DenseAlert: Incremental Dense-Subtensor Detection in Tensor Streams

Kijung Shin [1], Bryan Hooi [2], Jisu Kim [2], and Christos Faloutsos [1]

Carnegie Mellon University

[1]{kijungs, christos}@cs.cmu.edu, [2]{bhooi,jisuk1}@andrew.cmu.edu

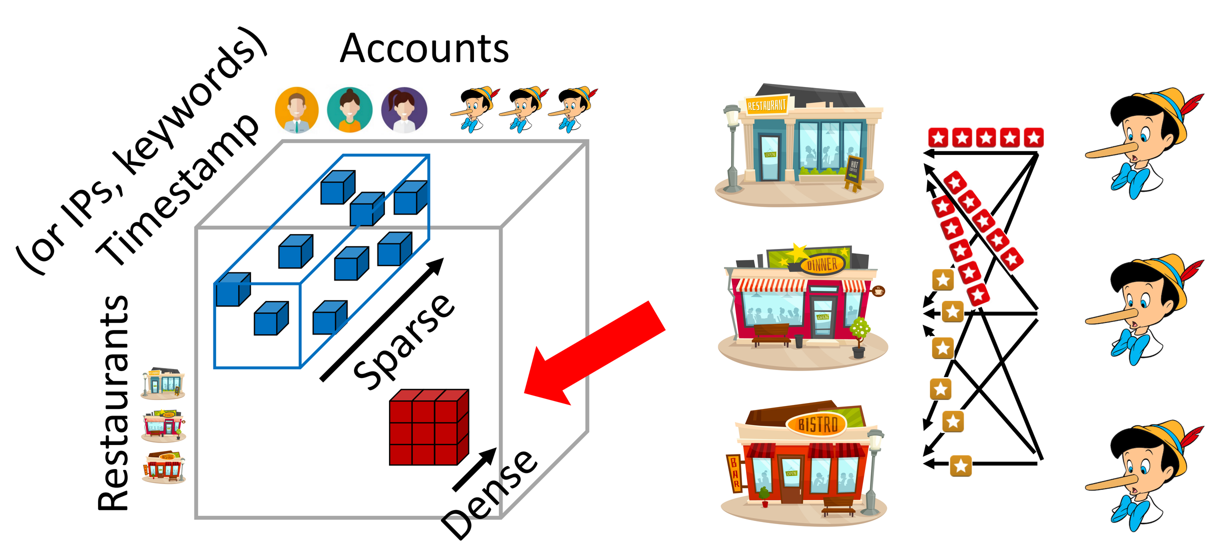
Code and Data: https://github.com/kijungs/densealert

Summary

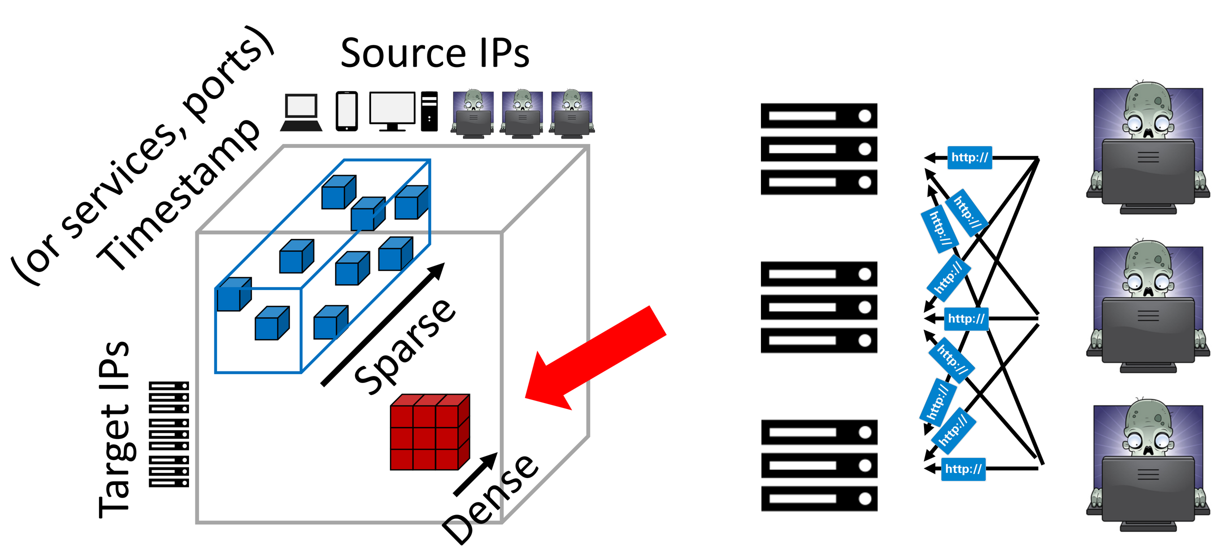
- **Goal: to detect and update dense subtensors in a dynamic tensor**
- **Previous Work:**
  - showed that dense subtensors in real-world tensors signal *anomalies* or *fraud*
  - proposed *batch algorithms* for fast and accurate dense-subtensor detection
- **Proposed Incremental Algorithms:**
  - **DenseStream:** incremental algorithm for detecting the *densest subtensor*
  - **DenseAlert:** incremental algorithm detecting *suddenly appearing dense subtensors*
- **Result:**
  - **Fast:** updates by our algorithms are up to *1,000,000X* faster than batch algorithms
  - **Provably Accurate**: our algorithms provide *theoretical accuracy guarantees*
  - **Effective**: our algorithms successfully detect anomalies in real-world tensors, including *network attacks, bot activities,* and rating manipulations

Motivations

- **Rating Data:** Rating manipulation results in dense subtensors



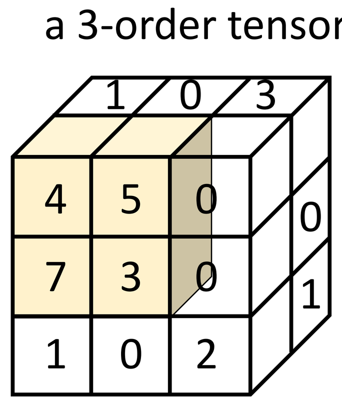
- **TCP Dumps:** Network attacks result in dense subtensors



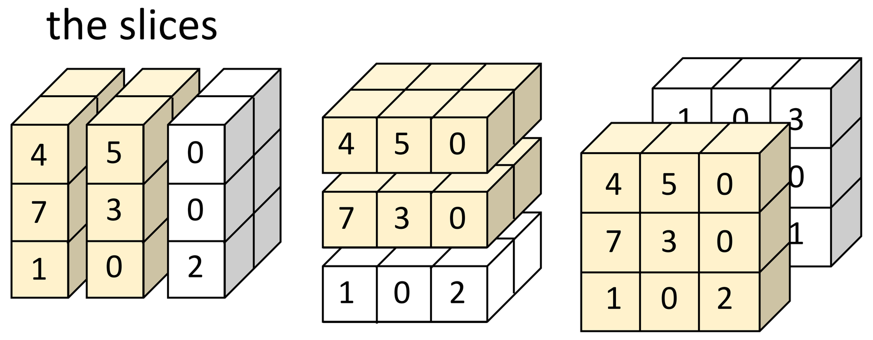
How can we detect dense subtensors incrementally in a dynamic tensor?

Basic Concepts

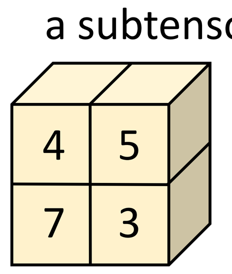
a 3-order tensor



the slices



a subtensor

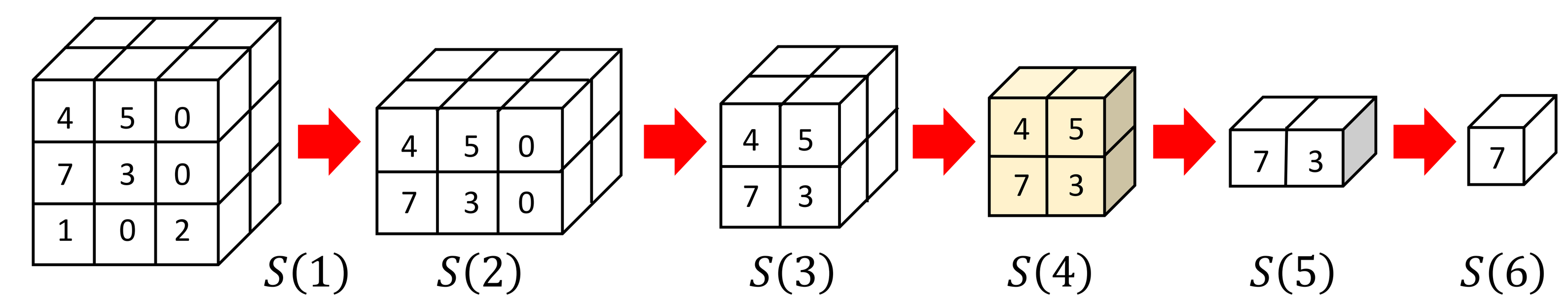


(density = $\frac{4+5+7+3}{5}$)

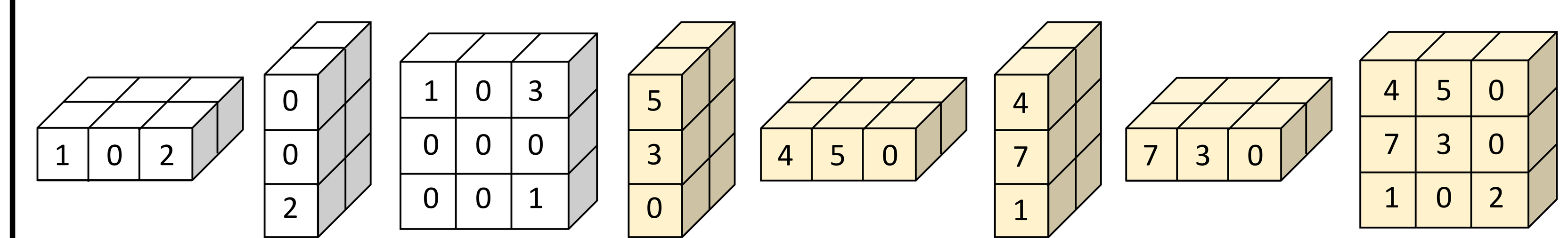
Density of a subtensor := $\frac{\text{Mass (i.e., the sum of entries) of the subtensor}}{\text{Number of slices composing the subtensor}}$

D-ordering

- **Greedy Shaving:** repeatedly removing a slice with the minimum mass in the remaining tensor



- **D-ordering**: the order by which Greedy Shaving removes the slices



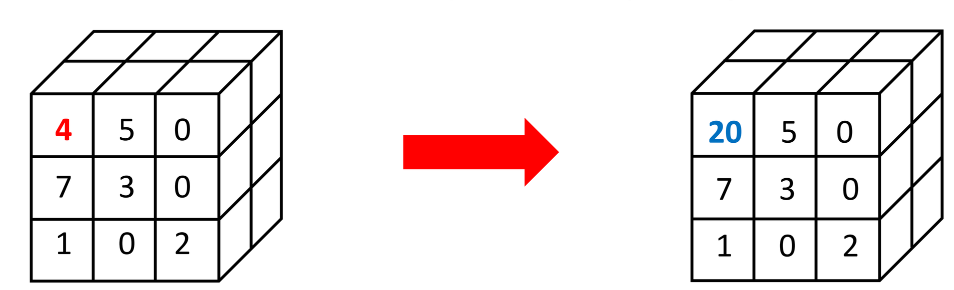
- **Accuracy guarantee**: Let i^* be the index i maximizing the density of $S(i)$. Then,

$$\text{Density of } S(i^*) \geq \frac{\text{Density of the densest subtensor}}{\text{Order of the input tensor}}$$

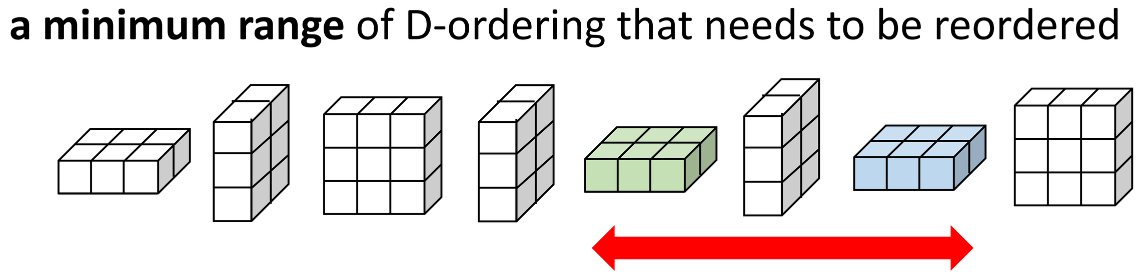
- **Speed guarantee of D-ordering**: Given a D-ordering and the mass when each slice is removed, finding such i^* and the density of $S(i^*)$ takes $O(\#\ slices)$

Proposed Algorithm 1: DenseStream

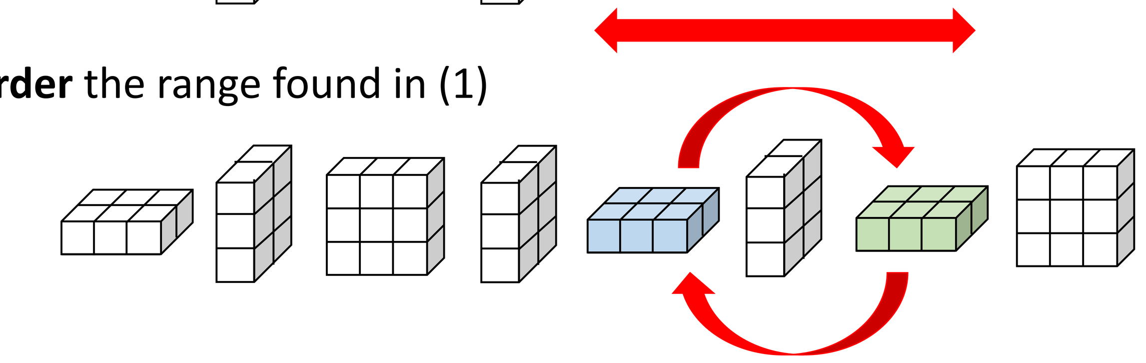
- **Goal:** to maintain a dense subtensor (by maintaining a D-ordering) while the input tensor changes
- **Procedure of DenseStream:**
  Given a change (increment or decrement) in the input dynamic tensor,



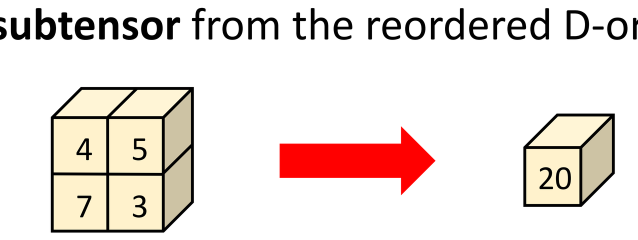
(1) **Find a minimum range** of D-ordering that needs to be reordered



(2) **Reorder** the range found in (1)



(3) **Update a dense subtensor** from the reordered D-ordering **if needed**



- **Accuracy guarantee of DenseStream:**

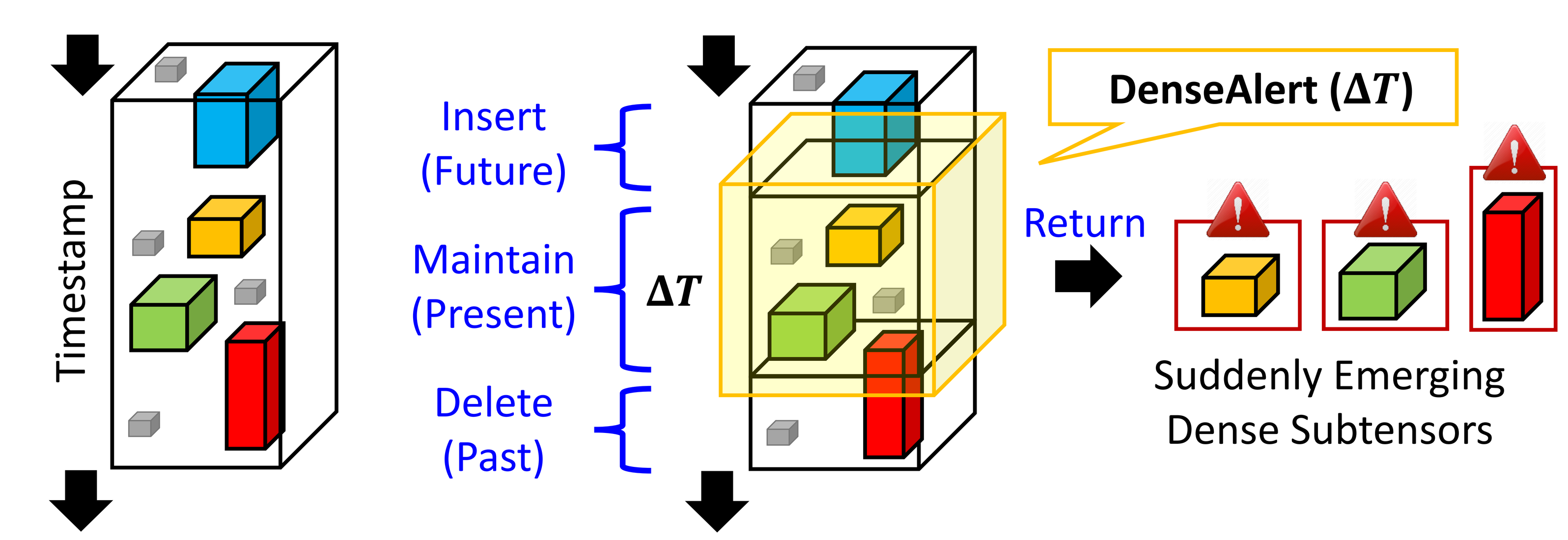
$$\text{Density of the subtensor maintained by \textbf{DenseStream}} \geq \frac{\text{Density of the densest subtensor}}{\text{Order of the input tensor}}$$

- **Speed of DenseStream:** Reordering by **DenseStream** is several orders of magnitude faster than computing a D-ordering from scratch

Proposed Algorithm 2: DenseAlert

- **Goal:** to detect **suddenly appearing dense subtensors** by maintaining a dense subtensor created within the latest ΔT time units in a timed tensor

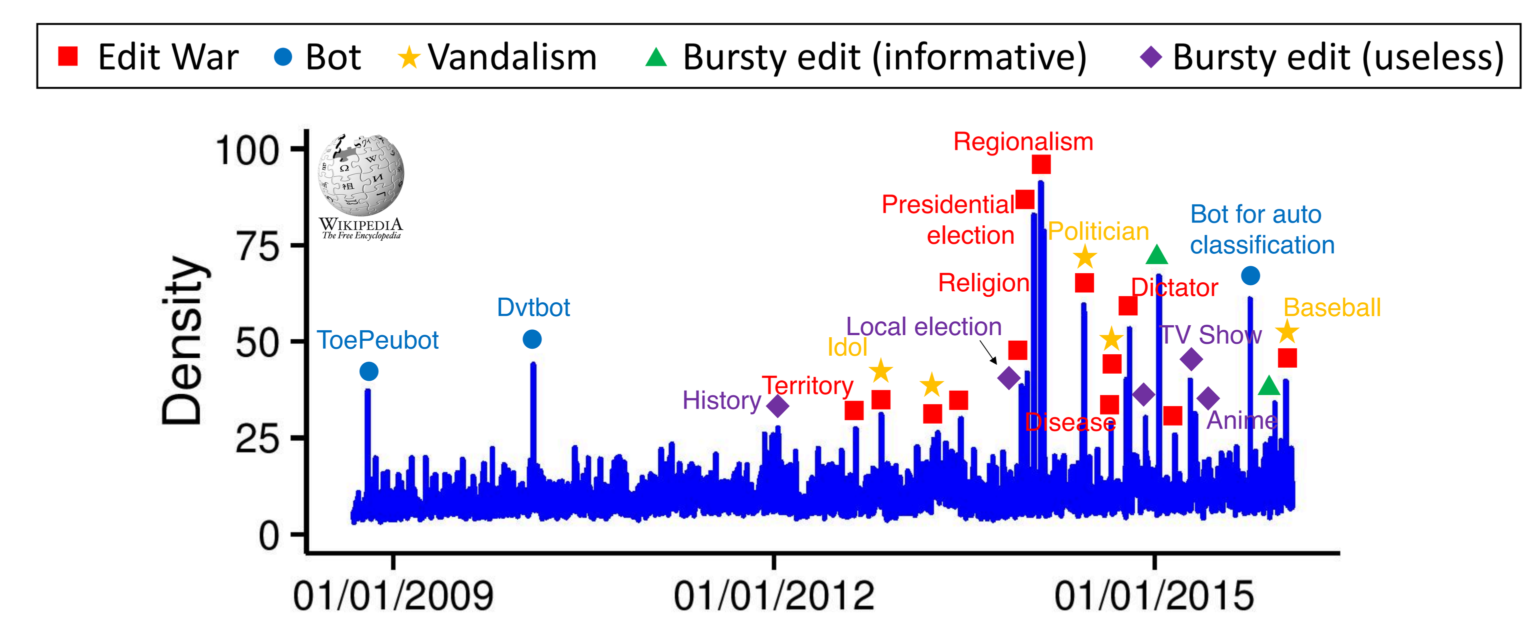
(a) timed tensor (b) DenseAlert



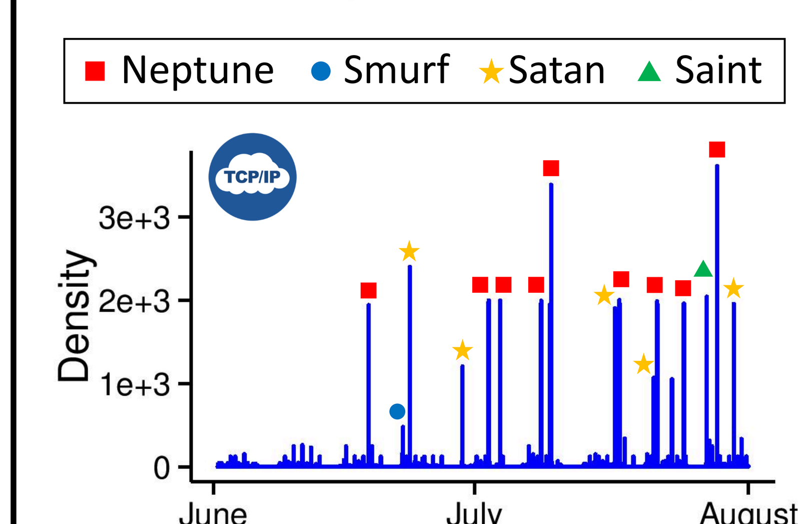
Experimental Results

- **Q1 Effectiveness:** what does DenseAlert detect in real-world tensors?

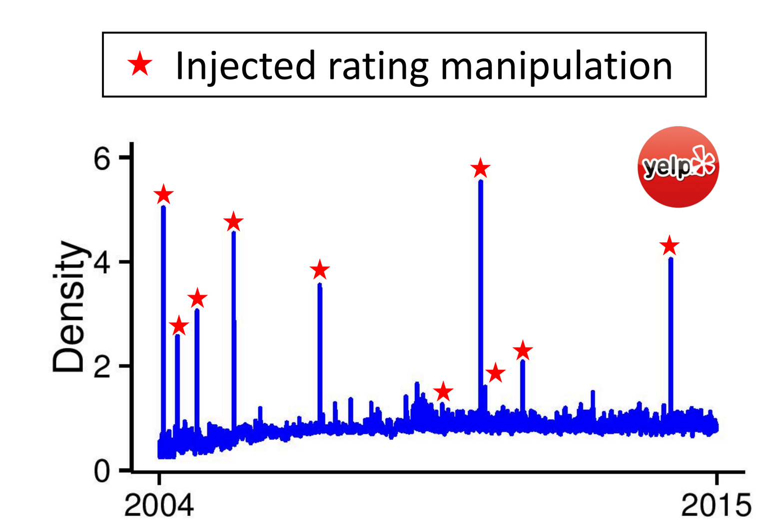
(1) Wikipedia revision history (users X pages X timestamps)



(2) TCP Dump (source IPs X target IPs X timestamps)



(3) Rating Data (users X businesses X ratings X timestamps)



- **Q2 Speed:** How fast are updates by **DenseStream** compared to batch algorithms?
- **Q3 Accuracy:** How accurately does **DenseStream** maintain a dense subtensor?