Scalable Clustering Algorithm,

BIG DATA, MACHINE LEARNING AND SOCIAL NETWORK ANALYSIS, DECEMBER 16

M. Ghesmoune, T. Sarazin, M. Lebbah, H. Azzag
Outline

● Context

● Clustering using MapReduce

● Deal with large data sets such as streams

● Conclusion & Perspectives
Context

Difficulties:
- Structure
- Similarity measure?
- Number of clusters?
  (Combinatory)
- Validation (unlabeled data)
- Data types: categorical, mixed...

Visualization

Clustering

Exploration

Tutorial, IEEE BigData 2014
Two alternatives

Massive data mining as stream mining

Data

Algorithm LEArning

MapReduce / Spark

Algorithm LEA
Spark as an alternative

[Sparks et al ICDM 2013]
Clustering
Implementation: K-Means

data = spark.textFile("hdfs://...")
    .map(parsePoint)
centroids = Array(
    Point(randX(), randY()),
    Point(randX(), randY()))
Compute distance with prototypes

closestCentroid(p, centroids)
Assignment

\[ \text{closestCentroid}(p, \text{centroids}) \]
Assignment

closestCentroid(p, centroids)
Map - Assignments

```scala
val closest = data.map(p =>
  (closestCentroid(p, centroids),
   (p, 1))
)
```
Reduce - update of prototypes

```scala
val pointStats = closest.reduceByKey{
  case ((p1, sum1), (p2, sum2)) =>
    (p1 + p2, sum1 + sum2)
}
```
Iteration 1

```scala
val pointStats = closest.reduceByKey{
  case ((p1, sum1), (p2, sum2)) =>
  (p1 + p2, sum1 + sum2)
}
pointStats.foreach{
  case (id, value) =>
  centroids(id) = value._1 / value._2
}
```
Iteration 2

```scala
for (i <- 1 until 10) {
    val closest = data.map(p =>
        (closestCentroid(p, centroids),
        (p, 1))
    )
    val pointStats = closest.reduceByKey{
        case ((p1, sum1), (p2, sum2)) =>
        (p1 + p2, sum1 + sum2)
    }
    pointStats.foreach{
        case (id, value) =>
            centroids(id) = value._1 / value._2
    }
}
```
Topological Map

Why?

- Topological organization
- Generalization of K-means
- Adapted to MapReduce (batch version)
- Used for visualisation
- Used for exploration phase
MapReduce / Spark

SOM

Assignment
map₁

Quantization
Reduce

Row assignments
map₁

Quantization
Reduce

Column assignment
map₂

BiTM

https://github.com/TugdualSarazin/spark-clustering
BiTM-MapReduce-SPARK

2 millions, 20 variables

2 millions, 40 variables
Two alternatives

Massive data mining as stream mining
Big Data as data stream

- A sequence $x_1, \ldots, x_n$ of observations
  - Potentially infinite
  - Non-stationary
- Have to be processed in this order in one pass
- Random access is not allowed
- Restriction in memory
Big Data as data stream

Framework online-offline of clustering data streams
Big Data as data stream

Framework online-offline of clustering data streams
G-STREAM : GNG + Data Stream

GNG : [Fritzke 95]
- Evolutive topology
- Number of cells is not fixed
### G-STREAM and others

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>based on</th>
<th>topology</th>
<th>WL</th>
<th>phases</th>
<th>remove</th>
<th>merge</th>
<th>split</th>
<th>fade</th>
</tr>
</thead>
<tbody>
<tr>
<td>G-Stream</td>
<td>NGas</td>
<td>✓</td>
<td>✓</td>
<td>online</td>
<td>✓</td>
<td>×</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>AING</td>
<td>NGas</td>
<td>✓</td>
<td>×</td>
<td>online</td>
<td>×</td>
<td>✓</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>CluStream</td>
<td>k-means</td>
<td>×</td>
<td>×</td>
<td>2 phases</td>
<td>✓</td>
<td>offline</td>
<td>×</td>
<td>×</td>
</tr>
<tr>
<td>DenStream</td>
<td>DbScan</td>
<td>×</td>
<td>×</td>
<td>2 phases</td>
<td>✓</td>
<td>offline</td>
<td>×</td>
<td>✓</td>
</tr>
<tr>
<td>SOStream</td>
<td>DbScan, SOM</td>
<td>×</td>
<td>×</td>
<td>online</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>E-Stream</td>
<td>k-means</td>
<td>×</td>
<td>×</td>
<td>2 phases</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>StreamKM++</td>
<td>k-means</td>
<td>×</td>
<td>×</td>
<td>2 phases</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>StrAP</td>
<td>AP</td>
<td>×</td>
<td>×</td>
<td>2 phases</td>
<td>✓</td>
<td>×</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>SVStream</td>
<td>SVC, SVDD</td>
<td>×</td>
<td>×</td>
<td>online</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
G-Stream: characteristics

- No initialization phase of the model,
- A graph representing the topological structure,
- Creating multiple nodes at the same time,
- **One single stage** (online), (no offline stage)
- Use of a reservoir.
## Data sets

<table>
<thead>
<tr>
<th>Datasets</th>
<th>#records</th>
<th>#features</th>
<th>#classes</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS1</td>
<td>9,153</td>
<td>2</td>
<td>14</td>
</tr>
<tr>
<td>DS2</td>
<td>5,458</td>
<td>2</td>
<td>13</td>
</tr>
<tr>
<td>letter4</td>
<td>9,344</td>
<td>2</td>
<td>7</td>
</tr>
<tr>
<td>Sea</td>
<td>60,000</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>HyperPlan</td>
<td>100,000</td>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>KddCup99</td>
<td>494,021</td>
<td>41</td>
<td>23</td>
</tr>
<tr>
<td>CoverType</td>
<td>581,012</td>
<td>54</td>
<td>7</td>
</tr>
<tr>
<td>Sensor</td>
<td>2,219,803</td>
<td>5</td>
<td>54</td>
</tr>
</tbody>
</table>
G-Stream: Example
G-Stream on letter4
G-Stream on DS1
G-Stream on DS2
G-Stream vs GNG online: accuracy

(a) DS1

(b) DS2

(c) letter4

(d) HyperPlan

(e) Sea
G-Stream vs GNG online: RMS Error

(a) DS1
(b) DS2
(c) letter4
(d) HyperPlan
(e) Sea
G-Stream vs GNG online: #Nodes

(a) DS1  (b) DS2  (c) letter4  (d) HyperPlan  (e) Sea
## Accuracy

<table>
<thead>
<tr>
<th>Datasets</th>
<th>G-Stream</th>
<th>StreamKM++</th>
<th>DenStream</th>
<th>ClusTree</th>
</tr>
</thead>
<tbody>
<tr>
<td>DS1</td>
<td>0.9809±0.0061</td>
<td>0.6754±0.0183</td>
<td>0.7740±0.0000</td>
<td>0.6864±0.0275</td>
</tr>
<tr>
<td>DS2</td>
<td>0.8632±0.0075</td>
<td>0.6261±0.0360</td>
<td>0.7190±0.0000</td>
<td>0.6220±0.0000</td>
</tr>
<tr>
<td>letter4</td>
<td>0.9832±0.0050</td>
<td>0.6871±0.0263</td>
<td>0.8110±0.0000</td>
<td>0.8110±0.0000</td>
</tr>
<tr>
<td>Sea</td>
<td>0.8386±0.0021</td>
<td>0.7886±0.0091</td>
<td>0.8240±0.0001</td>
<td>0.8224±0.0065</td>
</tr>
<tr>
<td>HyperPlan</td>
<td>0.4238±0.0021</td>
<td>0.3966±0.0055</td>
<td>0.4250±0.0000</td>
<td>0.4380±0.0089</td>
</tr>
<tr>
<td>KddCup99</td>
<td>0.9805±0.0050</td>
<td>0.6922±0.1140</td>
<td>0.9544±0.0031</td>
<td>0.8182±0.1304</td>
</tr>
<tr>
<td>CoverType</td>
<td>0.6085±0.0087</td>
<td>0.5266±0.0074</td>
<td>0.5850±0.0011</td>
<td>0.5850±0.0000</td>
</tr>
<tr>
<td>Sensor</td>
<td>0.0834±0.0002</td>
<td>0.0561±0.0014</td>
<td>0.0660±0.0000</td>
<td>0.0790±0.0000</td>
</tr>
<tr>
<td>Datasets</td>
<td>G-Stream</td>
<td>StreamKM++</td>
<td>DenStream</td>
<td>ClusTree</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
<td>-------------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>DS1</td>
<td>0.7289±0.0113</td>
<td>0.7021±0.0209</td>
<td>0.6973±0.0000</td>
<td>0.7064±0.0168</td>
</tr>
<tr>
<td>DS2</td>
<td>0.6700±0.0054</td>
<td>0.6242±0.0182</td>
<td>0.6228±0.0000</td>
<td>0.6231±0.0000</td>
</tr>
<tr>
<td>letter4</td>
<td>0.6265±0.0064</td>
<td>0.5532±0.0219</td>
<td>0.1637±0.0000</td>
<td>0.2425±0.0000</td>
</tr>
<tr>
<td>Sea</td>
<td>0.1380±0.0009</td>
<td>0.1463±0.0042</td>
<td>0.1646±0.0000</td>
<td>0.1583±0.0095</td>
</tr>
<tr>
<td>HyperPlan</td>
<td>0.0186±0.0009</td>
<td>0.0103±0.0023</td>
<td>0.0208±0.0000</td>
<td>0.0170±0.0042</td>
</tr>
<tr>
<td>KddCup99</td>
<td>0.6670±0.0089</td>
<td>0.3926±0.2815</td>
<td>0.6290±0.0300</td>
<td>0.5724±0.2974</td>
</tr>
<tr>
<td>CoverType</td>
<td>0.1403±0.0029</td>
<td>0.0874±0.0086</td>
<td>0.0475±0.0201</td>
<td>0.0362±0.0042</td>
</tr>
<tr>
<td>Sensor</td>
<td>0.1154±0.0012</td>
<td>0.0795±0.0038</td>
<td>0.3087±0.0000</td>
<td>0.3238±0.0000</td>
</tr>
</tbody>
</table>
Conclusion & perspectives

- Data set size has increased significantly
  - MapReduce is crucial for some algorithms
  - Deal with large data sets such as streams

- New approach
  - Resampling & Sketching
  - Boosting & bagging [Kleiner et al ICML 2012]
References


