Linear-Complexity Data-Parallel Earth Mover's Distance Approximations

Kubilay Atasu [1] **Thomas Mittelholzer** [2]

Abstract

The Earth Mover's Distance (EMD) is a state-of-the art metric for comparing discrete probability distributions, but its high distinguishability comes at a high cost in computational complexity. Even though linear-complexity approximation algorithms have been proposed to improve its scalability, these algorithms are either limited to vector spaces with only a few dimensions or they become ineffective when the degree of overlap between the probability distributions is high. We propose novel approximation algorithms that overcome both of these limitations, yet still achieve linear time complexity. All our algorithms are data parallel, and therefore, we can take advantage of massively parallel computing engines, such as Graphics Processing Units (GPUs). On the popular text-based 20 Newsgroups dataset, the new algorithms are four orders of magnitude faster than a multi-threaded CPU implementation of Word Mover's Distance and match its search accuracy. On MNIST images, the new algorithms are four orders of magnitude faster than Cuturi's GPU implementation of the Sinkhorn's algorithm while offering a slightly higher search accuracy.

1. Introduction

Earth Movers Distance (EMD) was initially proposed in the image retrieval field to quantify the similarity between images (Rubner et al., 1998). In the optimization theory, a more general formulation of EMD, called Wasserstein distance, has been used extensively to measure the distance between probability distributions (Villani, 2003). In statistics, an equivalent measure is known as Mallows distance (Levina & Bickel, 2001). This paper uses the EMD measure for similary search in image and text databases.

[1]IBM Research - Zurich [2]HSR Hochschule für Technik, Rapperswil. Correspondence to: Kubilay Atasu <kat@zurich.ibm.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

In the text retrieval domain, an adaptation of EMD, called Word Movers Distance (WMD), has emerged as a state-of-the-art semantic similarity metric (Kusner et al., 2015). WMD captures semantic similarity by using the concept of word embeddings in the computation of EMD. Word embeddings map words into a high-dimensional vector space such that the words that are semantically similar are close to each other. These vectors can be pre-trained in an unsupervised way, e.g., by running the Word2vec algorithm (Mikolov et al., 2013) on publicly available data sets. The net effect is that, given two sentences that cover the same topic, but have no words in common, traditional methods, such as cosine similarity, fail to detect the similarity. However, WMD detects and quantifies the similarity by taking the proximity between different words into account.

What makes EMD-based approaches attractive is their high search and classification accuracy. However, such an accuracy does not come for free. In general, the time complexity of computing these measures grows cubically in the size of the input probability distributions. Such a high complexity renders their use impractical for large datasets. Thus, there is a need for low-complexity approximation methods.

EMD can be computed in quadratic time complexity when an L_1 ground distance is used (Ling & Okada, 2007; Gudmundsson et al., 2007). In addition, approximations of EMD can be computed in linear time by embedding EMD into Euclidean space (Indyk & Thaper, 2003). However, such embeddings result in high distortions in high-dimensional spaces (Naor & Schechtman, 2006). An algorithm for computing EMD in the wavelet domain has also been proposed (Shirdhonkar & Jacobs, 2008), which achieves linear time complexity in the size of the input distributions. However, the complexity grows exponentially in the dimensionality of the underlying vector space. Thus, both linear-complexity approaches are impractical when the number of dimensions is more than three or four. For instance, they are not applicable to WMD because the word vectors typically have several hundred dimensions.

A linear-complexity algorithm for computing approximate EMD distances over high-dimensional vector spaces has also been proposed (Atasu et al., 2017). The algorithm, called Linear-Complexity Relaxed Word Mover's Distance (LC-RWMD), achieves four orders of magnitude improve-

ment in speed with respect to WMD. In addition, on compact and curated text documents, it computes high-quality search results that are comparable to those found by WMD. Despite its scalability, the limitations of LC-RWMD are not well understood. Our analysis shows that 1) it is not applicable to dense histograms, and 2) its accuracy decreases when comparing probability distributions with many overlapping coordinates. Our main contributions are as follows:

- We propose new distance measures that are more robust and provably more accurate than LC-RWMD.
- We show that the new measures effectively quantify the similarity between dense as well as overlapping probability distributions, e.g., greyscale images.
- We show that the new measures can be computed in linear time complexity in the size of the input probability distributions: the same as for LC-RWMD.
- We propose data-parallel algorithms that achieve four orders of magnitude speed-up with respect to state-of-the-art solutions without giving up any accuracy.

2. Background

EMD can be considered as the discrete version of the Wasserstein distance, and can be used to quantify the affinity between discrete probability distributions. Each probability distribution is modelled as a histogram, wherein each bin is associated with a weight and a coordinate in a multi-dimensional vector space. For instance, when measuring the distance between greyscale images, the histogram weights are given by the pixel values and the coordinates are defined by the respective pixel positions (see Fig. 1 (a)).

The distance between two histograms is calculated as the cost of transforming one into the other. Transforming a first histogram into a second one involves moving weights from the bins of the first histogram into the bins of the second, thereby constructing the second histogram from the first. The goal is to minimize the total distance travelled, wherein the pairwise distances between different histogram bins are computed based on their respective coordinates. This optimization problem is well studied in transportation theory and is the discrete formulation of the Wasserstein distance.

Assume that histograms $\mathbf{p}$ and $\mathbf{q}$ are being compared, where $\mathbf{p}$ has h_p entries and $\mathbf{q}$ has h_q entries. Assume also that an $h_p \times h_q$ nonnegative cost matrix $\mathbf{C}$ is available. Note that p_i indicates the weight stored in the ith bin of histogram $\mathbf{p}$, q_j the weight stored in the jth bin of histogram $\mathbf{q}$, and $C_{i,j}$ the distance between the coordinates of the ith bin of $\mathbf{p}$ and the jth bin of $\mathbf{q}$ (see Fig. 1 (b)). Suppose that the histograms are L_1-normalized: $\sum_i p_i = \sum_j q_j = 1$.

We would like to discover a non-negative flow matrix $\mathbf{F}$, where $F_{i,j}$ indicates how much of the bin i of $\mathbf{p}$ has to *flow* to the bin j of $\mathbf{q}$, such that the cost of moving $\mathbf{p}$ into $\mathbf{q}$ is minimized. Formally, the objective of EMD is as follows:

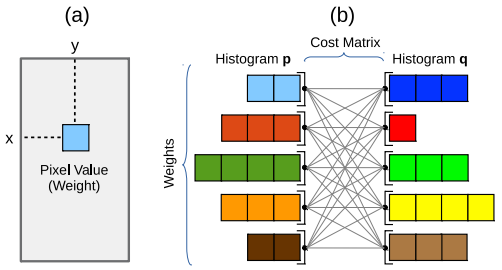

Figure 1. (a) Converting a 28x28 image into a histogram with h=28x28=784 bins. The weights are the pixel values and the embedding vectors are the pixel coordinates. (b) Computing the EMD between two flattened histograms for an hxh cost matrix $\mathbf{C}$.

$$\text{EMD}(\mathbf{p}, \mathbf{q}) = \min_{F_{i,j} \geq 0} \sum_{i,j} F_{i,j} \cdot C_{i,j}. \tag{1}$$

A valid solution to EMD has to satisfy the so-called *out-flow* (2) and *in-flow* (3) constraints. The *out-flow* constraints ensure that, for each i of $\mathbf{p}$, the sum of all the flows exiting i is equal to p_i. The *in-flow* constraints ensure that, for each j of $\mathbf{q}$, the sum of all the flows entering j is equal to q_j. These constraints guarantee that all the mass stored in $\mathbf{p}$ is transferred and $\mathbf{q}$ is reconstructed as a result.

$$\sum_j F_{i,j} = p_i \tag{2}$$

$$\sum_i F_{i,j} = q_j \tag{3}$$

Computation of EMD requires solution of a minimum-cost-flow problem on a bi-partite graph, wherein the bins of histogram $\mathbf{p}$ are the source nodes, the bins of histogram $\mathbf{q}$ are the sink nodes, and the edges between the source and sink nodes indicate the pairwise transportation costs. Solving this problem optimally takes supercubical time complexity in the size of the input histograms (Ahuja et al., 1993).

2.1. Relaxed Word Mover's Distance

To reduce the complexity, an approximation algorithm called Relaxed Word Mover's Distance (RWMD) was proposed (Kusner et al., 2015). RWMD computation involves derivation of two asymmetric distances. First, the *in-flow* constraints are relaxed and the relaxed problem is solved using only *out-flow* constraints. The solution to the first relaxed problem is a lower bound of EMD. After that, the *out-flow* constraints are relaxed and a second relaxed optimization problem is solved using only *in-flow* constraints,

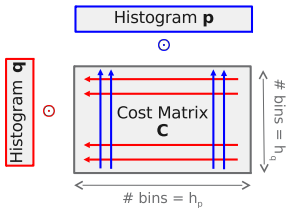

Figure 2. Quadratic-complexity RWMD computation

which computes a second lower bound of EMD. RWMD is the maximum of these two lower bounds. Therefore, it is at least as tight as each one. In addition, it is symmetric.

Finding an optimal solution to RWMD involves mapping the coordinates of one histogram to the closest coordinates of the other. Just like EMD, RWMD requires a cost matrix $\mathbf{C}$ that stores the pairwise distances between coordinates of $\mathbf{p}$ and $\mathbf{q}$. Finding the closest coordinates corresponds to row-wise and column-wise minimum operations in the cost matrix (see Fig. 2). To compute the first lower bound, it is sufficient to find the column-wise minimums in the cost matrix, and then perform a dot product with the weights stored in $\mathbf{p}$. Similarly, to compute the second lower bound, it is sufficient to find the row-wise minimums and then perform a dot product with the weights stored in $\mathbf{q}$. The complexity of RWMD is given by the cost of constructing the cost matrix $\mathbf{C}$: it requires quadratic time and space in the size of the input histograms. Computing the row-wise and column-wise minimums of $\mathbf{C}$ also has quadratic time complexity.

2.2. Linear-Complexity RWMD (LC-RWMD)

When computing RWMD between only two histograms, it is not possible to avoid a quadratic time complexity. However, in a typical information retrieval system, a query histogram is compared with a large database of histograms to identify the top-ℓ most similar histograms in the database. It is shown in (Atasu et al., 2017) that the RWMD computation involves redundant and repetitive operations in such cases, and that eliminating this redundancy reduces the average time complexity from quadratic to linear.

Assume that a query histogram is being compared with two database histograms. Assume also that the two database histograms have common coordinates. A simple replication of the RWMD computation would involve creation of two cost matrices with identical rows for the common coordinates. Afterwards, it would be necessary to perform reduction operations on these identical rows to compute the row-wise minimums. It is shown in (Atasu et al., 2017) that both of these redundant operations can be eliminated by 1) constructing a vocabulary that stores the union of the coordinates that occur in the database histograms, and 2) computing the minimum distances between the coordinates of the vocabulary and the coordinates of the query only once.

Table 1. Algorithmic Parameters

n	Number of database histograms
v	Size of the vocabulary
m	Dimensionality of the vectors
h	Average histogram size

Table 2. Complexity of Computing n RWMD Distances

	Time Complexity	Space Complexity
LC-RWMD	$O(vhm + nh)$	$O(nh + vm + vh)$
RWMD	$O(nh^2m)$	$O(nhm)$

Table 1 lists the algorithmic parameters that influence the complexity. Table 2 shows the complexity of RWMD and LC-RWMD algorithms when comparing one query histogram with n database histograms. When the number of database histograms (n) is in the order of the size of the vocabulary (v), the LC-RWMD algorithm reduces the complexity by a factor of the average histogram size (h). Therefore, whereas the time complexity of a brute-force RWMD implementation scales quadratically in the histogram size, the time complexity of LC-RWMD scales only linearly.

3. Related Work

A regularized version of the optimal transport problem can be solved more efficiently than network-flow-based approaches (Cuturi, 2013). The solution algorithm is based on Sinkhorn's matrix scaling technique (Sinkhorn, 1964), and thus, it is called Sinkhorn's algorithm. When computing the distance between two histograms of size $O(h)$, the algorithm can require up to $O(h)$ iterations for convergence in the worst-case, and the complexity of each iteration is $O(h^2)$. Therefore, its worst-case time complexity is $O(h^3)$. However, a cubical time complexity has not been observed empirically. In addition, convolutional implementations of Sinkhorn's algorithm can be used to reduce the time complexity (Solomon et al., 2015). More recently, a greedy coordinate-descent-based approach has been proposed, which reduces the time complexity to $O(h^2 \log h)$ (Altschuler et al., 2017). In each case, a cost matrix is constructed, which incurs an additional complexity of $O(h^2m)$ when using m-dimensional coordinates.

Several other lower bounds of EMD have been proposed (Assent et al., 2008; Xu et al., 2010; Ruttenberg & Singh, 2011; Wichterich et al., 2008; Xu et al., 2016; Huang et al., 2016; 2014). These lower bounds are typically used to speed-up the EMD computation based on pruning techniques. Alternatively, EMD can be computed approximately using a compressed representation (Uysal et al., 2016; Pele & Werman, 2009).

A greedy network-flow-based approximation algorithm has also been proposed (Gottschlich & Schuhmacher, 2014), which does not relax the *in-flow* or *out-flow* constraints.

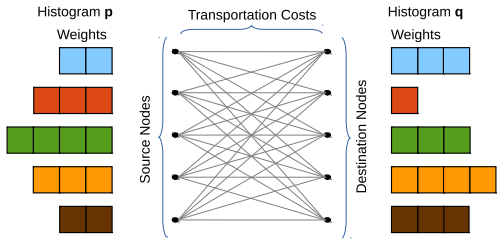

Figure 3. Different histograms with identical coordinates

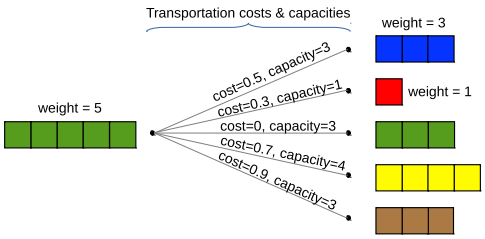

Figure 4. Imposing capacity constraints on the edges

Therefore, it is not a data-parallel algorithm and its complexity is quadratic in the histogram size. In addition, it produces an upper bound rather than a lower bound of EMD.

4. New Relaxation Algorithms

In this section, we describe improved relaxation algorithms that address the weaknesses of the RWMD measure and its linear-complexity implementation (LC-RWMD). Assume that we are measuring the distance between two histograms $\mathbf{p}$ and $\mathbf{q}$. Assume also that the coordinates of the two histograms fully overlap but the respective weights are different (see Fig. 3). In other words, for each coordinate i of $\mathbf{p}$, there is an identical coordinate j of q, for which $\mathbf{C}_{i,j} = 0$. Therefore, RWMD estimates the total cost of moving $\mathbf{p}$ into $\mathbf{q}$ and vice versa as zero even though p and q are not the same. This condition arises, for instance, when we are dealing with dense histograms. In other cases, the data of interest might actually be sparse, but some background noise might also be present, which results in denser histograms.

In general, the more overlaps there are between the coordinates of $\mathbf{p}$ and $\mathbf{q}$, the higher the approximation error of RWMD is. The main reason for the error is that when coordinate i of $\mathbf{p}$ overlaps with coordinate j of $\mathbf{q}$, RWMD does not take into account the fact that the respective weights p_i and q_j can be different. In an optimal solution, we would not be moving a mass larger than the minimum of p_i and q_j between these two coordinates. This is a fundamental insight that we use in the improved solutions we propose.

Given $\mathbf{p}$, $\mathbf{q}$ and $\mathbf{C}$, our goal is to define new distance measures that relax fewer EMD constraints than RWMD, and therefore, produce tighter lower bounds on EMD. Two asymmetric distances can be computed by deriving 1) the cost of moving $\mathbf{p}$ into $\mathbf{q}$ and 2) the cost of moving $\mathbf{q}$ into $\mathbf{p}$. If both are lower bounds on EMD$(\mathbf{p}, \mathbf{q})$, a symmetric lower bound can be derived, e.g., by using the maximum of the two. Thus, we consider only the computation of the cost of moving $\mathbf{p}$ into $\mathbf{q}$ without loss of essential generality.

When computing the cost of moving $\mathbf{p}$ to $\mathbf{q}$ using RWMD, the *in-flow* constraints of (3) are removed. In other words, all the mass is transferred from $\mathbf{p}$ to the coordinates of $\mathbf{q}$, but the resulting distribution is not the same as $\mathbf{q}$. Therefore, the cost of transforming $\mathbf{p}$ to $\mathbf{q}$ is underestimated by RWMD. To achieve better approximations of EMD$(\mathbf{p}, \mathbf{q})$, instead of removing the *in-flow* constraints completely, we propose the use of a relaxed version of these constraints:

$$F_{i,j} \leq q_j \quad \text{for all } i, j. \tag{4}$$

The new constraint ensures that the amount of weight that can be moved from a coordinate i of $\mathbf{p}$ to a coordinate j of $\mathbf{q}$ cannot exceed the weight q_j at coordinate j. However, even if (4) is satisfied, the total weight moved to coordinate j of $\mathbf{q}$ from all the coordinates of $\mathbf{p}$ can exceed q_j, potentially violating (3). Namely, (3) implies (4), but not vice versa.

When (4) is used in combination with (2), we have:

$$F_{i,j} \leq \min\left(p_i, q_j\right) \quad \text{for all } i, j. \tag{5}$$

Note that we are essentially imposing capacity constraints on the edges of the flow network (see Fig.4) based on (5).

We would like to stress that in the framework of this work, which considers the discrete and not the continuous case of Wasserstein distances, the only requirement on the cost matrix is that it is nonnegative. Since any nonnegative cost c between two locations can be written as the p-th power of the p-th root of c for $p \geq 1$, one can assume that we are dealing with a p-th Wasserstein distance (Villani, 2008).

In the following subsections, we describe three new approximation methods. The Overlapping Mass Reduction (OMR) method imposes the relaxed constraint (4) only between overlapping coordinates, and is the lowest-complexity and the least accurate approximation method. The Iterative Constrained Transfers (ICT) method imposes constraint (4) between all coordinates of $\mathbf{p}$ and $\mathbf{q}$, and is the most complex and most accurate approximation method. The Approximate Iterative Constrained Transfers (ACT) method imposes constraint (4) incrementally between coordinates of $\mathbf{p}$ and $\mathbf{q}$, and is an approximation of the ICT method. Therefore, both its complexity and its accuracy are higher than those of OMR, but lower than those of ICT.

4.1. Overlapping Mass Reduction

The OMR method imposes (4) only between overlapping coordinates. The main intuition behind OMR method is that if the coordinate i of $\mathbf{p}$ and the coordinate j of $\mathbf{q}$ overlap (i.e., $C_{i,j} = 0$), a transfer of $\min(p_i, q_j)$ can take place

free of cost between $\mathbf{p}$ of $\mathbf{q}$. After that, the remaining weight in p_i is transferred simply to the second closest coordinate in $\mathbf{q}$ as this is the next least costly move. Therefore, the method computes only the top-2 smallest values in each row of $\mathbf{C}$. A detailed description is given in Algorithm 1.

Algorithm 1 Optimal Computation of OMR

function OMR($\mathbf{p}, \mathbf{q}, \mathbf{C}$)
  $t = 0$ ▷ initialize transportation cost t
  for $i = 1 \ldots, h_p$ **do** ▷ iterate the indices of $\mathbf{p}$
    $\mathbf{s} = \text{argmin}_2(C_{i,[1\ldots h_q]})$ ▷ find top-2 smallest
    if $C_{i,\mathbf{s}[1]} == 0$ **then** ▷ if the smallest value is 0
      $r = \min(p_i, q_{\mathbf{s}[1]})$ ▷ size of max. transfer
      $p_i = p_i - r$ ▷ move r units of p_i to $q_{\mathbf{s}[1]}$
      $t = t + p_i \cdot C_{i,\mathbf{s}[2]}$ ▷ move the rest to $q_{\mathbf{s}[2]}$
    else
      $t = t + p_i \cdot C_{i,\mathbf{s}[1]}$ ▷ move all of p_i to $q_{\mathbf{s}[1]}$
    end if
  end for
  return t ▷ return transportation cost t
end function

4.2. Iterative Constrained Transfers

The ICT method imposes the constraint (4) between all coordinates of $\mathbf{p}$ and $\mathbf{q}$. The main intuition behind the ICT method is that because the inflow constraint (3) is relaxed, the optimal flow exiting each source node can be determined independently. For each source node, finding the optimal flow involves sorting the destination nodes in the ascending order of transportation costs, and then performing iterative mass transfers between the source node and the sorted destination nodes under the capacity constraints (4). Algorithm 2 describes the ICT method in full detail.

Algorithm 2 Optimal Computation of ICT

function ICT($\mathbf{p}, \mathbf{q}, \mathbf{C}$)
  $t = 0$ ▷ initialize transportation cost t
  for $i = 1 \ldots, h_p$ **do** ▷ iterate the indices of $\mathbf{p}$
    $\mathbf{s} = \text{argsort}(C_{i,[1\ldots h_q]})$ ▷ sort indices by value
    $l = 1$ ▷ initialize l
    while $p_i > 0$ **do** ▷ while there is mass in p_i
      $r = \min(p_i, q_{\mathbf{s}[l]})$ ▷ size of max. transfer
      $p_i = p_i - r$ ▷ move r units of p_i to $q_{\mathbf{s}[l]}$
      $t = t + r \cdot C_{i,\mathbf{s}[l]}$ ▷ update cost
      $l = l + 1$ ▷ increment l
    end while
  end for
  return t ▷ return transportation cost t
end function

Algorithm 3 describes an approximate solution to ICT (ACT), which offers the possibility to terminate the ICT iterations before all the mass is transferred from $\mathbf{p}$ to $\mathbf{q}$. After performing a predefined number $k-1$ of ICT iterations, the mass remaining in $\mathbf{p}$ is transferred to the k-th closest coordinates of $\mathbf{q}$, making the solution approximate.

Theorem 1 establishes the optimality of Algorithm 2. Theorem 2 establishes the relationship between different distance measures. The proofs and the derivation of the complexity of the algorithms are omitted for brevity.

Theorem 1. *(i) The flow F^* of Algorithm 2 is an optimal solution of the relaxed minimization problem given by (1), (2) and (4). (ii) ICT provides a lower bound on EMD.*

Theorem 2. *For two normalized histograms $\mathbf{p}$ and $\mathbf{q}$: RWMD$(\mathbf{p}, \mathbf{q}) \leq$ OMR$(\mathbf{p}, \mathbf{q}) \leq$ ACT$(\mathbf{p}, \mathbf{q}) \leq$ ICT$(\mathbf{p}, \mathbf{q}) \leq$ EMD$(\mathbf{p}, \mathbf{q})$.*

Algorithm 3 Approximate Computation of ICT

function ACT($\mathbf{p}, \mathbf{q}, \mathbf{C}, k$)
  $t = 0$ ▷ initialize transportation cost t
  for $i = 1 \ldots, h_p$ **do** ▷ iterate the indices of $\mathbf{p}$
    $\mathbf{s} = \text{argmin}_k(C_{i,[1\ldots h_q]})$ ▷ find top-k smallest
    $l = 1$ ▷ initialize l
    while $l < k$ **do**
      $r = \min(p_i, q_{\mathbf{s}[l]})$ ▷ size of max. transfer
      $p_i = p_i - r$ ▷ move r units of p_i to $q_{\mathbf{s}[l]}$
      $t = t + r \cdot C_{i,j}$ ▷ update cost
      $l = l + 1$ ▷ increment l
    end while
    if $p_i \neq 0$ **then** ▷ if p_i still has some mass
      $t = t + p_i \cdot C_{i,\mathbf{s}[k]}$ ▷ move the rest to $q_{\mathbf{s}[k]}$
    end if
  end for
  return t ▷ return transportation cost t
end function

5. Linear-Complexity Implementations

In this section, we focus on the ACT method because 1) it is a generalization of all the other methods presented, and 2) its complexity and accuracy can be controlled by setting the number k of iterations performed. We describe a data-parallel implementation of ACT, which achieves a linear time complexity when k is a constant. Unlike the previous section, we do not assume that the cost matrix is given. We compute the transportation costs on the fly, and take into account the complexity of computing these costs as well.

A high-level view of the linear-complexity ACT algorithm (LC-ACT) is given in Figure 5. LC-ACT is strongly inspired by LC-RWMD. Just like LC-RWMD, it assumes that 1) a query histogram is compared with a large number of database histograms, and 2) the coordinate space is populated by the members of a fixed-size vocabulary. The complexity is reduced by eliminating the redundant and repetitive operations that arise when comparing one query histogram with a large number of database histograms.

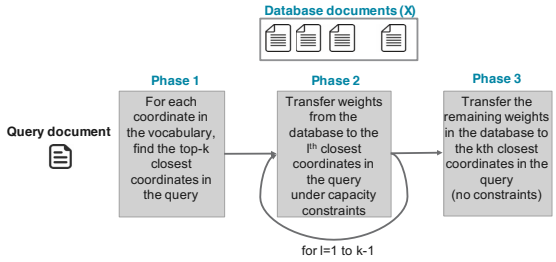

Figure 5. Linear Complexity ACT

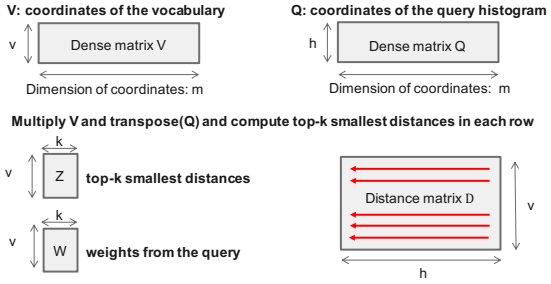

Figure 6. Phase 1 of LC-ACT

Suppose that the dimension of the coordinates is m and the size of the vocabulary is v. Let $\mathbf{V}$ be an $v \times m$ matrix that stores this information. Given a query histogram $\mathbf{q}$ of size h, we construct a matrix $\mathbf{Q}$ of size $h \times m$ that stores the coordinates of the histogram entries. Phase 1 of LC-ACT (see Fig. 6) performs a matrix-matrix multiplication between $\mathbf{V}$ and the transpose of $\mathbf{Q}$ to compute all pairwise distances between the coordinates of the vocabulary and the coordinates of the query. The result is a $v \times h$ distance matrix, denoted by $\mathbf{D}$. As a next step, the top-k smallest distances are computed in each row of $\mathbf{D}$. The result is stored in a $v \times k$ matrix $\mathbf{Z}$. Furthermore, we store the indices of $\mathbf{q}$ that are associated with the top-k smallest distances in a $v \times k$ matrix $\mathbf{S}$. We can then construct another $v \times k$ matrix $\mathbf{W}$, which stores the corresponding weights of $\mathbf{q}$ by defining $\mathbf{W}_{i,l} = \mathbf{q}_{S_{i,l}}$ for $i = 1, \ldots, v$ and $l = 1, \ldots, k$. The matrices $\mathbf{Z}$ and $\mathbf{W}$ are then used in Phase 2 to transport the largest possible mass, which are constrained by $\mathbf{W}$, to the smallest possible distances, which are given by $\mathbf{Z}$.

The database histograms are stored in a matrix $\mathbf{X}$ (see Fig. 7), wherein each row stores one histogram. These histograms are typically sparse. Thus, the matrix $\mathbf{X}$ is stored using a sparse representation, e.g., in compressed sparse rows (csr) format. For simplicity, assume that $\mathbf{X}$ is stored in a dense format and $\mathbf{X}_{u,i}$ stores the weight of the i-th coordinate of the vocabulary in the u-th database histogram. Note that if the histograms have h entries on average, the number of nonzeros of the matrix $\mathbf{X}$ would be equal to nh.

Phase 2 of ACT iterates the columns of $\mathbf{Z}$ and $\mathbf{W}$ and iteratively transfers weights from the database histograms $\mathbf{X}$ to the query histogram q. Let $\mathbf{X}^{(l)}$ represent the residual mass remaining in $\mathbf{X}$ after l iterations, where $\mathbf{X}^{(0)} = \mathbf{X}$. Let $\mathbf{Y}^{(l)}$ store the amount of mass that is transferred from $\mathbf{X}^{(l-1)}$ in iteration l, which is the difference between $\mathbf{X}^{(l-1)}$ and $\mathbf{X}^{(l)}$. Let $\mathbf{z}^{(l)}$ and $\mathbf{w}^{(l)}$ be the l-th columns of $\mathbf{Z}$ and $\mathbf{W}$, respectively; thus, $\mathbf{z}_u^{(l)}$ is the l-th smallest distance between the coordinate u of the vocabulary and the coordinates of the query, and $\mathbf{w}_u^{(l)}$ is the respective weight of the query coordinate that produces the l-th smallest distance. The iteration l of Phase 2 computes $\mathbf{Y}^{(l)}$ and $\mathbf{X}^{(l)}$:

$$\mathbf{Y}_{u,i}^{(l)} = \min_{\substack{u \in \{1 \ldots v\} \\ i \in \{1 \ldots v\}}} (\mathbf{X}_{u,i}^{(l-1)}, \mathbf{w}_u^{(l)}). \tag{6}$$

$$\mathbf{X}^{(l)} = \mathbf{X}^{(l-1)} - \mathbf{Y}^{(l)}. \tag{7}$$

The cost of transporting $\mathbf{Y}^{(l)}$ to q is given by $\mathbf{Y}^{(l)} \cdot \mathbf{z}^{(l)}$. Let $\mathbf{t}^{(l)}$ be a vector of size n that accumulates all the transportation costs incurred between iteration 1 and iteration l:

$$\mathbf{t}^{(l)} = \mathbf{t}^{(l-1)} + \mathbf{Y}^{(l)} \cdot \mathbf{z}^{(l)}. \tag{8}$$

After $k-1$ iterations of Phase 2, there might still be some mass remaining in $\mathbf{X}^{(l-1)}$. Phase 3 approximates the cost of transporting the remaining mass to q by multiplying $\mathbf{X}^{(l-1)}$ with $\mathbf{z}^{(k)}$. The overall transportation cost $\mathbf{t}^{(k)}$ is:

$$\mathbf{t}^{(k)} = \mathbf{t}^{(k-1)} + \mathbf{X}^{(l-1)} \cdot \mathbf{z}^{(k)}. \tag{9}$$

The main building blocks of LC-ACT are matrix-matrix and matrix-vector multiplications, row-wise top-k calculations, and parallel element-wise updates, all of which are data-parallel operations. Table 3 shows the complexity of computing LC-ACT between one query histogram and n database histograms. Note that when k is a constant, LC-ACT and LC-RWMD methods have the same complexity.

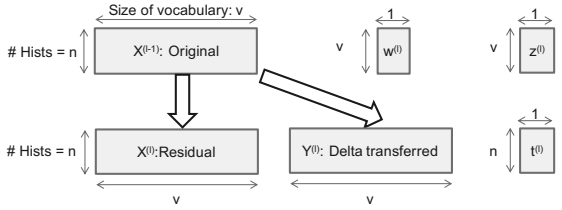

Figure 7. Phase 2 of LC-ACT

Table 3. Complexity of LC-ACT (n distances, k iterations)

Time	Space
$O(vhm + nhk)$	$O(nh + vm + vh + vk)$

6. Evaluation

We performed experiments on two public datasets: *20 Newsgroups* is a text database of newsgroup documents, partitioned evenly across 20 different classes[1], and *MNIST* is an image database of greyscale hand-written digits that are partitioned evenly across 10 classes[2]. The MNIST

[1] http://qwone.com/ jason/20Newsgroups/

[2] http://yann.lecun.com/exdb/mnist/

Table 4. Dataset properties: no. docs (n), average size of histograms (h), original vocabulary size (v), size of used vocabulary

	Size n	Average h	Original v	Used v
20 News	18828	78.8	3M	69682
MNIST	60000	149.9	784	717

images are mapped to histograms as illustrated in Fig. 1, wherein the weights are normalized pixel values and the embedding vectors indicate the coordinates of the pixels. The words in 20 Newsgroups documents are mapped to a 300-dimensional real-valued embedding space using Word2Vec vectors that are pre-trained on Google News[3], for which the size of the vocabulary (v) is 3M words and phrases. In our setup, each histogram bin is associated with a word from this vocabulary. The first 100 words of the vocabulary are treated as stop words. These stop words and the Word2Vec phrases are not mapped to histogram bins. The histogram weights indicate normalized frequencies of words found in each document. In addition, 20 Newsgroups histograms are truncated to store only the most-frequent 500 words found in each document. Table 4 shows some properties of the datasets and the results of our preprocessing. Note that the size of the vocabulary used has an impact on the complexity of our methods (see Tab. 3).

The results we provide in the remainder of the text are associated with linear complexity implementations of RWMD, OMR, and ACT methods. To improve the robustness of these methods, we compute two asymmetric lower bounds and take the maximum of the two as discussed in Section 2.1. Word2Vec embedding vectors are L_2-normalized, but MNIST embedding vectors are not normalized in our setup. In addition, when computing our approximations, the histogram weights are always L_1-normalized. Lastly, the transportation cost between two words or pixels is the Euclidean (L_2) distance between their embedding vectors.

In our experiments, we treated each document of the database as a query and compared it with every other document in the database. Based on the distance measure used in the comparison, for each query document, we identified the top-ℓ nearest neighbors in the database. After that, for each query document, we computed the percentage of documents in its nearest-neighbors list that have the same label. We averaged this metric over all the query documents and computed it as a function of ℓ. The result is the average *precision @ top-*ℓ for the query documents, and indicates the expected accuracy of nearest neighbors search queries.

We compared our new distance measures with simple baselines, such as, Bag-of-Words (BoW) cosine similarity and the Word Centroid Distance (WCD) (Kusner et al., 2015) measures, both of which exhibit a lower algorithmic complexity than our methods. The BoW approach does not use the proximity information provided by the embedding vectors. It simply computes a dot product between two sparse histograms after an L_2 normalization of the weights. The complexity of computing BoW cosine similarity between one query histogram and n database histograms is $O(nh)$. The WCD measure, on the other hand, is closely related to document embedding techniques. For each document, it first computes a centroid vector of size m, which is a weighted average of the embedding vectors, and then determines the Euclidean distances between these centroid vectors. The complexity of computing WCD between one query document and n database documents is $O(nm)$.

We compared our distance measures with state-of-the-art EMD approximations as well, such as WMD and Sinkhorn distance. WMD uses the FastEMD library[4] to approximate the EMD, which uses a thresholding technique (Pele & Werman, 2009) to reduce the time to compute EMD. In addition, WMD uses an RWMD-based pruning technique to reduce the number of calls to FastEMD (Kusner et al., 2015). To improve the WMD performance further, we developed a multi-threaded CPU implementation of the pruning technique. We also compared our methods with Cuturi's open-source implementation of Sinkhorn's algorithm[5], which can be executed on both CPUs and GPUs. We used $\lambda = 20$ as the entropic regularization parameter because it offered a good trade-off between the accuracy and the speed of Sinkhorn's algorithm.

We have developed GPU-accelerated implementations of the WCD, RWMD, OMR, ACT, and BoW cosine similarity methods and evaluated their performance on an NVIDIA® GTX 1080Ti GPU. Cuturi's Sinkhorn implementation has also been executed on the same GPU. Our multithreaded WMD implementation has been deployed on an 8-core Intel® i7-6900K CPU. Top-ℓ calculations have been performed on the same Intel® i7-6900K CPU in all cases.

Theorem 2 states that the more complex the considered algorithms, the smaller the gap to the EMD and, hence, the better the accuracy. The least complex ACT algorithm is the RWMD, which corresponds to the ACT-0 with zero iterations in Phase 2 (see Fig. 5). The second most complex algorithm is the OMR. The third most complex ACT algorithm is ACT-1 with a single iteration in Phase 2. The most complex ACT algorithm we considered was ACT-15 with 15 iterations in Phase 2. Our experiments show that the search accuracy improves with the complexity and, thus, illustrate the accuracy vs complexity trade-off. Typically, most of the improvement in the search accuracy is achieved by the first iteration of Phase 2, and subsequent iterations result in a limited improvement only. As a result, ACT-1 offers very favorable accuracy and runtime combinations.

[3] https://code.google.com/archive/p/word2vec/

[4] https://github.com/LeeKamentsky/pyemd

[5] http://marcocuturi.net/SI.html

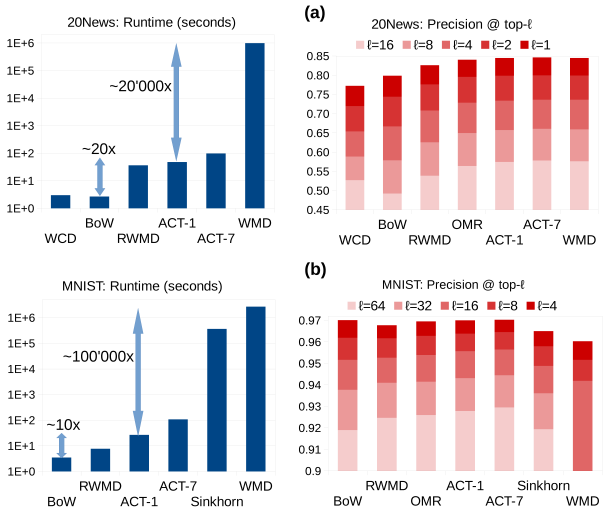

Figure 8. Runtime vs accuracy for 20News and the MNIST subset

Figure 8 (a) shows the accuracy and runtime trade-offs between different methods on the 20 Newsgroups dataset. Note that even though ACT-1 is approximately 20-fold slower than BoW cosine similarity, it offers a 4.5% to 7.5% higher search accuracy. Typically, the accuracy improvement with respect to BoW becomes larger as we increase ℓ. Fig. 8 (a) also shows that ACT-1 is approximately 20000-fold faster than WMD, but offers a similar search accuracy. It is only 30% slower than RWMD, but results in a 2% to 3.5% higher search accuracy. OMR is somewhere between RWMD and ACT-1 in terms of both runtime and search accuracy. Finally, ACT-7 is approximately 10000-fold faster than WMD, and offers a slightly higher search accuracy!

In case of MNIST, because the number of dimensions is small ($m = 2$), RWMD is almost as fast as BoW cosine similarity. However, the runtime of the Phase 2 of the ACT-1 method is much more significant than that of its Phase 1. Therefore, the runtime increase with respect to BoW is around ten fold for ACT-1. Nevertheless, when using ACT-1, computing all pairwise distances between 60000 MNIST training images (i.e., 3.6 billion distance computations) takes only 3.3 minutes. The accuracy comparisons between BoW, RWMD, and ACT methods for the complete MNIST database are given in Tab. 5. The accuracy is already very high when using BoW because the images are normalized and centered. Our methods are comparable to BoW for small ℓ, but outperform it for large enough ℓ.

Computing all pairwise distances for the complete set of MNIST training images would take months when using WMD and Sinkhorn's algorithm. To enable comparisons with these two methods, we have set up a simpler experiment. We used only the first 6000 MNIST training images as our query documents and compared them with the complete set of 60000 MNIST training images. The results are given in Fig. 8 (b). We observe that ACT-1 is four orders of magnitude faster than Sinkhorn's algorithm when running on the same GPU, yet it achieves a higher search accuracy! Similarly, ACT-1 is five orders of magnitude faster than WMD while achieving a higher search accuracy! These results show that the OMR and the ACT measures we proposed are meaningful on their own, and using more complex measures, such as WMD or Sinkhorn does not necessarily improve the accuracy of nearest-neighbors search.

Table 5. Precision @ top-ℓ for MNIST (without background)

ℓ	BoW	RWMD	ACT-1	ACT-3	ACT-7
1	0.9771	0.9752	0.9776	0.9780	0.9781
16	0.9480	0.9481	0.9510	0.9520	0.9521
128	0.8874	0.8963	0.8997	0.9014	0.9016

Table 6. Precision @ top-ℓ for MNIST (with background)

ℓ	BoW	RWMD	OMR	ACT-7	ACT-15
1	0.9771	0.1123	0.9707	0.9756	0.9783
16	0.9480	0.1002	0.9368	0.9470	0.9520
128	0.8874	0.1002	0.8692	0.8872	0.8999

Table 6 illustrates the sensitivity of RWMD to a minor change in the data representation. Here, we simply explore the impact of including the background (i.e. the black pixels) in the MNIST histograms. The most immediate result is that when comparing two histograms, all their coordinates overlap. As a result, the distance computed between the histograms by RWMD is always equal to zero, and the top-ℓ nearest neighbors are randomly selected, resulting in a precision of 10% for RWMD. The OMR technique solves this problem immediately even though its accuracy is lower than that of BoW cosine similarity. In fact, several iterations of ACT are required to outperform BoW. However, these results demonstrate the improved robustness and effectiveness of our methods in comparison to RWMD.

7. Conclusions

This paper offers new theoretical and practical results for improving the efficiency and the accuracy of approximate EMD computation in both high and low dimensions. We identify the shortcomings of the RWMD measure and propose improved lower bounds that result in a higher nearest-neighbors-search accuracy and robustness without increasing the computational complexity. Under realistic assumptions, the complexity of our methods scale linearly in the size of the probability distributions. In addition, our methods are data-parallel and well-suited for GPU acceleration. The experiments demonstrate a four orders of magnitude improvement of the performance without any loss of nearest-neighbors-search accuracy versus WMD on high-dimensional text datasets. Similar accuracy and performance improvements have been achieved with respect to Sinkhorn's algorithm on two-dimensional image datasets.

Acknowledgements

We would like to thank Celestine Dünner, Haralampos Pozidis, Ahmet Solak, and Slavisa Sarafijanovic from IBM Research – Zurich, and the anonymous reviewers of ICML 2019 Conference for their valuable comments on this work.

References

Ahuja, R. K., Magnanti, T. L., and Orlin, J. B. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993. ISBN 0-13-617549-X.

Altschuler, J., Weed, J., and Rigollet, P. Near-linear time approximation algorithms for optimal transport via sinkhorn iteration. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 1964–1974. Curran Associates, Inc., 2017.

Assent, I., Wichterich, M., Meisen, T., and Seidl, T. Efficient similarity search using the Earth Mover's Distance for large multimedia databases. In *2008 IEEE 24th International Conference on Data Engineering (ICDE)*, pp. 307–316, April 2008.

Atasu, K., Parnell, T. P., Dünner, C., Sifalakis, M., Pozidis, H., Vasileiadis, V., Vlachos, M., Berrospi, C., and Labbi, A. Linear-complexity relaxed word mover's distance with GPU acceleration. In *2017 IEEE International Conference on Big Data, BigData 2017, Boston, MA, USA, December 11-14, 2017*, pp. 889–896, 2017.

Cuturi, M. Sinkhorn distances: Lightspeed computation of optimal transport. In Burges, C. J. C., Bottou, L., Welling, M., Ghahramani, Z., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 26*, pp. 2292–2300. Curran Associates, Inc., 2013.

Gottschlich, C. and Schuhmacher, D. The shortlist method for fast computation of the earth mover's distance and finding optimal solutions to transportation problems. *PLOS ONE*, 9(10):1–10, 10 2014.

Gudmundsson, J., Klein, O., Knauer, C., and Smid, M. Small manhattan networks and algorithms for the earth movers distance. In *In Proc. 23rd European Workshop Comput. Geom. (EWCG07)*, pp. 174–177, 2007.

Huang, J., Zhang, R., Buyya, R., and Chen, J. MELODY-JOIN: Efficient Earth Mover's Distance similarity joins using MapReduce. In *2014 IEEE 30th International Conference on Data Engineering (ICDE)*, pp. 808–819, March 2014.

Huang, J., Zhang, R., Buyya, R., Chen, J., and Wu, Y. Heads-Join: Efficient Earth Mover's Distance Similarity Joins on Hadoop. *IEEE Transactions on Parallel and Distributed Systems*, 27(6):1660–1673, 2016.

Indyk, P. and Thaper, N. Fast image retrieval via embeddings. In *3rd International Workshop on Statistical and Computational Theories of Vision*, Nice, France, 2003.

Kusner, M., Sun, Y., Kolkin, N., and Weinberger, K. From word embeddings to document distances. In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 957–966, Lille, France, 07–09 Jul 2015. PMLR.

Levina, E. and Bickel, P. The Earth Mover's distance is the Mallows distance: some insights from statistics. In *Proceedings Eighth IEEE International Conference on Computer Vision. ICCV 2001*, volume 2, pp. 251–256, 2001. ISBN 0-7695-1143-0. doi: 10.1109/ICCV.2001.937632.

Ling, H. and Okada, K. An efficient earth mover's distance algorithm for robust histogram comparison. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29 (5):840–853, May 2007.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. *CoRR*, abs/1301.3781, 2013.

Naor, A. and Schechtman, G. Planar earthmover is not in l1. In *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)(FOCS)*, volume 00, pp. 655–666, 10 2006.

Pele, O. and Werman, M. Fast and robust earth mover's distances. In *2009 IEEE 12th International Conference on Computer Vision*, pp. 460–467, Sept 2009.

Rubner, Y., Tomasi, C., and Guibas, L. J. A metric for distributions with applications to image databases. In *Proceedings of the Sixth International Conference on Computer Vision*, ICCV '98, pp. 59–66, Washington, DC, USA, 1998. IEEE Computer Society. ISBN 81-7319-221-9.

Ruttenberg, B. E. and Singh, A. K. Indexing the earth mover's distance using normal distributions. *Proceedings of the VLDB Endowment*, 5(3):205–216, 2011.

Shirdhonkar, S. and Jacobs, D. W. Approximate earth movers distance in linear time. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–8, June 2008.

Sinkhorn, R. A Relationship Between Arbitrary Positive Matrices and Doubly Stochastic Matrices. *The Annals of Mathematical Statistics*, 35(2):876–879, 1964.

Solomon, J., de Goes, F., Peyré, G., Cuturi, M., Butscher, A., Nguyen, A., Du, T., and Guibas, L. Convolutional wasserstein distances: Efficient optimal transportation on geometric domains. *ACM Trans. Graph.*, 34(4):66:1–66:11, July 2015. ISSN 0730-0301.

Uysal, M. S., Sabinasz, D., and Seidl, T. Approximation-based efficient query processing with the Earth Mover's Distance. In *Proceedings, Part II, of the 21st International Conference on Database Systems for Advanced Applications - Volume 9643*, DASFAA 2016, pp. 165–180, New York, NY, USA, 2016. Springer-Verlag New York, Inc. ISBN 978-3-319-32048-9.

Villani, C. *Topics in Optimal Transportation*, volume 58 of *Graduate Studies in Mathematics*. AMS, 2003 edition, 2003. ISBN 978-0-8218-3312-4.

Villani, C. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer, 2009 edition, September 2008. ISBN 3540710493.

Wichterich, M., Assent, I., Kranen, P., and Seidl, T. Efficient EMD-based similarity search in multimedia databases via flexible dimensionality reduction. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, pp. 199–212, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-102-6.

Xu, J., Zhang, Z., Tung, A. K., and Yu, G. Efficient and effective similarity search over probabilistic data based on earth mover's distance. *Proceedings of the VLDB Endowment*, 3(1-2):758–769, 2010.

Xu, J., Zhang, J., Song, C., Zhang, Q., Lv, P., Li, T., and Chen, N. EMD-DSJoin: Efficient similarity join over probabilistic data streams based on Earth Mover's Distance. In *Web Technologies and Applications*, pp. 42–54. Springer International Publishing, 2016.