Higher-order Count Sketch: Dimension Reduction That Retains Efficient Tensor Operations

Data Compression Conference

Rakuten Institute of Technology Caltech UCI

Yang Shi - Rakuten Institute of Technology, Rakuten, Ink. ; UCIrvine **Animashree Anandkumar** - California Institute of Technology

Keypoints

- Propose HCS: it projects tensor to another tensor of different order with different dimensions, which are chosen by the user.
- Exponential saving (with respect to the order of the tensor)in the memory requirements of the hash functions.
- Efficient approximation of tensor product and tensor contraction.

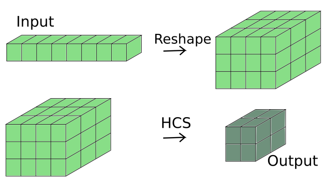


Figure 1: Higher-order count sketch

Intuition

- Memory constraints: applying CS on large size data requires same size hashing parameters.
- Applying traditional data dimension reduction methods to tensors is typically computationally expensive.

Related Work

- Structured data: SVD/PCA
- Sparsity/non-negativity constraints: CX and CUR matrix decomposition
- Data frequency: Count sketch[1]

Count Sketch

Count Sketch(CS) Given two 2-wise independent random hash functions h:$[n] \rightarrow [c]$ and s:$[n] \rightarrow \{\pm 1\}$. Count Sketch of a point x $\in \mathbb{R}^n$ is denoted by $CS(x) \in \mathbb{R}^c$ where $CS(x)_j = \Sigma_{h(i)=j} s(i)x_i$.
[2] use CS and propose a fast algorithm to compute count sketch of an outer product of two vectors using FFT properties. $CS(uv^T) = IFFT(FFT(CS(u)) \circ FFT(CS(v)))$. Computation complexity: $O(n^2) \rightarrow O(n + clogc)$.

Higher-order Count Sketch

HCS Given a vector $u \in \mathbb{R}^d$, random hash functions h_k:$[n_k] \rightarrow [m_k]$, $k \in [l]$, random sign functions s_k:$[n_k] \rightarrow \{\pm 1\}$, $k \in [l]$, and $d = \Pi_{k=1}^{l} n_k$, we propose HCS as:

$$\text{HCS}(u)_{t_1,\cdots,t_l} := \sum_{h_1(i_1)=t_1,\dots,h_l(i_l)=t_l} s_1(i_1)\cdots s_l(i_l) reshape(u)_{i_1\dots i_l} \quad (1)$$

Using tensor operations, we can denote HCS as:

$$\text{HCS}(u) = (\mathcal{S} \circ reshape(u))_{\times 1} H_1 \dots {}_{\times l} H_l \quad (2)$$

Here, $\mathcal{S} = s_1 \otimes \cdots \otimes s_l \in \mathbb{R}^{n_1 \times \cdots \times n_l}$, $H_k \in \mathbb{R}^{n_k \times m_k}$, $H_k(a,b) = 1$, if $h_k(a) = b$, otherwise $H_i(a,b) = 0$, for $\forall a \in [n_k], b \in [m_k], k \in [l]$. To recover the original tensor, we have

$$\hat{u}_j = s_1(i_1)\cdots s_l(i_l) \text{HCS}(u)_{h_1(i_1),\cdots,h_l(i_l)} \quad (3)$$

Theorem(HCS recovery analysis) Assume $\mathcal{T}_p$ is a pth-order tensor by fixing $l-p$ modes of a lth-order tensor $reshape(u)$: Given a vector $u \in \mathbb{R}^d$, assume T_p is the maximum frobenium norm of all $\mathcal{T}_p$, Equation 3 computes an unbiased estimator for u_{j*} with variance bounded by:

$$\mathbf{Var}(\hat{u}_{j*}) = O(\sum_{p=1}^{l} \frac{T_p^2}{m^p}) \quad (4)$$

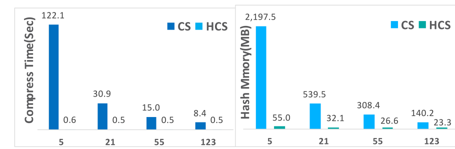
Efficient Tensor Operations

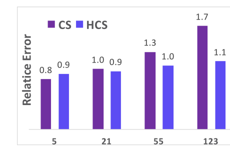
Table 1: General tensor operation estimation (Assume A is a set of indices with length p, B is a set of indices with length q, each index value $O(n)$, assume the size of R is l with each index value $O(r)$, $g = \max(p,q)$)

Tensor Product: $\mathcal{A} \in \mathbb{R}^A$, $\mathcal{B} \in \mathbb{R}^B$		
Operator	Computation	Memory
$\text{CS}(\mathcal{A} \otimes \mathcal{B}) = \text{CS}(vec(\mathcal{A}) \otimes vec(\mathcal{B}))$	$O(n^g + c\log c)$	$O(c + n^g)$
$\text{HCS}(\mathcal{A} \otimes \mathcal{B}) = \text{HCS}(\mathcal{A}) * \text{HCS}(\mathcal{B})$	$O(n^g + c\log c)$	$O(c + gn)$
Tensor Contraction: $\mathcal{A} \in \mathbb{R}^A$, $\mathcal{B} \in \mathbb{R}^B$ with contraction indices R		
Operator	Computation	Memory
$\text{CS}(\mathcal{A}\mathcal{B}) = \Sigma_R \text{CS}(A_{:R} \otimes B_{R:})$	$O(r^l n^g + cr^l \log c)$	$O(c + cr^l + n^g)$
$\text{HCS}(\mathcal{A}\mathcal{B}) = \text{HCS}(\mathcal{A})\text{HCS}(\mathcal{B})$	$O(r^l n^g + cr^l)$	$O(c + c^{\frac{g}{p+q}} r^l + gn)$

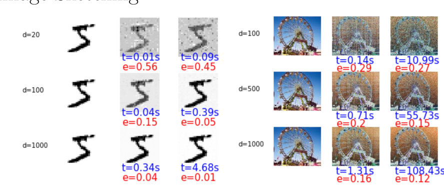
Experiments

- Tensor Contraction Estimation

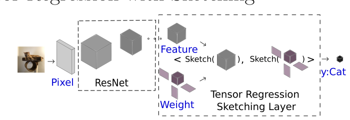


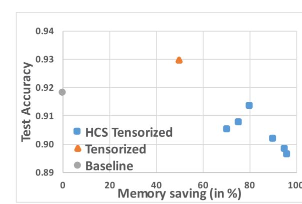


- Image Sketching



- Tensor Regression with Sketching





Reference

[1] M. Charikar, K. Chen, and M. Farach-Colton. Finding frequent items in data streams.

[2] Rasmus Pagh. Compressed matrix multiplication.