DenseAlert: Incremental Dense-SubTensor Detection in Tensor Streams

Feb-03-2017

Kijung Shin (kijungs@cs.cmu.edu)

1 General Information

- Version: 1.0
- Date: Feb-03-2017
- Authors: Kijung Shin (kijungs@cs.cmu.edu)

2 Introduction

DenseStream is an incremental algorithm for detecting dense subtensors in tensor streams. **DenseAlert** is an incremental algorithm for spotting suddenly emerging dense subtensors. They have the following properties:

- *Fast and 'Any Time'*: By maintaining and updating a dense subtensor, our algorithms detect a dense subtensor in a tensor stream significantly faster than batch algorithms.
- *Provably Accurate*: Our algorithms provide theoretical guarantees on their accuracy, and show high accuracy in practice.
- *Effective*: Our algorithms successfully identify anomalies, such as bot activities, rating manipulations, and network intrusions, in real-world tensors.

Detailed information about our algorithms are explained in the following paper

- Kijung Shin, Bryan Hooi, Jisu Kim, and Christos Faloutsos. "*DenseAlert: Incremental Dense-Subtensor Detection in Tensor Streams*", ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD) 2017, Halifax, Canada

3 Installation

- This package requires that java 1.7 or greater be installed in the system and set in PATH.

- For compilation (optional), type *./compile.sh*
- For packaging (optional), type *./package.sh*
- For demo (optional), type *make*

4 APIs for DenseStream

4.1 Package: *densealert*

4.2 Class: *DenseStream*

4.3 Algorithms:

- public *DenseStream* (int *order*)
    - create a *DenseStream* object for a tensor with the given order
    - *order*: the order of the considered tensor
- public void *insert* (int[] *entry*)
    - process an insertion of or an increment in the tensor entry
    - *entry*: an integer array in the form of $(i_1, i_2, \dots, i_N, \delta+)$ where $(i_1, i_2, \dots, i_N)$ is the index of the considered entry, and $\delta+$ is the increment in the value of the entry with index $(i_1, i_2, \dots, i_N)$. $\delta+$ should be greater than 0.
- public void *delete* (int[] *entry*)
    - process a deletion of or a decrement in the tensor entry
    - *entry*: an integer array in the form of $(i_1, i_2, \dots, i_N, \delta-)$ where $(i_1, i_2, \dots, i_N)$ is the index of the considered entry, and $\delta-$ is the decrement in the value of the entry with index $(i_1, i_2, \dots, i_N)$. $\delta-$ should be greater than 0.
- public double *getDensity* ()
    - return the density of the current maintained subtensor
- public Map<Integer, int[]> *getSliceIndices*()
    - return the slice indices in the input tensor that compose the maintained subtensor
    - *return*: a map whose keys are modes and values are the lists of indices in the corresponding mode composing the maintained subtensor

4.4 Example Code: see *DenseStreamExample.java* for an example code using *DenseStream*

5 APIs for DenseStream

5.1 Package: *densealert*

5.2 Class: *DenseAlert*

5.3 Algorithms:

- public *DenseAlert* (int *order*, long *window*)
  - create a *DenseAlert* object for a tensor with the given order
  - *order*: the order of the considered tensor
  - *window*: the length in seconds of the time window
- public void *insert* (int[] *entry,* long *timestamp*)
  - process an insertion of or an increment in the tensor entry
  - *entry*: an integer array in the form of $(\mathrm{i}_1, \mathrm{i}_2, \dots, \mathrm{i}_N, \delta+)$ where $(\mathrm{i}_1, \mathrm{i}_2, \dots, \mathrm{i}_N)$ is the index of the considered entry, and $\delta+$ is the increment in the value of the entry with index $(\mathrm{i}_1, \mathrm{i}_2, \dots, \mathrm{i}_N)$. $\delta+$ should be greater than 0.
  - *timestamp*: unixtime when this change happened in the input tensor
- public double *getDensity* ()
  - return the density of the current maintained subtensor
- public Map<Integer, int[]> *getSliceIndices* ()
  - return the slice indices in the input tensor that compose the maintained subtensor
  - *return*: a map whose keys are modes and values are the lists of indices in the corresponding mode composing the maintained subtensor

5.4 Example Code: see *DenseAlertExample.java* for an example code using *DenseAlert*