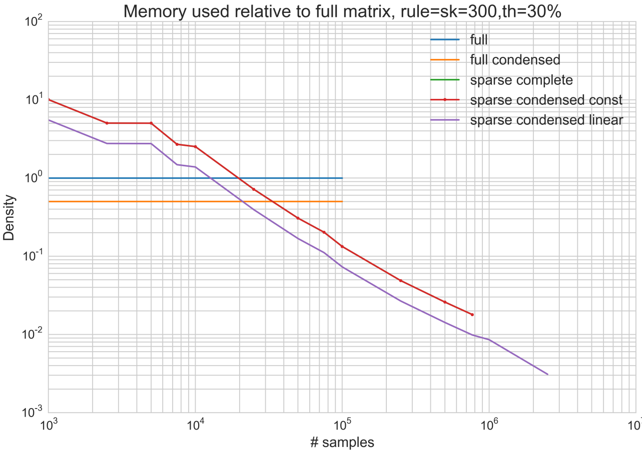
Total time: disk



Association density



Memory "density"



Conclusions

- Lessons learned: time vs. memory efficiency
- GPU is a valuable and widely widely available resource for data processing
- EAC was successfully scaled:
  - GPU K-Means
  - Efficient sparse matrix building
  - MST equivalence with SL
  - Disk-based algorithms
- Contributions:
  - Widen the range of EAC's applicability
  - New of way of building sparse matrix
  - New K_{min} rules for sparsity optimizations
- Plenty of possible extensions
- Selected parts of this work was compiled and submitted as a conference article