Scalable Big Data Clustering by Random Projection Hashing

Lee Carraher, Aniruddha Motia, Sayantan Dey and Philip A Wilsey (PI)
Dept of Electrical Engineering & Computing Systems, University of Cincinnati

Abstract
RPHash provides a solution to the approximate k-means clustering problem for very large distributed datasets. Distributed data models have gained popularity in recent years following the efforts of commercial, academic and government organizations, to make data more widely accessible. Due to the sheer volume of available data, in-memory single-core computation quickly becomes infeasible, requiring distributed multi-processing. Our solution achieves comparable clustering performance to other popular clustering algorithms, with improved overall complexity growth while being amenable to distributed processing frameworks such as MapReduce. Our solution also maintains certain guarantees regarding data privacy de-anonymization.

Applications
- Bio-Informatics
  - Medical Imaging
  - Computer Vision
- Recommender Systems
- Social Network Analysis
- Market Research

Key Features
- Scalability through Naive Parallelism
- Predictable Ram Time
- Low Intra-Process Communication
- Data Security
- Cloud Based Resource Allocation

Key Components
- Multi-Plane Random Projection
- Fast Johnson-Lindenstrauss Transform
- Locality Sensitive Hashing
- Universally Generative Lattices
- Lossy Frequency Counting

Clustering Performance Results

Multi Core Speedup
- RPHash exhibits near linear parallel speedup.
- Few sequential bottlenecks.

Runtime Results

Security via Random Projection
- Increased Data Security and RPHash Compliance Concerns Make Commercial Cloud Computing an Expensive Resource
- Random Projection, in RPHash can Provide Significant Resistance to De-Anonymization
- Probability of Re-association under Random Projection is exponentially dependent upon the difference in the projection dimension and original dimension

Open Source Software Release
Source available at:
- github.com/wilsey/phash-java
- github.com/wilsey/phash-golang
- github.com/wilsey/phash-reference-implementation
- github.com/wilsey/phash-spark

Figure: Probability of Vector Re-association for MIMIC II Biometric Signatures

Figure: Parallel Speedup as a Function of Number of Processors.

Figure: RPHash Distributed Spark Implementation
- Spark implementation of RPHash is based on 2-Phase RPHash
- Streaming RPHash Spark version in development

Human Activity Recognition Dataset
- Metric: 2Pass RPHash
- Metric: 10102.4 20091.7
- Dune Index: 0.08522 0.08779
- ULPendLocor Localized Dataset
- Metric: 2Pass RPHash
- Metric: 10102.4 20091.7
- Dune Index: 0.08522 0.08779

Figure: RPHash Algorithm Diagram

<table>
<thead>
<tr>
<th>Key Feature</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalability through Naive Parallelism</td>
<td>Predictable Ram Time</td>
</tr>
<tr>
<td>Low Intra-Process Communication</td>
<td>Data Security</td>
</tr>
<tr>
<td>Cloud Based Resource Allocation</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key Component</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Multi-Plane Random Projection</td>
<td>Fast Johnson-Lindenstrauss Transform</td>
</tr>
<tr>
<td>Locality Sensitive Hashing</td>
<td>Universally Generative Lattices</td>
</tr>
<tr>
<td>Lossy Frequency Counting</td>
<td></td>
</tr>
</tbody>
</table>

Figure: Streaming RPHash Algorithm Diagram
- foreach Key K do
- forall the v in K do
- if ||v-x||^2 ≤ R^2 then
- M[v] += 1
- end
- end
- end
- Result: RFReweightedCluster(M items)