Detection Methods for Multiple Dense Blocks in Tensor Data (Postprint)

Authors: Fan Weijun, Cheng Yanyun

Date: 2018-05-20T00:00:00+00:00

Abstract

Numerous prior studies have demonstrated that dense regions in real-world tensor data exhibit anomalous or fraudulent behaviors, such as Weibo zombie follower activities and network attacks. Consequently, researchers have proposed various methods for dense block extraction; however, these approaches suffer from low accuracy and recall. To address these deficiencies, we propose a multi-dense block detection method based on binary tree search, abbreviated as DDB-BST, which performs evaluation metric-driven local searches on tensor data to identify sub-tensors with optimal evaluation metric values, partitions the data into left and right child nodes, and determines binary tree growth termination by continuously comparing the numerical relationships between parent and child node evaluation metric values. We provide a rigorous mathematical proof for the termination condition. Experimental results on both synthetic and real-world datasets demonstrate that DDB-BST improves the F1 score by nearly 30% compared to the existing M-zoom multi-dense block method.

Full Text

Abstract

Past studies have shown that dense blocks in real-world tensor data often indicate anomalous or fraudulent behavior, such as zombie follower activities on social media or network attacks. Consequently, researchers have proposed various methods for dense block extraction. However, these methods suffer from low accuracy and recall rates. To address these limitations, we propose a multi-dense-block detection method based on binary tree search, abbreviated as DDB-BST. This approach performs localized searches on tensor data based on evaluation metrics to identify sub-tensors with maximal metric values, then partitions the data into left and right child nodes. By continuously comparing the metric values between parent and child nodes, the algorithm determines whether the binary tree should continue growing. We provide rigorous mathematical proofs

for the termination conditions. Experiments on both synthetic and real-world datasets demonstrate that DDB-BST improves the F1-score by nearly 30% compared to the existing M-zoom multi-dense-block detection method.

Key Words: tensor data; dense blocks; binary tree search; termination condition

0 Introduction

Consider the task of detecting anomalous network connections: how can we extract suspicious behaviors such as a group of IP addresses sending connection requests every few seconds to specific ports on a target IP, or obtain characteristic information about network connections over time such as protocol type, byte counts from source to destination, and number of connections to the same host within the past two seconds? Many studies have addressed such problems using tensor models, where dense blocks in tensors represent synchronized behaviors among groups of users—behaviors that are often suspicious. Consequently, dense block extraction from tensors has been widely applied in network intrusion detection, detecting boosted retweet activities on microblogs, identifying zombie fan activities, and genetics research.

Currently, three main categories of methods exist for fast and accurate dense block detection in tensors. The first category comprises tensor decomposition-based approaches, such as HOSVD and CP decomposition, which have been applied to dense sub-tensor mining. Recent improvements include distribution-based models, sampling-based methods, and tensor decomposition for hundreds of millions of scale data. However, tensor decomposition-based dense block mining suffers from several drawbacks: it fails to consider background data properties, lacks scalability under density metrics, and cannot provide reasonable boundaries.

The second category involves dense subgraph mining methods. Recent approaches in this area include: finding dense subgraphs using maximum overall density with limited overlap; discovering dense subgraphs based on core decomposition; identifying dense subgraphs on uncertain graphs using novel evaluation metrics; and dynamic dense subgraph mining based on data streams or distributed computing. The latest research defines new criteria for dense block detection in tensors. The CrossSpot algorithm, for instance, randomly selects a block and uses a greedy-like method to continuously adjust its dimensions until reaching a local optimum. However, CrossSpot fails to provide appropriate boundaries, resulting in poor detection performance on tensor data containing multiple dense blocks. The M-zoom algorithm similarly employs a greedy-like approach, starting from the entire tensor and iteratively removing values along each dimension until reaching a local optimum (i.e., a dense block). After removing the dense block, it repeats the process on the remaining data. While this achieves multi-dense-block detection, its accuracy is significantly compromised.

To overcome these limitations of existing metric-based algorithms, this paper

proposes DDB-BST (Detect Dense Block with Binary Search Tree), a binary tree multi-dense-block search algorithm based on the Suspiciousness evaluation metric. This method effectively detects multi-dense-blocks of various forms in tensor data. Through rigorous mathematical proofs of the binary tree growth conditions, we enable determination of whether dense blocks exist in root data. Experimental results on diverse datasets demonstrate that our algorithm can effectively detect dense blocks in tensor data.

1 Preliminaries

1.1 Notation Definitions

Table 1 lists the symbols used throughout this paper and their meanings.

Table 1: Symbol Definitions

Symbol	Description
D(A ,⋯,A_K,X)	Dataset D containing K nominal attributes and one non-negative numerical attribute
K	Number of nominal attributes in D, i.e., dimensionality of tensor data
A_j	The j-th nominal attribute in D
X	Numerical attribute data in D
B(B ,⋯,B_K,X)	Sub-tensor data record in D
(B,D)	Evaluation metric value of sub-tensor B in D
S_D (or S_B)	Sum of X in tensor D (or sub-tensor B)
V_D (or V_B)	Volume of tensor D (or sub-tensor B)
a_j^i	The i-th value in A_j
Δs_{a_j^i}	Sum of all X values under the i-th value in A_j

Example 1: Network Access History. Consider a dataset D(user, ip, date, count) where each record d(u, IP, d, c) represents user u accessing IP address c times at time d. The nominal attributes are A = user, A = IP, A = date, and the numerical attribute is X = count. Let B = {Lucy, Nike}, B = {192.168.5.2, 192.168.2.1}, and B = {2017-1-2}. Then B represents the shaded sub-tensor in Figure 1(b), with S_B = 16, V_B = 4, a [1] = {5}, and Δs_{a [1]} = {7,3,1}.

1.2 Evaluation Metric

This paper adopts the Suspiciousness evaluation metric proposed in literature [16].

Definition 1 (Suspiciousness): In tensor data D, the Suspiciousness value of sub-tensor B is calculated as:

$$\rho(B,D) = \frac{S_B}{V_B}\ln\left(\frac{S_B}{V_B}\right) - \frac{S_B - S_B/V_B}{V_D - V_B}\ln\left(\frac{S_D - S_B}{V_D - V_B}\right)$$

The Suspiciousness metric satisfies two key axioms:

Axiom 1 (Density Axiom): If two sub-tensors have the same size, the one with a larger sum of numerical attribute values has greater Suspiciousness.

Axiom 2 (Concentration Axiom): If two sub-tensors have the same sum of numerical attribute values, the smaller one has greater Suspiciousness.

1.3 Problem Definition

Problem 1: Given a K-dimensional tensor dataset D containing m dense blocks, detect these m dense blocks using a binary tree search and local search algorithm based on the evaluation metric in Equation (1).

2 Multi-Dense-Block Detection via Metric-Based Binary Tree Search

2.1 Algorithm Flow

The DDB-BST algorithm consists of two main components. First, it preprocesses the original data to obtain a format suitable for algorithmic processing. Second, it performs Algorithm 2 on the processed data to obtain the left and right child nodes of the binary tree. This process repeats on the left and right nodes until termination conditions are met, thereby identifying multiple anomalous blocks in the original data.

Algorithm 1: Binary Tree Search Algorithm

Input: Original tensor data D_ori

Output: k dense blocks

1. Initialize nodeR = 0, R = D_ori, nodeR is the root node ID
2. Insert data R into the binary tree, assign node ID as nodeR with key value key = (R, D_ori)
3. Perform Algorithm 2 on data R to obtain left child B_l and right child B_r, with node IDs nodeR+ “1” and nodeR+ “0” , and key values key = (B_l, D_ori) and key = (B_r, D_ori) respectively
4. If key_root key_left + key_right, then save R to list dense_block; otherwise continue to step 2 with R = B_l and R = B_r
5. Repeat until all searches complete, return dense_block

Algorithm 2: Local Search for Single Dense Block

Input: Tensor data R, seed

Output: Left and right sub-tensor data frames B_L and B_R

1. Initialize B_L = R
2. For each dimension j = 1···K, sort elements in A_j in descending order of Δs_{a_j^i}, keep elements in other dimensions unchanged, then iteratively add elements to B_L while continuously calculating (B_L, R) until it reaches maximum
3. Set B_R = R B_L
4. Repeat steps 1-2 until Suspiciousness converges

2.2 Algorithm Complexity Analysis

The time complexity of Algorithm 2 (local search for single dense block) is O(T × K × (E/N + N log N)), where T is the number of iterations, K is the dimensionality of tensor data, E is the number of non-zero elements, and N is the maximum length in any dimension. Typically, T and K are treated as constants, making Algorithm 2' s time complexity linear with respect to the number of non-zero elements in the tensor.

The time complexity of Algorithm 1 (binary tree search) is O(c × O(Algorithm 2)), where c is the number of times Algorithm 2 is executed. Therefore, Algorithm 1' s time complexity is linear with respect to Algorithm 2' s complexity, making the overall algorithm complexity O(T × K × (E/N + N log N) × c).

2.3 Binary Tree Growth Conditions

Definition 2: Assume the initial data is a sparse matrix (i.e., < 1). Performing Algorithm 2 on the original data yields the sub-tensor with maximum Suspiciousness value, defined as the left subtree (ID = 1). Removing the left subtree from the original data yields the right subtree (ID = 0).

Let the Suspiciousness values calculated based on the original background data be:

$$\text{key}_{\text{root}} = \rho(R, D_{\text{ori}})$$

$$\text{key}_{\text{left}} = \rho(B_l, D_{\text{ori}})$$

$$\text{key}_{\text{right}} = \rho(B_r, D_{\text{ori}})$$

Expanding Equation (1) yields Equation (2):

$$\text{key}_{\text{root}} = \frac{S_R}{V_R} \ln\left(\frac{S_R}{V_R}\right) - \frac{S_D - S_R}{V_D - V_R} \ln\left(\frac{S_D - S_R}{V_D - V_R}\right)$$

Theorem 1: If the root data is completely anomalous or completely non-anomalous, then:

$$\text{key}_{\text{root}} \geq \text{key}_{\text{left}} + \text{key}_{\text{right}}$$

Proof: For the case where root data is completely anomalous, using a cutting approach, we have S_R S_l + S_r and V_R V_l + V_r. Simplifying Equation (2) yields Equation (3):

$$\text{key}_{\text{root}} \approx \frac{S_l + S_r}{V_l + V_r} \ln\left(\frac{S_l + S_r}{V_l + V_r}\right) - \frac{S_D - (S_l + S_r)}{V_D - (V_l + V_r)} \ln\left(\frac{S_D - (S_l + S_r)}{V_D - (V_l + V_r)}\right)$$

Further simplification leads to Equation (4), establishing that key_root key_left + key_right. For completely non-anomalous root data, similar reasoning applies. Combining both cases proves the theorem.

Theorem 2: If the root data contains anomalous components, then:

$$\text{key}_{\text{root}} < \text{key}_{\text{left}} + \text{key}_{\text{right}}$$

Proof: For non-completely anomalous blocks containing anomalous components, non-anomalous portions must exist. Since the original data is sparse, we have _l > _root and _r > _root. This leads to:

$$\frac{S_l}{V_l} > \frac{S_R}{V_R} \quad \text{and} \quad \frac{S_r}{V_r} > \frac{S_R}{V_R}$$

Consequently, key_root < key_left + key_right, proving that the binary tree should continue growing when anomalous components are present.

3 Experiments

To validate the proposed algorithm' s effectiveness, we conducted extensive experiments comparing both computational efficiency and detection accuracy (precision and recall) against relevant anomaly detection methods, particularly focusing on comparisons between DDB-BST and the M-zoom algorithm [12] for multi-anomaly detection.

All algorithms were implemented in R and executed on a PC with Intel(R) Core(TM) i5-2450M CPU (2.5 GHz), 4 GB RAM, running Windows 7 32-bit. Table 2 summarizes the datasets used. The LBNL dataset comprises public network packets from Lawrence Berkeley National Lab, while the AirForce dataset originates from the 1999 KDD Cup network packet data. S-Data-2 and S-Data-3 are synthetic datasets containing multiple anomaly clusters.

Table 2: Dataset Summary

Dataset	Dimensions	Size	Non-zero Elements
S-Data-2D	2	500×500	83.5K
S-Data-3D	3	500×500×500	512.3K
LBNL	4	-	3.73M
AirForce (10%)	7	-	64.6M

3.1 Synthetic Dataset Experiments

We compared DDB-BST and M-zoom on synthetic datasets S-Data-2D and S-Data-3D.

S-Data-2D: Generated using an ERP model with parameters: (1) K = 2 dimensions, (2) size 500×500, (3) Poisson distribution = 0.01. Injected three 2D anomaly clusters of sizes 10×10, 20×20, and 30×30. The task was to detect these three clusters from the random dataset.

S-Data-3D: Generated with parameters: (1) K = 3 dimensions, (2) size 500×500×500, (3) Poisson distribution = 0.01. Injected three 3D anomaly clusters of sizes 15×10×15, 10×10×10, and 20×10×30.

Table 3 presents the detection results. DDB-BST achieves significantly higher F1-scores than both M-zoom and CrossSpot, with improvements of approximately 20% over M-zoom and 40% over CrossSpot. This superiority stems from the fact that while M-zoom and CrossSpot only find sub-tensors with maximal evaluation metrics, the highest metric does not guarantee purely anomalous data. DDB-BST further discriminates to ensure detected blocks contain no normal data.

Table 3: Performance Comparison on Synthetic Datasets

Algorithm	S-Data-2D			S-Data-3D		
	Precision	Recall	F1	Precision	Recall	F1
DDB-BST	-	-	-	-	-	-
M-zoom	-	-	-	-	-	-
CrossSpot	-	-	-	-	-	-

3.2 Real-World Dataset Experiments

We validated DDB-BST' s effectiveness on two real-world datasets. The LBNL dataset consists of four nominal attributes (time, source IP, destination IP, port number) and one numerical attribute (packet count). The AirForce dataset contains seven attributes (protocol, service, source bytes, destination bytes, flag, count, service count) without IP information, with connection count as the numerical attribute.

Table 4 shows DDB-BST' s network attack detection results on real datasets. Anomaly clusters typically comprise various network attacks, and dense portions formed by packet counts or connection counts in high-dimensional nominal attribute tensors correspond precisely to anomalous regions.

Table 4: Network Intrusion Detection Results Using DDB-BST

Dataset	Detected Block Dimensions	Precision
LBNL	3610×472×49×35	99.4%
LBNL	3610×81×1100×331	97.1%
AirForce	3×66×11×3×1×224×36	99.7%
AirForce	3×66×11×12×7×184×106	97.3%
AirForce	3×66×11×297×278×24×25	95.8%

Table 5 compares DDB-BST' s performance with M-zoom and CrossSpot on real datasets. DDB-BST achieves F1-scores approximately 20% higher than M-zoom and 40% higher than CrossSpot, confirming its superior ability to identify pure anomaly clusters.

Table 5: Performance Comparison on Real Datasets

Algorithm	LBNL			AirForce		
	Precision	Recall	F1	Precision	Recall	F1
DDB-BST	-	-	-	-	-	-
M-zoom	-	-	-	-	-	-
CrossSpot	-	-	-	-	-	-

4 Conclusion

This paper investigated multi-dense-block detection methods in high-dimensional tensor data, proposing a binary tree-based multi-dense-block detection algorithm. We established binary tree growth conditions and mathematically proved their validity. Experimental results demonstrate that the proposed algorithm achieves high precision and recall rates, significantly outperforming existing methods.

References

[1] Maruhashi K, Guo F, Faloutsos C. Multi aspect forensics: pattern mining on large-scale heterogeneous networks with tensor analysis [C]// Proc of International Conference on Advances in Social Networks Analysis and Mining. 2011: 203-210.

[2] Shah N, Beutel A, Gallagher B, et al. Spotting suspicious link behavior with fBox: an adversarial perspective [C]// Proc of Robot World Cup XVIII. Springer International Publishing, 2014: 295-305.

[3] Rodrigues T, Cunha T, IencomD, et al. Retweet patterns: detection of spatio-temporal patterns of retweets [M]// New Advances in Information Systems and Technologies. [S. l.]: Springer International Publishing, 2016.

[4] 王越, 张剑金, 刘芳芳. 一种多特征微博僵尸粉检测方法与实 [J]. 中国科技论文, 2014 (1): 81-86.

[5] Shin K, Kang U. Distributed methods for high-dimensional and large-scale tensor factorization [C]// Proc of IEEE International Conference on Data Mining. 2015: 989-994.

[6] Papalexakis E E, Faloutsos C, Sidiropoulos N D. ParCube: sparse parallelizable tensor decompositions [J]. ACM Transactions on Knowledge Discovery from Data, 2012, 10 (1): 521-536.

[7] Jeon I, Papalexakis E E, Kang U, et al. HaTen2: billion-scale tensor decompositions [C]// Proc of International Conference on Data Engineering. 2015: 1047-1058.

[8] Lee V E, Ning R, Jin R, et al. A survey of algorithms for dense subgraph discovery [M]// Managing and Mining Graph Data. 2010: 303-336.

[9] Balalau O D, Bonchi F, Chan T H H, et al. Finding subgraphs with maximum total density and limited overlap [C]// Proc of the 8th ACM International Conference on Web Search and Data Mining. New York: ACM Press, 2015: 379-388.

[10] Sariyuce A E, Seshadhri C, Pinar A, et al. Finding the hierarchy of dense subgraphs using nucleus decompositions [C]// Proc of International Conference on World Wide Web. 2015: 927-937.

[11] 朱鎔, 邹兆年, 李建中. 不确定图上的 Top-k 稠密子图挖掘算 [J]. 计算机学报, 2016, 39 (8): 1570-1582.

[12] Bahmani B, Kumar R, Vassilvitskii S. Densest subgraph in streaming and MapReduce [J]. Computer Science, 2012: 454-465.

[13] Bahmani B, Goel A, Munagala K. Efficient primal-dual graph algorithms for MapReduce [M]// Algorithms and Models for the Web Graph. Springer International Publishing, 2014: 59-78.

[14] Jiang M, Beutel A, Cui P, et al. A General Suspiciousness Metric for Dense Blocks in Multimodal Data [C]// IEEE International Conference on Data Mining. 2015: 781-786.

[15] Shin K, Hooi B, Faloutsos C. M-zoom: fast dense-block detection in tensors with quality guarantees [M]// Machine Learning and Knowledge Discovery in Databases. 2016.

[16] Jiang M, Beutel A, Cui P, et al. Spotting suspicious behaviors in multimodal data: a general metric and algorithms [J]. IEEE Transactions on Knowledge & Data Engineering, 2016: 1.

Note: Figure translations are in progress. See original paper for figures.