L1 NORM BASED MULTIPLICATION-FREE COSINE SIMILARITY MEASURES FOR BIG DATA ANALYSIS

Cem Emre Akbas, Alican Bozkurt, Musa Tunc Arslan, Huseyin Aslanoglu, A. Enis Cetin

Department of Electrical and Electronic Engineering
Bilkent University, 06800 Bilkent, Ankara, Turkey
akbas@ee.bilkent.edu.tr, alican@ee.bilkent.edu.tr, mtarslan@ee.bilkent.edu.tr, hsynaslanoglu@gmail.com, cetin@bilkent.edu.tr

ABSTRACT

The cosine similarity measure is widely used in big data analysis to compare vectors. In this article a new set of vector similarity measures are proposed. New vector similarity measures are based on a multiplication-free operator which requires only additions and sign operations. A vector 'product' using the multiplication-free operator is also defined. The new vector product induces the $\ell 1$-norm. As a result, new cosine measure-like similarity measures are normalized by the $\ell 1$-norms of the vectors. They can be computed using the MapReduce framework. Simulation examples are presented.

Index Terms— Multiplication-free operator, cosine similarity, big data, MapReduce.

1. INTRODUCTION

The cosine similarity between two vectors is computed using the inner product of two vectors divided by the $\ell 2$-norms of the vectors. It is widely used in big data analysis [1, 2]. In this article, a set of low-power vector similarity measures are proposed to compare large vectors. The new measures are low-power measures because they can be computed without performing any multiplications. In C++, 1 million addition operations take 18 milliseconds, while 1 million multiplication operations take 25 milliseconds on AMD Phenom II X6 1090T Processor. Computing infrastructures should be as low power as possible to make big data analysis as economical as possible [3].

Regular cosine similarity measure is based on multiplication as shown in (1).

$$cos(\mathbf{x}, \mathbf{y}) = \frac{< \mathbf{x} \cdot \mathbf{y} >}{\|\mathbf{x}_1\|_2 \cdot \|\mathbf{y}_1\|_2}, \tag{1}$$

The goal of this paper is to reduce the number of multiplications as much as possible.

We recently introduced a new multiplication-free vector product [4–8] . It is based on an additive operator. Let a and b be two real numbers. The new operator is defined as follows:

$$a \odot b = sign(a \times b) \cdot (|a| + |b|), \tag{2}$$

where

$$sign(a \times b) = \begin{cases} 1, & \text{if } a \cdot b > 0, \\ 0, & \text{if } a \cdot b = 0, \\ -1, & \text{if } a \cdot b < 0. \end{cases} \tag{3}$$

The operator $\odot$ is basically a summation operation. However, the sign of the result of $a \odot b$ is the same as $a \times b$. Therefore, the $\odot$ operator behaves like the multiplication operation.

We define a new "vector product" of two N-dimensional vectors $\mathbf{x}_1$ and $\mathbf{x}_2$ in R^N as follows:

$$< \mathbf{x}_1 \odot \mathbf{x}_2 >= \sum_{i=1}^{N} x_1(i) \odot x_2(i), \tag{4}$$

where $\mathbf{x}_1(i)$ and $\mathbf{x}_2(i)$ are the i-th entries of the vectors $\mathbf{x}_1$ and $\mathbf{x}_2$, respectively. Notice that the vector product of a vector $\mathbf{x}$ with itself reduces to a scaled l_1 norm of $\mathbf{x}$ as follows:

$$< \mathbf{x} \odot \mathbf{x} >= \sum_{i=1}^{N} x(i) \odot x(i) = 2 \sum_{i=1}^{N} |x(i)| = 2||\mathbf{x}||_1. \tag{5}$$

Other related vector products are defined in Seciton 4.

Organization of the paper is as follows. The $\ell 1$ norm based new multiplication-free cosine similarity measures are defined in Section 2. Properties of these new vector comparison measures are discussed in Section 3. In Section 4, hash function generation using the multiplication-free operators is described. In Section 5, simulation examples are presented. In Section 6, conclusions are drawn.

2. SIMILARITY MEASURE DEFINITIONS

Based on the new vector product, we define several vector similarity measures between the two vectors x and y as follows:

$$c_1(\mathbf{x}, \mathbf{y}) \triangleq \frac{< \mathbf{x} \odot \mathbf{y} >}{\|\mathbf{x}\|_1 + \|\mathbf{y}\|_1}, \tag{6}$$

where $\|\mathbf{x}\|_1$ and $\|\mathbf{y}\|_1$ are the ℓ_1 norms of vectors $\mathbf{x}$ and $\mathbf{y}$, respectively. Similar to the ordinary cosine similarity measure, the numerator contains the vector product of the two vectors and the vector product is normalized by the sum of the ℓ_1 norms of the two vectors $\mathbf{x}$ and $\mathbf{y}$. When $\mathbf{x} = \mathbf{y}$, $c_1(\mathbf{x}, \mathbf{y}) = 1$ because $< \mathbf{x} \odot \mathbf{x} > = 2\|x\|_1$.

It is also possible to define other related measures using relation operators:

$$c_2(\mathbf{x}, \mathbf{y}) \triangleq \frac{\sum_{i=1}^{N} sign(x_i \times y_i) \cdot max(|x_i|, |y_i|)}{(\|\mathbf{x}\|_1 + \|\mathbf{y}\|_1)/2}, \tag{7}$$

where $sign(x_i \times y_i)$ is defined as in (3) and the vectors $\mathbf{x} = [x_1, x_2, ..., x_N]^T$ and $\mathbf{y} = [y_1,\ y_2,\ ...\ ,\ y_N]$, respectively.. A related third cosine similarity measure is defined as follows:

$$c_3(\mathbf{x}, \mathbf{y}) \triangleq \frac{\sum_{i=1}^{N} sign(x_i \times y_i) \cdot min(|x_i|, |y_i|)}{(\|\mathbf{x}\|_1 + \|\mathbf{y}\|_1)/2} \tag{8}$$

978-1-4799-7971-4/14/$31.00 ©2014 IEEE

where the maximum operation in (7) is replaced by the minimum operation. The similarity measure c_1 can also be normalized in a different manner as follows:

$$c_4(\mathbf{x},\mathbf{y}) \triangleq \frac{\sum_{i=1}^{N} sign(x_i \times y_i) \cdot (|x_i| + |y_i|)}{2 \cdot \sum_{i=1}^{N} max(|x_i|, |y_i|)} \tag{9}$$

Clearly, when $\mathbf{x} = \mathbf{y}$, $c_4(\mathbf{x},\mathbf{x}) = 1$. Similarly, c_2 and c_3 can also be normalized as in c_4.

In all similarity measures defined (6 - 9), the term $sign(x_i \times y_i)$ is common and provides the correlation information between the two vectors **x** and **y**. As a result, the computational complexity and power consumption due to signal analysis can be decreased significantly.

Obviously, $c_1(\mathbf{x},\mathbf{y}) = c_1(\mathbf{y},\mathbf{x})$ in all three definitions [7–10]. Also, when **x=y**, they all produce the same result, i.e., $c_1(\mathbf{x},\mathbf{y}) = c_2(\mathbf{x},\mathbf{y}) = c_3(\mathbf{x},\mathbf{y}) = c_4(\mathbf{x},\mathbf{y}) = 1$. Multiplication requires more power than addition and relational max or min operations. Since all three measures can be computed without performing any multiplications, they are all low-power vector similarity measures.

3. PROPERTIES OF NEW VECTOR COMPARISON MEASURES

It is easy to show that

$$-1 \leq c_1(\mathbf{x},\mathbf{y}) \leq 1, \tag{10}$$

Therefore, a vector similarity measure similar to the ordinary cosine similarity measure can be defined using Definition (6). As pointed out in Section I, when **x = y**, new similarity measures are all equal to 1:

$$c_1(\mathbf{x},\mathbf{x}) = \frac{2\|\mathbf{x}\|}{\|\mathbf{x}\| + \|\mathbf{x}\|} = 1, \tag{11}$$

and

$$c_2(\mathbf{x},\mathbf{x}) = c_3(\mathbf{x},\mathbf{x}) = c_4(\mathbf{x},\mathbf{x}) = \frac{\|\mathbf{x}\|}{\|\mathbf{x}\|} = 1 \tag{12}$$

It is possible to compute the new cosine similarity measures when the vectors **x** and **y** are big in size so that they do not fit in their entirety in the main memory of a computer. We can divide big vectors into stripes of equal width so that each portion of the vectors can fit into the main memory at a compute node in a cloud as shown in Figure 1.

Therefore, each Map task can be assigned a chunk of the "vector product" operation. The Map and Reduce tasks can then compute the vector product.

A related vector similarity measure defined by Kleinberg and Tardos [9, 10] is given by:

$$sim(\mathbf{x},\mathbf{y}) = \frac{\sum_{i=1}^{N} min(x_i, y_i)}{\sum_{i=1}^{N} max(x_i, y_i)}, \tag{13}$$

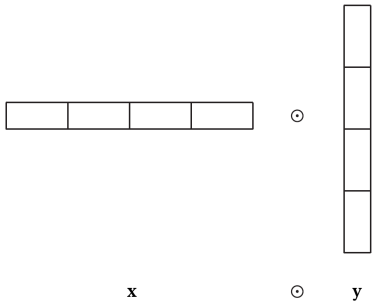

Fig. 1. Vector Product in MapReduce

Equation (13) cannot handle vectors with negative entries. When **x** or **y** is equal to an all zero vector, $sim(\mathbf{x},\mathbf{y})$ becomes

$$sim(\mathbf{x},0) = \frac{\sum_{i=1}^{N} min(x_i, 0)}{\sum_{i=1}^{N} max(x_i, 0)} = 1, \tag{14}$$

which is not a desirable result. The measure $sim(\mathbf{x},\mathbf{y})$ may deviate significantly from the cosine similarity measure for some **x**, **y** pairs.

Even if the vectors in a database contain only positive entries, it may be advantageous to subtract the means of the vectors from the vectors. It may be advantageous to compute the similarity on zero mean vectors as pointed out in [1]. The similarity measure defined in (6) is not useful when the entries of the vectors x and y are all positive or all (negative). In this case the measure produces +1 for all vectors with positive values. Therefore, means of the vectors must be removed for meaningful results to use c_1 properly. Other measures c_2, c_3 and c_4 can produce good similarity results for both non-negative and real valued vectors.

4. RANDOM HYPERPLANE BASED HASH FUNCTIONS BASED ON THE MULTIPLICATION-FREE VECTOR PRODUCT

Hash functions are a computationally efficient way to reduce the computational cost of big data analysis. Hash functions can reduce the dimension of the vectors. As a result, they reduce the computational cost of big data analysis. Due to large number of dimensions in big data problems, it is not practical to process the data as it is.

We define a family of hash functions given for a given set of vectors $\mathbf{u}_i, i = 1, 2, ..., L$ in R^N as follows. We generate M random vectors, $\mathbf{r}_1, \mathbf{r}_2, ..., \mathbf{r}_M$ whose entries are drawn from i.i.d., zero-mean Gaussian distributions with variance σ. Each random vector $\mathbf{r}_j, j = 1, 2, ..., M$ represents a hyperplane. For each random vector $\mathbf{r}_j$ and a given vector $\mathbf{u}_i$, we define a hash function $\mathbf{h}_r$ as follows:

$$\mathbf{h}_r(\mathbf{u}_i) = \begin{cases} 1, & \text{if} < \mathbf{r} \odot \mathbf{u}_i > \ \geq 0 \\ 0, & \text{if} < \mathbf{r} \odot \mathbf{u}_i > \ < 0 \end{cases} \tag{15}$$

Results of $\mathbf{h}_{r_j}(u_i)$, $j = 1, 2, ..., M$ are concatenated to construct an M-bit hash vector [11]. It is also possible to define new

vector products and use them to construct hash functions in this article based on maximum and minimum operations as follows:

$$< \mathbf{x} \oplus \mathbf{y} >= \sum_{i=1}^{N} sign(x_i \times y_i) \cdot max(|x_i|, |y_i|) \qquad (16)$$

and

$$< \mathbf{x} \ominus \mathbf{y} >= \sum_{i=1}^{N} sign(x_i \times y_i) \cdot min(|x_i|, |y_i|) \qquad (17)$$

The proposed vector products and cosine similarity measures are applied to hash vectors and the data can be classified using a k-nearest neighbor classifier. Simulation examples are presented in the next section.

5. EXPERIMENTAL RESULTS

The Gesture Phase Segmentation data set [12] was made available by UC Irvine Machine Learning Repository. The data set contains 5 classes and 1747 gesture phase data each having 18 attributes. In this paper, simulations are carried out using the first two classes (202 instances), the first three classes (900 instances) and the whole dataset (1747 instances).

In all simulation studies that use the Gesture Segmentation Dataset (Table: 1 - 6), we have a leave-one-out strategy. The size of the test set is one and the training set contains the remaining data. The test set is circulated to cover all instances. 1-nearest neighbor classification is done using c_1, c_2, c_3, c_4 and cosine similarity measures. Classification accuracies for the two class data is given in Table 1.

Table 1. Classification accuracies (Percentage) for the 2 class 1-nearest neighbor classification with 6 different similarity measures. The last row is the ordinary cosine similarity measure.

Similarity Measure	Classification Accuracy	Classification Accuracy with Zero-Mean Vectors
c_1	80.2	84.2
c_2	15.8	84.7
c_3	98.5	98.5
c_4	98.5	99.0
sim	92.8	94.5
cosine	98.0	97.5

As shown in Table 1, using zero-mean input vectors improves classification accuracies as compared to the classification accuracies with standard input vectors. In this case, c_1, c_2 and sim similarity measures classify the two class data worse than the ordinary cosine similarity measure. However, c_3 and c_4 similarity measures outperform the cosine similarity measure with both standard and zero-mean input vectors.

1-nearest neighbor classification accuracies for the three class data is given in Table 2.

Three class 1-nearest neighbor classification accuracies are similar to the previous case as shown in Table 2. c_1, c_2 and sim similarity measures fail to classify the data. However, c_3 and c_4 similarity measures classify the data better than the cosine similarity measure.

As shown in Table 3, the five class 1-nearest neighbor classification accuracies are similar to the two and three class case. c_1, c_2 and sim similarity measures classify the data worse than the cosine similarity measure. However, c_3 and c_4 similarity measures produces better results than the cosine similarity measure in this dataset.

Table 2. Classification accuracies (Percentage) for the 3 class 1-nearest neighbor classification with 6 different similarity measures. The last row is the ordinary cosine similarity measure.

Similarity Measure	Classification Accuracy	Classification Accuracy with Zero-Mean Vectors
c_1	5.4	28.8
c_2	1.9	21.8
c_3	97.9	97.8
c_4	97.9	97.8
sim	85.9	87.6
cosine	97.3	97.1

Table 3. Classification accuracies (Percentage) for the 5 class 1-nearest neighbor classification with 6 different similarity measures. The last row is the ordinary cosine similarity measure.

Similarity Measure	Classification Accuracy	Classification Accuracy with Zero-Mean Vectors
c_1	5.9	21.6
c_2	30.6	33.3
c_3	94.8	93.8
c_4	94.8	93.7
sim	68.8	69.9
cosine	92.7	92.2

Table 4. Classification accuracies (Percentage) for the 2 class 2-nearest neighbor classification with 6 different similarity measures. The last row is the ordinary cosine similarity measure.

Similarity Measure	Classification Accuracy	Classification Accuracy with Zero-Mean Vectors
c_1	81.2	84.2
c_2	15.8	84.7
c_3	97.5	98.0
c_4	97.6	98.0
sim	97.3	94.8
cosine	97.1	97.5

Two class 2-nearest neighbor classification accuracies are shown in Table 4. For c_1, c_2, c_3, c_4 and cosine similarity measures, classification accuracies are almost the same as in 1-nearest neighbor case (Table 1). However, sim similarity measure classifies the data better in 2-nearest neighbor case than in 1-nearest neighbor case.

As shown in Table 5, hashing is applied to the input vectors before classification. Since hash function uses random Gaussian numbers to hash the data, classification accuracies are clearly worse than no-hashing cases. In this case, c_2 and sim similarity measures classify the data worse than the cosine similarity measure and c_1, c_3 and c_4 similarity measures classify the data better than the cosine similarity measure. This is the first case that c_1 similarity measure can outperform the cosine similarity measure.

In Table 6, hashing is done using 6 different operators and classification is done using only the cosine similarity measure. Two class, 1-nearest neighbor classification with ordinarily hashed input vectors using cosine similarity measure is also presented in the last row

Table 5. Classification accuracies (Percentage) for the 2 class 1-nearest neighbor classification using 16 bit hashed input vectors with 6 different similarity measures. The last row is the ordinary cosine similarity measure.

Similarity Measure	Classification Accuracy	Classification Accuracy with Zero-Mean Vectors
c_1	73.3	74.8
c_2	70.8	71.8
c_3	73.8	76.7
c_4	72.1	73.8
sim	68.3	69.3
cosine	71.3	72.3

Table 6. Cosine similarity classification accuracies (Percentage) for 2 class 1-nearest neighbor classification with 16 bit hashed input vectors created by 6 different hashing operators. The last row is the ordinary hashing operation.

Hashing Operator	Classification Accuracy	Classification Accuracy with Zero-Mean Vectors
c_1	71.3	71.7
c_2	73.3	67.8
c_3	68.8	69.2
c_4	68.8	74.3
sim	60.7	61.7
dot product	68.1	70.3

of Table 5. Most of the alternative operators perform worse than the ordinary hashing operator. Hashing with c_1 and c_2 operators can outperform ordinary hashing operator only with standard input vectors.

5.1. Query Retrieval using TF-IDF Vectors Based on the Multiplication-Free Vector Similarity Measures

TF-IDF is a computationally efficient method to compute similarity of two text documents [13]. The TF-IDF matrix is computed as in [13]. Ordinary TF-IDF methods use the cosine similarity measure to compare TF-IDF vectors of documents. We also apply the new multiplication-free similarity measures defined in Equations (6 - 9) in order to further reduce the computational cost of query retrieval. Simulation examples are presented in the next section.

In this example, 964 randomly chosen song lyrics are used. The query is chosen as: "she", "believe", "peace", "again" and TF-IDF vector of query is constructed. Then, TF-IDF values of each keyword is computed for each song seperately in order to construct 964 TF-IDF vectors. Mean value of each TF-IDF vector is subtracted from itself in order to obtain zero-mean vectors. Similarity values between TF-IDF vector of each song and the TF-IDF vector of query is computed using 6 different similarity measures. First ten songs with highest similarity values are presented as query results. The similarity measure sim (Equation (13)) produces meaningless results with zero-mean vectors, therefore simulations are carried out with standard TF-IDF vectors in this case.

As shown in Table 7, c_1, c_2, c_3 and c_4 similarity measures can return the first ten songs the same as the cosine similarity does. The similarity measures c_1, c_2, c_3 and c_4 can make little mistakes by confusing the places of two consecutive songs. Among 5 approximate similarity measures, c_1 is the most successful one, which returns 8 query results in the same order as the cosine similarity does. sim (Equation (13)) similarity measure can return 8 songs the same as cosine similarity does, but it fails to return them in the right order.

Table 7. First ten songs with the highest TF-IDF similarity values in the order for query = "she believe peace again" using 5 different vector similarty measures. The first column is the ordinary cosine similarity measure.

Cosine	c_1	c_2	c_3	c_4	sim
#551	#551	#551	#551	#551	#834
#590	#590	#590	#590	#590	#829
#321	#321	#321	#805	#321	#443
#805	#805	#805	#321	#805	#590
#443	#443	#443	#834	#443	#421
#834	#834	#834	#443	#834	#350
#421	#829	#829	#829	#829	#563
#829	#421	#421	#421	#421	#551
#350	#350	#429	#429	#350	#357
#429	#429	#350	#350	#429	#805

6. CONCLUSIONS

In this article, new vector similarity measures based on multiplication-free operators are defined. New operators are low power operators because they use only additions, sign comparisons and minimum and maximum operations. It is experimentally observed that the new measures produce comparable results as the cosine similarity measure in two datasets.

References

[1] Anand Rajaraman and Jeffrey David Ullman, *Mining of Massive Datasets*, Cambridge University Press, New York, NY, USA, 2011.

[2] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze, *Introduction to Information Retrieval*, Cambridge University Press, New York, NY, USA, 2008.

[3] Katie Fehrenbacher, "The world of big data needs low power computing," `http://gigaom.com/2011/03/24/the-world-of-big-data-needs-low-power-computing/`, Accessed: 2011-03-24.

[4] A Suhre, F. Keskin, T. Ersahin, R. Cetin-Atalay, R. Ansari, and AE. Cetin, "A multiplication-free framework for signal processing and applications in biomedical image analysis," in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, May 2013, pp. 1123–1127.

[5] H. Tuna, I Onaran, and AE. Cetin, "Image description using a multiplier-less operator," *Signal Processing Letters, IEEE*, vol. 16, no. 9, pp. 751–753, Sept 2009.

[6] Onur Yorulmaz, Tom C. Pearson, and A.Enis Cetin, "Detection of fungal damaged popcorn using image property covariance features," *Computers and Electronics in Agriculture*, vol. 84, no. 0, pp. 47 – 52, 2012.

[7] Kaan Duman and A. Enis Cetin, "Target detection in sar images using codifference and directional filters," 2010.

[8] Yusuf Hakan Habiboglu, Osman Gunay, and A.Enis Cetin, "Covariance matrix-based fire and flame detection method in video," *Machine Vision and Applications*, vol. 23, no. 6, pp. 1103–1113, 2012.

[9] J. Kleinberg and E. Tardos, "Approximation algorithms for classification problems with pairwise relationships: metric labeling and markov random fields," in *Foundations of Computer Science, 1999. 40th Annual Symposium on*, 1999, pp. 14–23.

[10] Moses S. Charikar, "Similarity estimation techniques from rounding algorithms," in *Proceedings of the Thiry-fourth An-*

nual ACM Symposium on Theory of Computing, New York, NY, USA, 2002, STOC '02, pp. 380–388, ACM.

[11] R. Da Silva Villaca, L. Bernardes de Paula, R. Pasquini, and M. Ferreira Magalhaes, "A similarity search system based on the hamming distance of social profiles," in *Semantic Computing (ICSC), 2013 IEEE Seventh International Conference on*, Sept 2013, pp. 90–93.

[12] Wagner P. K. Peres S. M. Madeo, R. C. B., "UCI machine learning repository," 2014.

[13] Juan Ramos, "Using tf-idf to determine word relevance in document queries," 1999.