MULTIPLICATION-AVOIDING VARIANT OF POWER ITERATION WITH APPLICATIONS

Hongyi Pan   Diaa Badawi   Runxuan Miao   Erdem Koyuncu   Ahmet Enis Cetin

Department of Electrical and Computer Engineering, University of Illinois at Chicago, Chicago, IL

ABSTRACT

Power iteration is a fundamental algorithm in data analysis. It extracts the eigenvector corresponding to the largest eigenvalue of a given matrix. Applications include ranking algorithms, recommendation systems, principal component analysis (PCA), among many others. In this paper, We introduce multiplication-avoiding power iteration (MAPI), which replaces the standard ℓ_2-inner products that appear at the regular power iteration (RPI) with multiplication-free vector products which are Mercer-type kernel operations related with the ℓ_1 norm. Precisely, for an $n \times n$ matrix, MAPI requires n multiplications, while RPI needs n^2 multiplications per iteration. Therefore, MAPI provides a significant reduction of the number of multiplication operations, which are known to be costly in terms of energy consumption. We provide applications of MAPI to PCA-based image reconstruction as well as to graph-based ranking algorithms. When compared to RPI, MAPI not only typically converges much faster, but also provides superior performance.

Index Terms— Power iteration, multiplication-free algorithms, principal component analysis, PageRank algorithm.

1. INTRODUCTION

Let $\mathbf{A}$ be a diagonalizable matrix and $\mathbf{b}$ be some initial vector. Power iteration is described by the update equation or recurrence relation

$$\mathbf{b} \leftarrow \frac{\mathbf{Ab}}{\|\mathbf{Ab}\|}, \tag{1}$$

where $\| \cdot \|$ represents the Euclidean norm. It is well-known that Eq. (1) converges to the eigenvector of $\mathbf{A}$ corresponding to the dominant eigenvalue. Note that the normalization can be omitted, i.e., $\mathbf{b} \leftarrow \mathbf{Ab}$ results in the same direction as the dominant eigenvector.

A primary application of the power iteration Eq. (1) is conventional (or, L^2) principal component analysis (PCA): Suppose that we collect members of a zero-mean D-dimensional dataset $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \subset \mathbb{R}^D$ to a $D \times N$ matrix $\mathbf{X} = [\mathbf{x}_1 \ \mathbf{x}_2 \ ... \ \mathbf{x}_N] \in \mathbb{R}^{D \times N}$. Consider the corresponding sample covariance matrix

$$\mathbf{C} = \frac{1}{N}\mathbf{XX}^T. \tag{2}$$

The first principal vector, say $\mathbf{p}_1$, is then the dominant eigenvector of $\mathbf{C}$. The eigenvector $\mathbf{p}_1$ can be extracted via the power iteration in Eq. (1) applied with the substitution $\mathbf{A} = \mathbf{C}$. The ith principal vector can then be extracted via power iteration as the dominant eigenvector of $(\mathbf{I} - \sum_{k=1}^{i-1} \mathbf{p}_k\mathbf{p}_k^T)\mathbf{C}$.

In the process of computing the PCA of $\mathbf{C}$ through Eq. (1), one effectively computes the Euclidean inner product of the candidate principal vector $\mathbf{b}$ with the dataset elements $\mathbf{x}_i, \ i = 1, \ldots, N$ at each iteration. The fundamental underlying operation for each such Euclidean inner product is the multiplication of the components of $\mathbf{b}$ with the components of $\mathbf{x}_i$. On the other hand, it is well-known that multiplication operation will overamplify the effects of outliers or noise in the dataset. To increase robustness, our idea is to replace the Euclidean inner products that appear in Eq. (1) with Multiplication Avoiding Vector Products (MAVPs), which were originally developed in in the context of neural networks [1–6]. Two of the MAVPs satisfy the Mercer-type kernel requirement [7, 8]. The resulting multiplication-avoiding power iteration (MAPI) becomes a power iteration in the Reproducing Kernel Hilbert Space (RKHS) defined by the kernel.

Energy efficiency and improved computational complexity provides another major motivation for utilizing MAPIs in lieu of ordinary power iteration. In fact, MAVPs rely only on minimum operations and sign changes, and avoid the energy consuming multiplication operation. Therefore, compared to an Euclidean inner product, an MAVP can be executed in an energy efficient manner in many processors. The same benefits transfer to MAPI, which utilize MAVPs.

Kernel-PCA was introduced by Scholkopf et al. [8, 9]. It has been applied to many signal and image processing problems [10–13]. Related work includes recursive ℓ_1-PCA and PCA methods using the similarity measures related with the ℓ_1-norm [14–19]. However all of the above mentioned methods are computationally costly in large covariance matrices because they either require the eigen-decomposition of the covariance matrix or the solution of a complex optimization problem. To the best of our knowledge, this paper is the first paper describing power iterations in the kernel domain. This is probably due to the fact that other kernel PCA methods require costly kernel computations compared to the regular vector dot product [13,20–23]. On the other hand, MAPI kernel operations are more energy efficient than the dot product. In the context of PCA, MAPI can not only improve robustness but provide significant improvements in computational complexity. MAPI provides similar benefits in other applications in which the power iteration is used. Examples will be provided throughout the paper.

The rest of the paper is organized as follows: In Section 2, we formally introduce the MAPI. In Section 3, we describe applications of MAPI and report corresponding results. Finally, in Section 4, we draw our main conclusions.

2. METHODOLOGY

In this section, we provide an overview of the MAVPs, and formally introduce the MAPI method that utilizes MAVPs.

2.1. Multiplication-Avoiding Vector Products (MAVPs)

MAVPs were first studied in [1, 24] to develop a robust region co-variance matrix. They were used in computationally efficient neural networks. In this work, we will utilize the MAVP as

$$\mathbf{w}^T \odot \mathbf{x} \triangleq \sum_{i=1}^{n} \text{sign}(w_i x_i) \min(|w_i|, |x_i|). \tag{3}$$

We define another related dot-product as follows:

$$\mathbf{w}^T \odot_m \mathbf{x} \triangleq \sum_{i=1}^{D} \mathbf{1}\left(\text{sign}(w_i) = \text{sign}(x_i)\right) \min(|w_i|, |x_i|) \tag{4}$$

where $\mathbf{1}(\cdot)$ is the indicator function. We call Eq. (3) as min1 operation and Eq. (4) as min2 operation. In the following sessions, we will denote the two min-operations as $\oplus = \{\odot, \odot_m\}$. Note that unlike an ordinary Euclidean inner product, Eq. (3) and (4) do not contain any multiplication operations. Energy efficiency of $\oplus$ operation varies from processor to processor. For example, multiplication-based regular dot-product operation consumes about 4 times more energy compared to the multiplication avoiding dot product operations defined in Eq. (3) and (4) in compute-in-memory (CIM) implementation at 1 GHz operating frequency [6].

The MAVPs induce the ℓ_1-norm as

$$\mathbf{x}^T \oplus \mathbf{x} = \sum_{i=1}^{n} \min(|x_i|, |x_i|) = \|\mathbf{x}\|_1. \tag{5}$$

The MAVP in Eq. (3) and (4) can be extended to matrix multiplications as follows: Let $\mathbf{W} \in \mathbb{R}^{n \times m}$ and $\mathbf{X} \in \mathbb{R}^{n \times p}$ be arbitrary matrices, then

$$\mathbf{W}^T \oplus \mathbf{X} \triangleq \begin{bmatrix} \mathbf{w}_1^T \oplus \mathbf{x}_1 & \mathbf{w}_1^T \oplus \mathbf{x}_2 & \dots & \mathbf{w}_1^T \oplus \mathbf{x}_p \\ \mathbf{w}_2^T \oplus \mathbf{x}_1 & \mathbf{w}_2^T \oplus \mathbf{x}_2 & \dots & \mathbf{w}_2^T \oplus \mathbf{x}_p \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{w}_m^T \oplus \mathbf{x}_1 & \mathbf{w}_m^T \oplus \mathbf{x}_2 & \dots & \mathbf{w}_m^T \oplus \mathbf{x}_p \end{bmatrix} \tag{6}$$

where $\mathbf{w}_i$ is the ith column of $\mathbf{W}$ for $i = 1, 2, \dots, m$ and $\mathbf{x}_j$ is the jth column of $\mathbf{X}$ for $j = 1, 2, \dots, p$. In brief, the definition is similar to the matrix multiplication $\mathbf{W}^T \mathbf{X}$ by only changing the element-wise product to element-wise min-operation.

Recall from Section 1 that given a dataset $\mathbf{X} = [\mathbf{x}_1 \mathbf{x}_2 ... \mathbf{x}_N] \in \mathbb{R}^{D \times N}$, the eigendecomposition of $\mathbf{C} = \frac{1}{N}\mathbf{X}\mathbf{X}^T$ yields the ordinary PCA. Alternatively, one can construct the sample covariance matrix through MAVPs. Recall from Section 1 that given a dataset $\mathbf{X} = [\mathbf{x}_1 \mathbf{x}_2 ... \mathbf{x}_N] \in \mathbb{R}^{D \times N}$, the eigendecomposition of $\mathbf{C} = \frac{1}{N}\mathbf{X}\mathbf{X}^T$ yields the ordinary PCA. Alternatively, one can construct the sample covariance matrix through MAVPs. In specie, in [19], we considered the eigendecomposition of the "min-covariance matrix"

$$\mathbf{A} = \frac{1}{N}\mathbf{X} \oplus \mathbf{X}^T \tag{7}$$

We have shown in [19] that the resulting "Min-PCA" provides better resilience against impulsive noise than regular PCA in image reconstruction experiments.

2.2. Multiplication-Avoiding Power Iteration (MAPI)

We are now ready to introduce the MAPI. We replace the standard products in Eq. (1) with a MAVP. To further reduce the computational complexity, we replace the normalization by ℓ_2-norm by a normalization by ℓ_1-norm. Our revisions yield the iteration

$$\mathbf{w} \leftarrow \frac{\mathbf{A} \oplus \mathbf{w}}{\|\mathbf{A} \oplus \mathbf{w}\|_1} \tag{8}$$

We have observed that such a change of normalization does not effect the final performance greatly. The final MAPI algorithm is summarized in Algorithm 1. For applications to PCA, we can normalize the final $\mathbf{w}_t$ by its l_2-norm for extraction of subsequent principal vectors.

Algorithm 1 MAPI

Input: $\mathbf{A} \in \mathbb{R}^{N \times N}$, iteration times T.
Output: Dominant-pseudo-eigenvector $\mathbf{w}_T \in \mathbb{R}^{N \times 1}$.
1: Initialize $\mathbf{w}_0$ as a random vector with $\|\mathbf{w}_0\| = 1$;
2: **for** $t = 0, 1, ..., T - 1$ **do**
3:     $\mathbf{w}_{t+1} = \mathbf{A} \oplus \mathbf{w}_t$;
4:     $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_{t+1}/\|\mathbf{w}_{t+1}\|_1$;
5: **end for**
6: (optional) $\mathbf{w}_T = \mathbf{w}_T/\|\mathbf{w}_T\|_2$;
7: **return** $\mathbf{w}_T$.

Each step of the MAPI defined in Eq. 8 corresponds to a transformation in the RKHS. As a result the convergence of MAPI depends on the matrix $\mathbf{A}$. Since we normalize the iterations in (8) at each step, the iterates are bounded and they satisfy $\|\mathbf{w}_t\|_1 = 1$. We have observed that the MAPI in Algorithm 1 converges in all experiments that we have tried and the resulting vector can be used in practical applications. Note that MAPI only requires N divisions per iteration as opposed to the RPI, which requires N^2 multiplications and N divisions. Therefore, we expect MAPI to consume significantly less energy compared to RPI in most processors.

3. APPLICATIONS AND NUMERICAL RESULTS

3.1. Image Reconstruction Example

We consider image reconstruction example studied in [14]. Our experiment follows the same structure in [14, 19]. We use Regular Power Iteration (RPI) and MAPI to compare the image reconstruction results of the MAPI. In this experiment, the image size $D \times D = 128 \times 128$, and the pixel values are in the range of $[0, 1]$. Suppose that we want to reconstruct a gray-scale image (Fig. 1a) from its $N = 10$ occluded versions. As Fig. 1b shows, the corrupted images are created by partitioning the original image into sixteen tiles of size 32×32 and replacing three arbitrarily selected tiles by 32×32 gray-scale noise patches. The noise patches are generated using the uniform distribution in the interval $[0, 1]$. The reconstruction algorithm is described in Algorithm 2. In particular, in Step 5 and Step 6, we obtain the first two dominant generalized eigenvectors of the min-covariance matrix of images via MAPI. We then reconstruct the image using these two generalized eigenvectors in Line 8.

In Table 1, we compare the peak signal-to-noise ratio (PSNR) performances provided by different algorithms. As Fig. 1d and 1e show, MAPI provides higher PSNRs than RPI, globally. The average PSNR of min2-PI is 1.62 dB higher than RPI. The MAPI based

---

Algorithm 2 Image Reconstruction via MAPI

Input: N corrupted versions of an image: $\mathbf{I}_1, \mathbf{I}_2, ..., \mathbf{I}_N \in \mathbb{R}^{D \times D}$, pixels are in range of $[0, 1]$.

Output: Reconstructed image $\hat{\mathbf{I}}$.

1: Reshape $\mathbf{I}_i$ into the column vector form $\mathbf{v}_i \in \mathbb{R}^{D^2 \times 1}$ for i from 1 to N;

2: $\mathbf{V} = [\mathbf{v}_1 \ \mathbf{v}_2 \ ... \ \mathbf{v}_N] \in \mathbb{R}^{D^2 \times N}$;

3: Reduce mean: $\mathbf{m} = \text{mean}(\mathbf{V}, 2) \in \mathbb{R}^{D^2 \times 1}, \mathbf{V} = \mathbf{V} - \mathbf{m}$;

4: Construct the min-covariance matrix of $\mathbf{V}$: $\mathbf{C} = \frac{1}{N-1}\mathbf{V} \oplus \mathbf{V}^T \in \mathbb{R}^{D^2 \times D^2}$;

5: Perform MAPI on $\mathbf{C}$ to get the dominant eigenvector $\mathbf{w}_1 \in \mathbb{R}^{D^2 \times 1}$;

6: Perform MAPI on $\mathbf{C} - (\mathbf{w}_1 \oplus \mathbf{w}_1^T)\mathbf{C}$ to get the second dominant eigenvector $\mathbf{w}_2 \in \mathbb{R}^{D^2 \times 1}$;

7: $\mathbf{w} = [\mathbf{w}_1 \ \mathbf{w}_2] \in \mathbb{R}^{D^2 \times 2}$;

8: Reconstruct image: $\hat{\mathbf{v}} = (\mathbf{w} \oplus \mathbf{w}^T)(\mathbf{v}_i - \mathbf{m}) + \mathbf{m}$;

9: Reshape $\hat{\mathbf{v}}$ back to the matrix form $\hat{\mathbf{I}}$;

10: **return** $\hat{\mathbf{I}}$.

---

reconstruction (24.8 dB) is also superior to Recursive L1-PCA (24.2 dB). MAPI is as robust as L1-PCA based restoration and its computational cost is much lower than the regular L1-PCA methods.

Table 1: Image Reconstruction PSNRs (dB)

Image	Occluded	Recursive L1-PCA	RPI	Min1-PI	Min2-PI
Statue 1	17.67	26.86	26.67	**27.12**	27.08
Statue 2	16.87	25.02	24.61	**25.45**	25.34
Earth	14.91	21.74	22.13	20.78	**25.03**
Pikachu	15.29	**22.71**	18.86	22.70	22.58
Flower	16.40	24.41	21.22	**24.59**	24.44
Orange	15.77	23.69	23.59	22.50	**26.27**
Cat	16.91	**24.82**	24.80	24.28	24.52
Lenna	16.77	24.71	22.86	**25.59**	24.73
Food	15.94	23.85	22.72	23.68	**23.70**
Car	15.42	23.37	23.15	**23.43**	23.41
Cobra	16.81	25.10	22.31	25.02	**25.57**
River	17.27	25.23	24.50	25.07	**25.88**
Butterfly	16.66	**24.91**	24.40	23.93	23.92
Bridge	15.66	22.91	22.23	22.55	**24.34**
Average	16.31	24.24	23.15	24.05	**24.77**

3.2. Comparison with Stochastic PCA Method

In this section, we will compare the MAPI with the stochastic power iteration method on a synthetic dataset as in [25]. The synthetic dataset $\mathbf{X} \in \mathbb{R}^{10^6 \times 10}$ is generated using the singular value decomposition. In detail, let a diagonal matrix $\mathbf{\Sigma} = \text{diag}\{1, \sqrt{0.9}, ..., \sqrt{0.9}\} \in \mathbb{R}^{10 \times 10}$, a random orthogonal projection matrix $\mathbf{U} \in \mathbb{R}^{10^6 \times 10}$ and a random orthogonal matrix $\mathbf{V} \in \mathbb{R}^{10 \times 10}$,

then the dataset $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T$ guarantees that the matrix $\mathbf{C} = \mathbf{X}^T\mathbf{X}$ has an eigen-gap of 0.1.

MAPI method can be also implemented into the mini-batch power method with momentum algorithm (Algorithm 1 in [25]). We take the MAPI on $\mathbf{A} = \mathbf{X}^T \odot \mathbf{X}$ with the same mini-batch strategy, as Algorithm 3 shows. The different between Algorithm 3 in this paper and Algorithm 1 in [25] is that we change all regular multiplication operations with our min1 operation Eq. (3) and normalize the iterates using the ℓ_1-norm. At the end of iterations we normalize the final vector using the ℓ_2-norm of the vector. In this experiment, we cannot use min2-PI operation because the min2 operation Eq. (4) cannot return negative entries. The vector $\mathbf{u}_1$ can have negative entries.

In Fig. 2, we compute $(1 - (\mathbf{w}_t^T\mathbf{u}_1/||\mathbf{w}_t||)^2)$ after each iteration, where $\mathbf{u}_1$ is the dominant eigenvector obtained from eigendecomposition as in [25]. The MAPI iteration converges to a vector very close to the actual eigenvector because $(1 - (\mathbf{w}_t^T\mathbf{u}_1/||\mathbf{w}_t||)^2)$ is very close to zero as shown in Fig. 2a for t greater than 20. However, $\mathbf{w}_T$ obtained using the MAPI method is not exactly the same as the eigenvector $\mathbf{v}_1$ as we see from Fig. 2b whose vertical axis has a different range from Fig. 2a. This is expected because we perform the iterations in the Reproducing Kernel Hilbert Space (RKHS) domain. Nevertheless, based on our observation, ranks of the entries of $\mathbf{w}_T$ obtained from the min1-PI are the same as the ranks obtained from $\mathbf{w}_T$ of the RPI. After 100 iterations, $\mathbf{w}_T$=[-0.1433, 0.4171, -0.1166, 0.3863, 0.1285, -0.1315, -0.4330, -0.2097, -0.5770, -0.2106] from the RPI and $\mathbf{w}_T$=[-0.1649, 0.4166, -0.1371, 0.3887, 0.1480, -0.1508, -0.4306, -0.2359, -0.5362, -0.2370] from the min1-PI. When we order from the largest to the smallest, both vectors produce the same ranks {2, 4, 5, 3, 6, 1, 8, 10, 7, 9}. Therefore, considering that our min1-PI converges significantly faster than the conventional power iteration and the min2-PI is identical to the min1-PI if all values are non-negative, our MAPI can be employed in the Google PageRank algorithm [26]. We will discuss it in Section 3.3. Moreover, if we want $(1 - (\mathbf{w}_t^T\mathbf{u}_1/||\mathbf{w}_t||)^2)$ to reach 0, we can optimize the vector via the MAPI first and then switch to the RPI for further converging.

Algorithm 3 Mini-batch MAPI with Momentum

Input: Data $\mathbf{X} \in \mathbb{R}^{N \times D}$, iteration times T, batch size s, momentum parameter β.

Output: Dominant-pseudo-eigenvector $\mathbf{w}_T \in \mathbb{R}^{N \times 1}$.

1: Initialize $\mathbf{w_0}$ as a random vector with $||\mathbf{w_0}|| = 1$;

2: **for** $t = 0, 1, ..., T - 1$ **do**

3:     Generate a mini batch of independent and identically distributed samples $B = \{\tilde{\mathbf{A}}_{t_1}, ..., \tilde{\mathbf{A}}_{t_s}\}$

4:     $\mathbf{w}_{t+1} = (\frac{1}{s}\sum_{i=1}^s \tilde{\mathbf{A}}_{t_i}) \odot \mathbf{w}_t - \beta\mathbf{w}_{t-1}$;

5:     $\mathbf{w}_t = \mathbf{w}_t/||\mathbf{w}_{t+1}||_1, \mathbf{w}_{t+1} = \mathbf{w}_{t+1}/||\mathbf{w}_{t+1}||_1$;

6: **end for**

7: (optional) $\mathbf{w}_T = \mathbf{w}_T/||\mathbf{w}_T||$;

8: **return** $\mathbf{w}_T$.

3.3. PageRank Algorithm Using MAPI

PageRank algorithm uses the hyperlink structure of the web to view inlinks into a page as a recommendation of that page from the author of the inlinking page [27]. More specifically, it runs power iteration
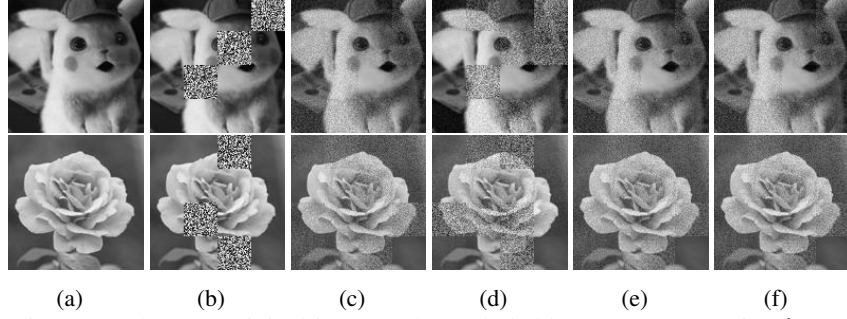
Fig. 1: Image reconstruction examples: (a) original images; (b) occluded images; (c) recursive ℓ_1-PCA [14] results; (d) PCA using RPI results; (e) min1-PI results; (f) min2-PI results. MAPI is as robust as L1-PCA based restoration.
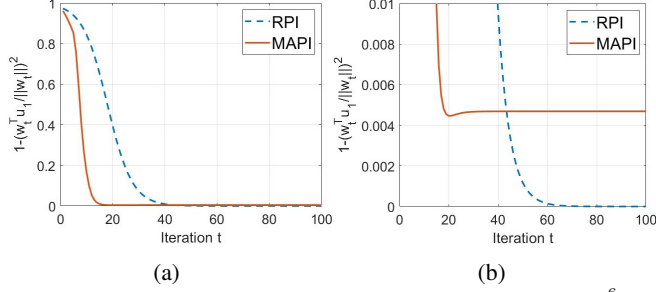

Fig. 2: RPI versus MAPI on a synthetic dataset $\mathbf{X} \in \mathbb{R}^{10^6 \times 10}$ where the covariance matrix has eigen-gap $\Delta = 0.1$.

algorithm on the Google Matrix

$$\mathbf{G} = \alpha \mathbf{H} + (1 - \alpha) \mathbf{1} \frac{1}{N} \qquad (9)$$

where $\mathbf{H}$ is the network adjacent matrix, $\mathbf{1}$ is an all-ones matrix, and $\alpha = 0.85$ is known as a damping factor.

We first run the PageRank algorithm with RPI and with our MAPI on a network graph as Fig. 3a, respectively. Then, we compute $\|\mathbf{w}_t - \mathbf{w}_{t-1}\|$ after each iteration to compare the convergence.
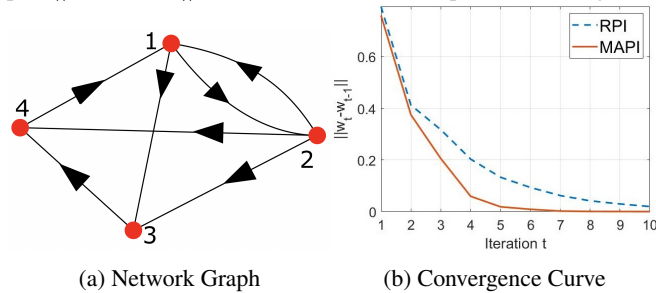

(a) Network Graph   (b) Convergence Curve
Fig. 3: A PageRank example.

We apply one ℓ_2-normalization after the final iteration for easier comparison. After 20 iterations, from the RPI, $w_T = [0.6335, 0.3452, 0.4399, 0.5348]$, and from the MAPI, $w_T = [0.5675, 0.3486, 0.5201, 0.5347]$. Consequently, though the weights are different, both index ranks are $\{1, 4, 3, 2\}$.

We further try the PageRank algorithm with the RPI and with the MAPI on Gnutella peer-to-peer network datasets (Gnutella08 [28], Gnutella09 [29]). Gnutella08 contains 6,301 nodes with 20,777 edges, and Gnutella09 contains 8,114 nodes with 26,013 edges. The convergence curves are shown in Fig. 4. The MAPI still converges remarkably faster than the RPI on these large network datasets. However, because the size of the network is very large, the ranks of the two methods are not the same. After 10 iterations, from

Gnutella08, the top-10 ranks of the indices from the RPI are $\{367, 249, 145, 264, 266, 123, 127, 122, 1317, 5\}$, while the MAPI $\{266, 123, 367, 127, 424, 249, 145, 264, 427, 251\}$. On the other hand, from Gnutella09, the top-10 ranks of the indices from the RPI are $\{351, 563, 822, 534, 565, 825, 1389, 1126, 356, 530\}$, while the MAPI returns $\{351, 822, 51, 1389, 563, 565, 530, 825, 356, 1074\}$. Therefore, there are 7 common top-10 ranks of indices $\{367, 249, 145, 265, 226, 123, 127\}$ from the 6,301-node dataset Gnutella08 and 8 common top-10 ranks of indices $\{351, 563, 822, 565, 825, 1389, 356, 530\}$ from the 8,114-node dataset Gnutella09. Therefore, if we implement the MAPI in the page-rank-based web search system, what links displayed on the first page is very close to the conventional page-rank based system, but the search time can be reduced efficiently.
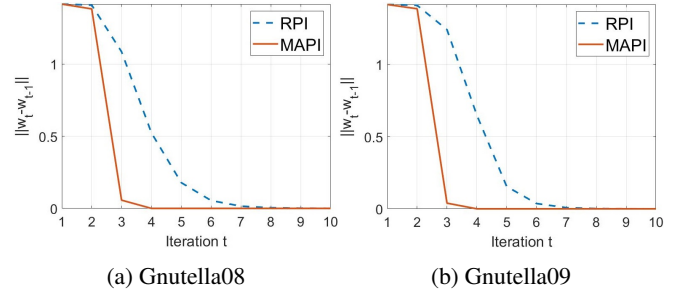

(a) Gnutella08   (b) Gnutella09
Fig. 4: PageRank convergence curves on Gnutella.

4. CONCLUSION

In this paper, we proposed two types of MAPI algorithms which replace the standard vector product operations in regular power iteration (RPI) with multiplication-free vector products performed in RKHS defined by the kernel. The MAPI is energy efficient because it reduces the number of multiplication operations significantly, which are known to be costly in terms of energy consumption in many processors. The MAPI reduces the number of multiplications from n^2 multiplications to n multiplications per iteration for an $n \times n$ matrix. The MAPI also produces as robust results as the ℓ_1-PCA algorithm because MAPI kernels are related with the ℓ_1-norm. Compared to the RPI, the min2-PI reaches an average PSNR 1.62 dB higher in our image reconstruction experiment. According to our graph-based ranking experiment, the MAPI is superior to the RPI in terms of convergence speed and energy efficiency. Though the final ranking obtained from the MAPI is not always exactly identical to the ranking obtained from the RPI but they are very close to each other.

5. REFERENCES

[1] Hakan Tuna, Ibrahim Onaran, and A Enis Cetin. Image description using a multiplier-less operator. *IEEE Signal Processing Letters*, 16(9):751–753, 2009.

[2] Arman Afrasiyabi, Baris Nasir, Ozan Yildiz, Fatos T Yarman Vural, and A Enis Cetin. An energy efficient additive neural network. In *2017 25th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4. IEEE, 2017.

[3] Arman Afrasiyabi, Diaa Badawi, Baris Nasir, Ozan Yildi, Fatos T Yarman Vural, and A Enis Çetin. Non-euclidean vector product for neural networks. In *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6862–6866. IEEE, 2018.

[4] Hongyi Pan, Diaa Badawi, Xi Zhang, and Ahmet Enis Cetin. Additive neural network for forest fire detection. *Signal, Image and Video Processing*, pages 1–8, 2019.

[5] Tolga Ergen, Ali H Mirza, and Suleyman Serdar Kozat. Energy-efficient lstm networks for online learning. *IEEE transactions on neural networks and learning systems*, 31(8):3114–3126, 2019.

[6] Shamma Nasrin, Diaa Badawi, Ahmet Enis Cetin, Wilfred Gomes, and Amit Ranjan Trivedi. Mf-net: Compute-in-memory sram for multibit precision inference using memory-immersed data conversion and multiplication-free operators. *IEEE Trans. Circuits and Systems*, 2021.

[7] Hongyi Pan, D Badawi, E Koyuncu, and AE Cetin. Robust principal component analysis using a novel kernel related with the l1-norm. In *to appear in Proceedings of EUSIPCO - European Signal Processing Conference; a longer version is available at arxiv.org/abs/2105.11634*, 2021.

[8] B. Schölkopf, A. Smola, and K. Müller. Kernel principal component analysis. In *Artificial Neural Networks — ICANN'97*, Lecture Notes in Computer Science, vol 1327, Heidelberg, Berlin, Germany, 1997.

[9] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319, 1998.

[10] H Goldberg, Heesung Kwon, and Nasser M Nasrabadi. Kernel eigenspace separation transform for subspace anomaly detection in hyperspectral imagery. *IEEE Geoscience and Remote Sensing Letters*, 4(4):581–585, 2007.

[11] Nasser M Nasrabadi. Hyperspectral target detection: An overview of current and future challenges. *IEEE Signal Processing Magazine*, 31(1):34–44, 2013.

[12] Hien Van Nguyen, Vishal M Patel, Nasser M Nasrabadi, and Rama Chellappa. Kernel dictionary learning. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2021–2024. IEEE, 2012.

[13] Heiko Hoffmann. Kernel pca for novelty detection. *Pattern recognition*, 40(3):863–874, 2007.

[14] Panos P Markopoulos, George N Karystinos, and Dimitris A Pados. Optimal algorithms for l_1-subspace signal processing. *IEEE Transactions on Signal Processing*, 62(19):5046–5058, 2014.

[15] P. P. Markopoulos, G. N. Karystinos, and D. A. Pados. Optimal algorithms for l1-subspace signal processing. *IEEE Transactions on Signal Processing*, 62(19):5046 – 5058, Jul. 2014.

[16] Panos P Markopoulos, Sandipan Kundu, Shubham Chamadia, and Dimitris A Pados. Efficient l1-norm principal-component analysis via bit flipping. *IEEE Transactions on Signal Processing*, 65(16):4252–4264, 2017.

[17] Panos P Markopoulos and Fauzia Ahmad. Indoor human motion classification by l1-norm subspaces of micro-doppler signatures. In *2017 IEEE Radar Conference (RadarConf)*, pages 1807–1810. IEEE, 2017.

[18] Panos P Markopoulos, Mayur Dhanaraj, and Andreas Savakis. Adaptive l1-norm principal-component analysis with online outlier rejection. *IEEE Journal of Selected Topics in Signal Processing*, 12(6):1131–1143, 2018.

[19] Hongyi Pan, Diaa Badawi, Erdem Koyuncu, and A Enis Cetin. Robust principal component analysis using a novel kernel related with the l1-norm. *arXiv preprint arXiv:2105.11634*, 2021. Presented in EUSIPCO 2021.

[20] Y Xiao, Huangang Wang, and Wenli Xu. Model selection of gaussian kernel pca for novelty detection. *Chemometrics and Intelligent Laboratory Systems*, 136:164–172, 2014.

[21] Cheolmin Kim and Diego Klabjan. L1-norm kernel pca. *arXiv preprint arXiv: 1709.10152*, 2017.

[22] Carolina Varon, Carlos Alzate, and Johan AK Suykens. Noise level estimation for model selection in kernel pca denoising. *IEEE transactions on neural networks and learning systems*, 26(11):2650–2663, 2015.

[23] Samuele Battaglino and Erdem Koyuncu. A generalization of principal component analysis. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3607–3611. IEEE, 2020.

[24] Cem Emre Akbas, Alican Bozkurt, Musa Tunc Arslan, Huseyin Aslanoglu, and A Enis Cetin. L1 norm based multiplication-free cosine similarity measures for big data analysis. In *2014 International Workshop on Computational Intelligence for Multimedia Understanding (IWCIM)*, pages 1–5. IEEE, 2014.

[25] Peng Xu, Bryan He, Christopher De Sa, Ioannis Mitliagkas, and Chris Re. Accelerated stochastic power iteration. In *International Conference on Artificial Intelligence and Statistics*, pages 58–67. PMLR, 2018.

[26] Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab, 1999.

[27] Catherine Benincasa, Adena Calden, Emily Hanlon, Matthew Kindzerske, Kody Law, Eddery Lam, John Rhoades, Ishani Roy, Michael Satz, Eric Valentine, et al. Page rank algorithm. *Department of Mathematics and Statics, University of Massachusetts, Amherst, Research*, 2006.

[28] Gnutella peer-to-peer network. https://snap.stanford.edu/data/p2p-Gnutella08.html Accessed: 2021-10-03.

[29] Gnutella peer-to-peer network. https://snap.stanford.edu/data/p2p-Gnutella09.html Accessed: 2021-10-03.