SS-ZG548: ADVANCED DATA MINING

Lecture-12: Clustering on Data Stream, Big Data

Dr. Kamlesh Tiwari
Assistant Professor
Department of Computer Science and Information Systems Engineering,
BITS Pilani, Rajasthan-333031 INDIA

March 10, 2018 (WILP @ BITS-Pilani Jan-Apr 2018)

Clustering over Evolving Data Stream (DenStream)

Clustering over Evolving Data Stream

With limited memory and one-pass constraint one want to determine arbitrary number of clusters of arbitrary shape by efficiently handling outliers.

Use DenStream [1]

- Damped window model: Weights with time t are $f(t) = 2^{-\lambda.t}$. (other models landmark and sliding window)
- core-micro-cluster, potential-micro-cluster and outlier-micro-cluster structures
- Guarantees the precision of the weights of the micro-clusters

[1] Cao, Feng and Ester, Martin and Qian, Weining and Zhou, Aoying, "Density-Based Clustering over an Evolving Data Stream with Noise.", in International Conference on Data Mining, pages 328–339, vol 6, SIAM, 2006

Clustering over Evolving Data Stream

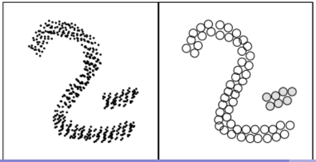
- Beside limited memory and one-pass constraints, we require
  - No assumption on the number of clusters,
  - Discovery of clusters with arbitrary shape and
  - Ability to handle outliers
- DenStream [2], is a new approach for discovering clusters in an evolving data stream.
- Uses core-micro-cluster to summarize the clusters
- Along with potential core-micro-cluster and outlier micro-cluster structures
- Designed a pruning strategy that guarantees the precision of the weights of the micro-clusters
- User damped window model. Weights with time t are $f(t) = 2^{-\lambda . t}$. (other models landmark and sliding window)

---

[2] Cao, Feng and Ester, Martin and Qian, Weining and Zhou, Aoying, "Density-Based Clustering over an Evolving Data Stream with Noise.", in International Conference on Data Mining, pages 328–339, vol 6, SIAM, 2006

Clustering over Evolving Data Stream

- **Definition (core object)** It is an object in whose ϵ neighborhood the overall weight of data points is at least μ
- **Definition (density-area)** A density area is defined as the union of the ϵ neighborhoods of core objects
- **Definition (core-micro-cluster)** At time t it is defined as CMC(w, c, r) for a group of close points $p_{i_1}, p_{i_2}, ..., p_{i_n}$ with time stamp $T_{i_1}, T_{i_2}, ..., T_{i_n}$. $w = \sum_{j=1}^{n} f(t - T_{i_j})$, $w \geq \mu$ is the weight. $c = \frac{\sum_{j=1}^{n} f(t-T_{i_j}) p_{i_j}}{w}$ is the center, $r = \frac{\sum_{j=1}^{n} f(t-T_{i_j}) dist(p_{i_j}, c)}{w}$, $r \leq \epsilon$ is the radius, where $dist(p_{i_j}, c)$ denotes the Euclidean distance between point p_{i_j} and center c
- When a clustering request arrives, each c-micro-cluster will be labeled to get the final result.



Clustering over Evolving Data Stream

Definition (potential-micro-cluster) A potential-micro-cluster (or p-micro-cluster) at time t for a group of close points $p_{i_1}, p_{i_2}, ..., p_{i_n}$ with time stamp $T_{i_1}, T_{i_2}, ..., T_{i_n}$ is defined as $\{\overline{CF^1}, \overline{CF^2}, w\}$.
$w = \sum_{j=1}^{n} f(t - T_{i_j})$, $w \geq \beta\mu$ is the weight. β, $0 < \beta \leq$, is the parameter to determine the threshold of outlier relative to c-micro-clusters.
$\overline{CF^1} = \sum_{j=1}^{n} f(t - T_{i_j})p_{i_j}$ is the weighted linear sum of the points.
$\overline{CF^2} = \sum_{j=1}^{n} f(t - T_{i_j})p_{i_j}^2$ is the weighted squired linear sum of the points. Centre of p-micro-cluster is $c = \frac{\overline{CF^1}}{w}$ and the radius of a p-micro-cluster $r = \sqrt{\frac{\overline{CF^2}}{w} + (\frac{\overline{CF^1}}{w})^2}$, $r \leq \epsilon$

Clustering over Evolving Data Stream

Definition (outlier-micro-cluster) An outlier-micro-cluster (or o-micro-cluster) at time t for a group of close points $p_{i_1}, p_{i_2}, ..., p_{i_n}$ with time stamp $T_{i_1}, T_{i_2}, ..., T_{i_n}$ is defined as $\{\overline{CF^1}, \overline{CF^2}, w, t_0\}$. The definition of $w, \overline{CF^1}, \overline{CF^2}$ center and radius are the same as p-micro-cluster. $t_0 = T_{i_1}$, denotes the creation time of o-micro-cluster which is used to define the life span of o-micro-cluster. However $w < \beta\mu$.

Note: p-micro-cluster and o-micro-cluster can be maintained incrementally.

Clustering Algorithm has two parts:

- Online part of micro-cluster maintenance
- Offline part of generation of final clusters, on demand of user

Merging of P

p-micro-clusters and o-micro-clusters are maintained in an online way.

Algorithm 1 Merging (p)

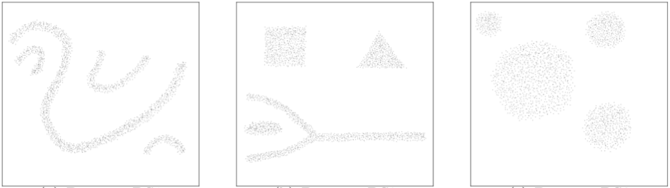
Try to merge p into its nearest p-micro-cluster c_p;
if r_p (the new radius of c_p) $\leq \epsilon$ **then**
    Merge p into c_p;
else
    Try to merge p into its nearest o-micro-cluster c_o;
    if r_o (the new radius of c_o) $\leq \epsilon$ **then**
        Merge p into c_o;
        if w (the new weight of c_o) $> \beta\mu$ **then**
            Remove c_o from outlier-buffer and create a new p-micro-cluster by c_o;
        end if
    else
        Create a new o-micro-cluster by p and insert it into the outlier-buffer;
    end if
end if

DenStream Algorithm

Algorithm 2 DenStream $(DS, \epsilon, \beta, \mu, \lambda)$

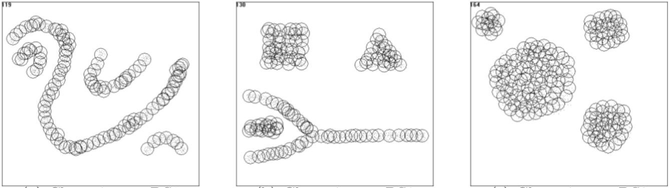
$T_p = \lceil \frac{1}{\lambda} \log(\frac{\beta\mu}{\beta\mu - 1}) \rceil$;
Get the next point p at current time t from data stream DS;
Merging(p);
if (t mod T_p)=0 **then**
    for each p-micro-cluster c_p **do**
        if w_p(the weight of c_p)$< \beta\mu$ **then**
            Delete c_p;
        end if
    end for
    for each o-micro-cluster c_o **do**
        $\xi = \frac{2^{-\lambda(t - t_o + T_p)} - 1}{2^{-\lambda T_p} - 1}$;
        if w_o(the weight of c_o)$< \xi$ **then**
            Delete c_o;
        end if
    end for
end if
if a clustering request arrives **then**
    Generating clusters;
end if

Performance



(a) Data set DS1 (b) Data set DS2 (c) Data set DS3

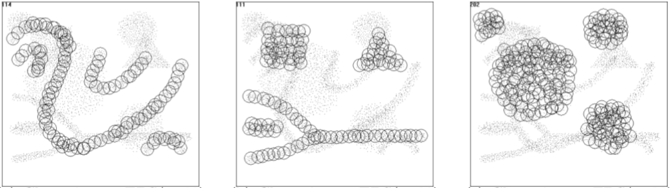
Figure 4: Synthetic data sets



(a) Clustering on DS1 (b) Clustering on DS2 (c) Clustering on DS3

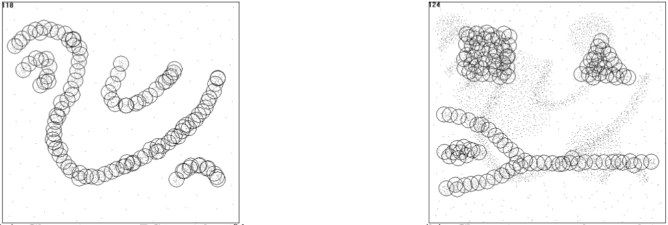
Figure 5: Clustering on DS1, DS2 and DS3

Performance



(a) Clustering on EDS(t=10) (b) Clustering on EDS(t=20) (c) Clustering on EDS(t=30)

Figure 6: Clustering on the evolving data stream EDS



(a) Clustering on DS1 with 5% noise (b) Clustering on the evolving stream EDS with 5% noise(t=20)

Figure 7: Clustering on data streams with noise

Performance

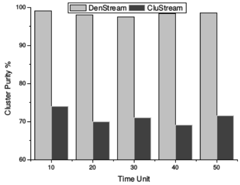


Figure 8: Clustering quality(EDS data stream, horizon=2, stream speed=2000)

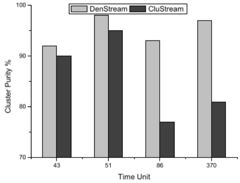


Figure 10: Clustering quality(Network Intrusion da set, horizon=1, stream speed=1000)

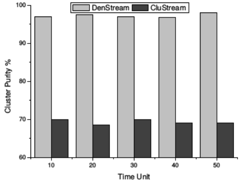


Figure 9: Clustering quality(EDS data stream, horizon=10, stream speed=1000)

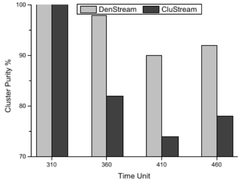


Figure 11: Clustering quality(Network Intrusion da set, horizon=5, stream speed=1000)

Performance

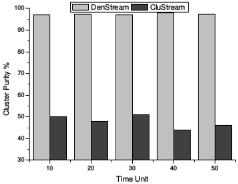


Figure 12: Clustering quality(EDS data stream with 1% noise, horizon=2, stream speed=2000)

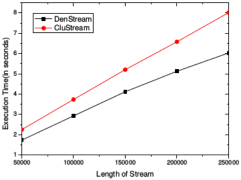


Figure 14: Execution time vs. length of stream(Netw
Intrusion data set)

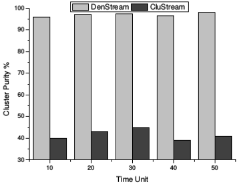


Figure 13: Clustering quality(EDS data stream with 5% noise, horizon=10, stream speed=1000)

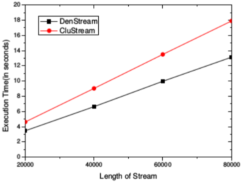


Figure 15: Execution time vs. length
stream(Charitable Donation data set)

Performance

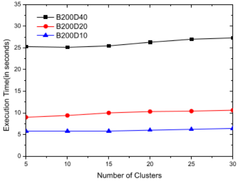


Figure 16: Execution time vs. number of clusters

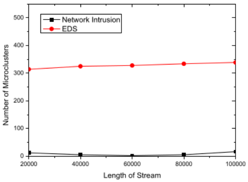


Figure 18: Memory usage

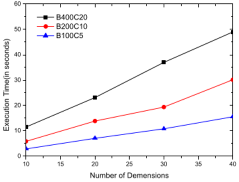


Figure 17: Execution time vs. dimensionality

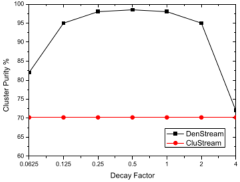
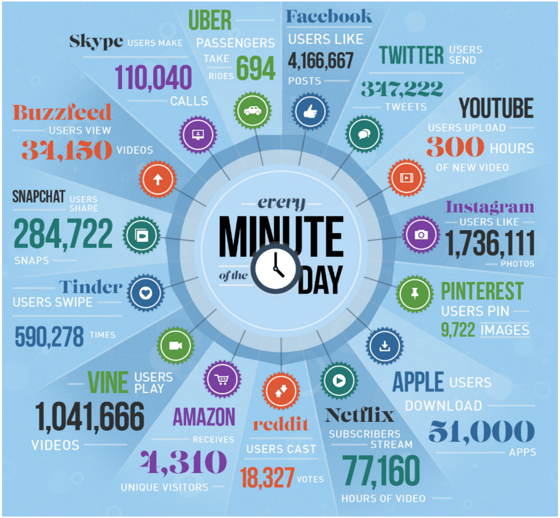


Figure 19: Clustering quality vs. decay factor λ

Big Data

What is big data?[3]

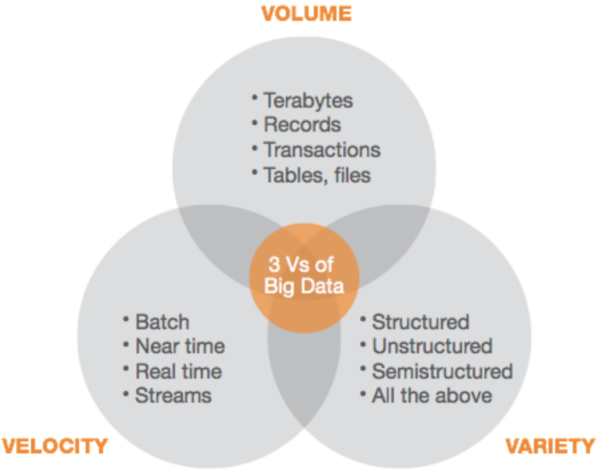
- 2.5 quintillion (1×10^{18}) bytes of data/day
- 90% generated in last 2 year. 80% of it is unstructured.
- Facebook, 4 million posts/min
- Aircraft: each engine generates 10 TB data in 30 min. UK to New York would generate 640 TB of data



[3] http://wersm.com/how-much-data-is-generated-every-minute-on-social-media/

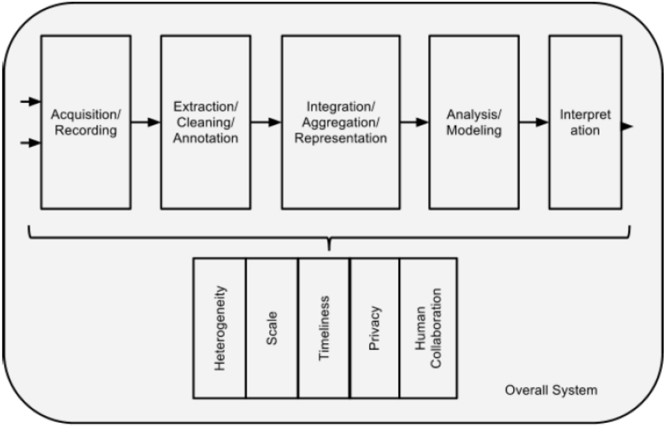
Big Data

3V's: The "BIG" in big data is not only for volume



Big Data

Major steps in analysis of big data: [4]



Solution is **Distributed computing**

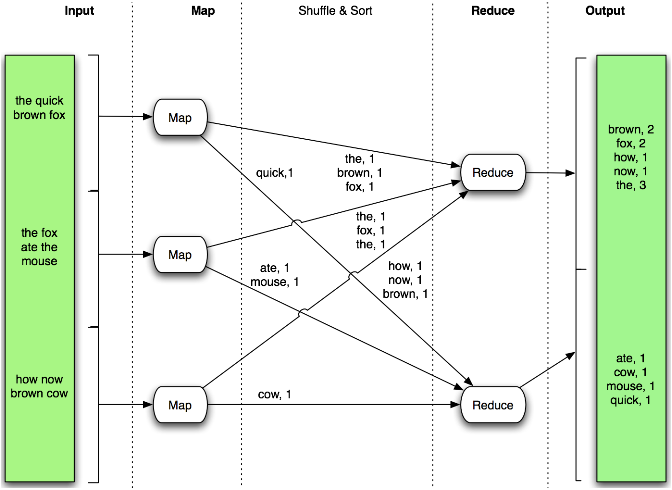
[4] "Challenges and Opportunities with Big Data" A community white paper developed by leading researchers across the United States

Big Data

- **Grid** (hetomogeneous) or **cluster** (homogeneous) computing
- Distributed computing **have to deal with** synchronization, deadlocks, data dependency, mutual exclusion, replication, reliability, platform scalability and provisioning
- Ready-made solution is **MapReduce/Hadoop**[5] or **spark** that provides a high level of abstraction for data parallel tasks
- Hadoop/PIG combo is very effective
- Need is there for a scalable distributed computing framework that provides both abstraction and performance (by exploiting all kinds of parallelisms that exist in an algorithm)

[5] MapReduce: Simplified Data Processing on Large Clusters Jeffrey Dean and Sanjay Ghemawat, OSDI, 2004

Big Data



map is a function that executes on each partition.

PK-Means

- **K-Means:** steps involved are
  1. Select Seed
  2. Assignment //(most compute intensive)
  3. Compute Centroid
- **PK-Means:** [6]
  - Input dataset is stored on GFS/HDFS. A sequence file of $< key, value >$ pairs. (Key: offset, Value: string of whole record)
  - Dataset is split and globally, and broadcast to all mappers

Map

  - Distance calculations are parallel executed
  - Each mapper has array of centers
  - Computes closest center for each sample
  - Intermediate values/output: $< key, value >$ (Key: index of the closed centre, value: sample)

[6] Du, Zhihua and Wang, Yiwei and Ji, Zhen, "PK-means: A new algorithm for gene clustering" in Computational Biology and Chemistry, pages=243–247, vol 32(4), Elsevier 2008

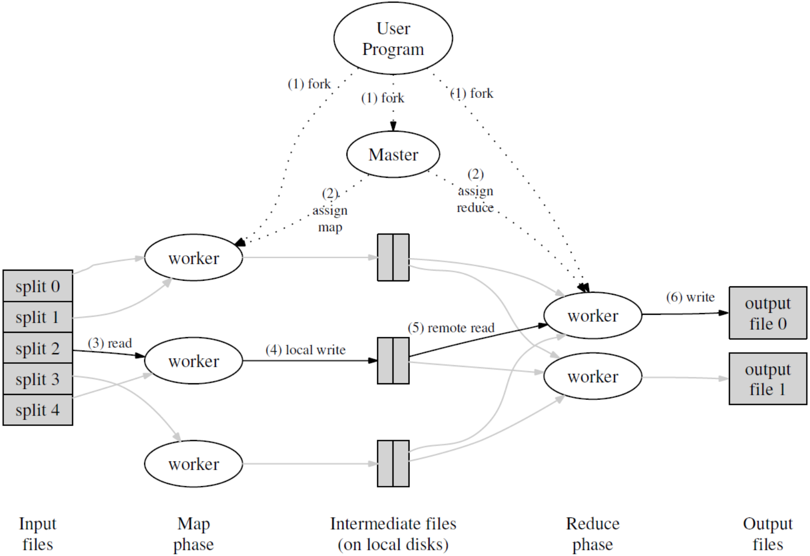
PK-Means

- **PK-Means:**
  Combiner
  - Combines intermediate data of each map task and stores locally
  - Partial sum the values assigned to the same cluster
    - Record number of samples in each cluster and
    - Sum of values at each dimension
  - Key: key & Value: string of num and sums

  Reducer
  - Input: output of combiner
  - Compute all the samples assigned to a center
  - Calculate new centers

Scalability is high

How MapReduce Works

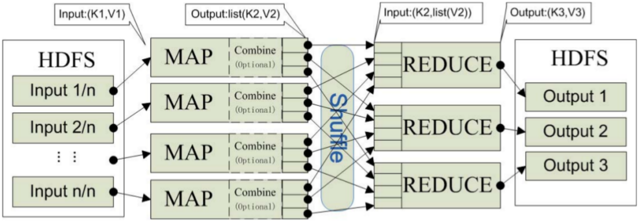


How MapReduce Works

- Map function invokes data partitioning across multiple machines
  - M map tasks and R reduce tasks
  - A map worker reads the corresponding input, parses key value pair and execute user defined map function
  - Intermediate key value pair is generated
  - Periodically buffered pairs are written to local disk, partitioned into R regions and locations of the partitions are passed to the master
- Reduce invocations partition intermediate key result into R pieces
  - Reduce worker is notified by the master about partitions
  - Remote procedure calls are used to read partitioned data
  - After reading data is sorted by reduce worker and user defined reduce function is execute
  - The output of the reduce function is appended in a file
- The number of partitions (R) and the partitioning function are specified by the user

MR-DBSCAN

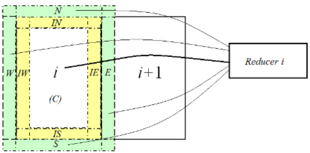
- MR-DBSCAN [7] involves following steps
  1. Preprocessing
  2. Local DBSCAN
  3. Find Merging Mapping
- Uses quadtree, a spacial data structure
- Extended regions (ϵ-extended) is taken in partition

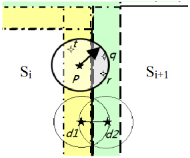


[7] MR-DBSCAN: An Efficient Parallel Density-based Clustering Algorithm using MapReduce 2011 IEEE 17th International Conference on Parallel and Distributed Systems

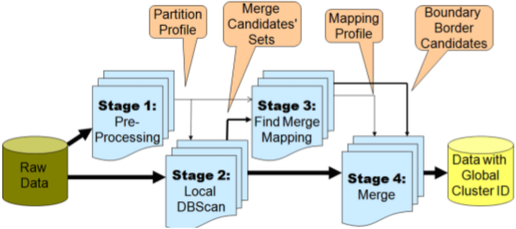
MR-DBSCAN

- Extended regions (ϵ-extended) is taken in partition





- Cross connection files are processed during reduce
- This makes the algorithm data parallel



Thank You!

Thank you very much for your attention!

Queries ?