FALCON: A Fast Drop-In Replacement of Citation KNN for Multiple Instance Learning

Shuai Yang
North Carolina State University
Raleigh, North Carolina
syang16@ncsu.edu

Xipeng Shen
North Carolina State University
Raleigh, North Carolina
xshen5@ncsu.edu

ABSTRACT

Citation KNN is an important but compute-intensive algorithm for multiple instance learning (MIL). This paper presents FALCON, a fast replacement of Citation KNN. FALCON accelerates Citation KNN by removing unnecessary distance calculations through two novel optimizations, *multi-level triangle inequality-based distance filtering* and *heap optimization*. The careful design allows it to produce the same results as the original Citation KNN does while avoiding 84–99.8% distance calculations. On seven datasets of various sizes and dimensions, FALCON consistently outperforms Citation KNN by one or two orders of magnitude, making it a promising drop-in replacement of Citation KNN for multiple instance learning.

CCS CONCEPTS

• **Information systems** → *Nearest-neighbor search*; *Clustering and classification*;

KEYWORDS

Citation KNN; Triangle Inequality; Multiple-instance Learning

ACM Reference Format:
Shuai Yang and Xipeng Shen. 2018. FALCON: A Fast Drop-In Replacement of Citation KNN for Multiple Instance Learning. In *The 27th ACM International Conference on Information and Knowledge Management (CIKM '18), October 22–26, 2018, Torino, Italy*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3269206.3271787

1 INTRODUCTION

Different from the traditional supervised learning problems where an object is represented by an instance, in multiple-instance learning (MIL), an object is represented by a set of instances which is defined as a bag. Labels are assigned to bags rather than individual instances. Based on a collection of labeled bags, MIL classifier attempts to classify unknown bags. MIL is an important way especially for modeling many complicated learning problems in the real world. For example, a whole image can be represented by a bag while small patches of the image are described as instances; an article could be represented as a bag while paragraphs or sentences are described as instances. In the most common case of MIL binary classification, a bag is labeled positive if there is at least one positive instance, otherwise, the bag is labeled negative.

MIL is originally proposed by Dietterich et al. [7] to solve musky molecule prediction task. Since then, the idea of MIL has rapidly spread out to other fields. Maron and others [16] have applied MIL on image classification for natural scene. An image is divided into subimages (instances) and labeled based on the contents of subimages. For example, the image is labeled as waterfall only if there is at least one subimage is waterfall. If an image is labeled as non-waterfall, none of these subimages contains a waterfall. Yang and others [30] have explored the application of MIL on image retrieval in a similar approach. MIL has also been introduced to document categorization [2, 21], web mining [34], computer aided diagnosis [8, 9, 22], spam detection [10], stock selection [15], remote sensing imagery data mining [25], object tracking [5, 32], human action recognition [1], and so on.

A number of MIL algorithms have been developed throughout the years [2, 4, 15, 19, 27, 31]. For instance, Citation KNN [27] predicts the label of a new bag by examining both its nearest neighbors and citers (explained later); MI-DD tries to find a concept point that is close to at least one instance of every positive bag and no instances from any negative bag and then use distances to the concept point for classification [15]; and MI-SVM [2] extends SVM in the MIL settings to maximize the margin around a hyperplane which separates positive from negative instances/bags. There are many other work on MIL that we cannot include here for sake of space; readers may see a prior reference [18] for a comprehensive survey. Recently, DNN is also introduced into MIL [28, 29].

One of the most important barriers for the use of MIL algorithms is their long running time. As the nature of MIL requires the considerations of the relations among many bags of instances, these algorithms are compute-intensive, taking lots of time to run.

This work aims to address the main barrier for MIL. Particularly, it concentrates on Citation KNN, for three reasons.

(1) First, Citation KNN is one of the most effective algorithms for MIL. Since its introduction [27], Citation KNN has repeatedly shown superior performance over alternative MIL algorithms. For instance, Nguyen and others [18] compared the performance of 21 MIL algorithms on 26 datasets, and Citation KNN consistently remains in the most competitive list in terms of both accuracy and speed. A similar observation has been made in some earlier work [33] as well as in the later many applications of Citation KNN in various fields [14, 24, 26].

(2) Second, Citation KNN is robust and generally applicable. Unlike some other MIL algorithms which are based on certain assumptions on data distributions and other properties of the problem [25], or designed for specific applications [7], Citation KNN

CIKM '18, October 22–26, 2018, Torino, Italy

ACM ISBN 978-1-4503-6014-2/18/10…$15.00
https://doi.org/10.1145/3269206.3271787

is not subject to any of those assumptions and is general-purpose. Experiments have shown its effectiveness on various kinds of data and problems [14, 18, 24, 26, 27].

(3) Third, Citation KNN is easy to use and intuitive to interpret. As a non-parametric method, Citation KNN is easy to deploy and the results are intuitively interpretable. In comparison, DNN-based methods require a long training process on a large amount of well-labeled data, and the results are often difficult to interpret.

For these reasons, despite the proposals of many other methods throughout the years, Citation KNN remains one of the most important and popular MIL algorithms. A dramatic improvement of its efficiency could hence help advance the practical adoptions of MIL in a broader range of machine learning problems.

Citation KNN is based on Hausdorff distance, and is time-consuming due to the many costly high-dimensional distance calculations. Even if we avoid repetitive calculations, the number of distance calculations is still quadratic to the number of instances which could be large in many MIL datasets. Moreover, the distance calculation is usually between high-dimensional vectors. In most MIL applications, an instance is often characterized by a high-dimensional vector of over 100 dimensions [11, 12, 34]. These factors make Citation KNN computationally expensive. Two previous proposals tried to alleviate the issue by replacing Hausdorff distance with other similarity measures. Vatsavai [25] modeled each bag as a Gaussian distribution, thus Hausdorff distance calculation is replaced by measuring difference between probability distributions. Li and others [13] treated each bag as a graph and then replaced Hausdorff distance with graph similarity. These proposals unfortunately are unable to maintain the same results as Citation KNN gives.

This paper tries to solve the problem at a different level, aiming to speed up Citation KNN without altering its results at all. By strictly preserving its semantic, the method ensures that the optimized algorithm can serve as *a safe drop-in replacement* of the popular algorithm in all scenarios.

Specifically, this paper introduces a **fa**st multi-**l**evel optimization algorithm for **c**itati**on** KNN named FALCON, which accelerates Citation KNN by avoiding most distance calculations without changing the final results. FALCON employs two novel optimizations. The first is *multi-level triangle inequality-based distance filtering*. It efficiently maintains a series of lower bounds and upper bounds of distances at both bag and instance levels. Equipped with a sequence of carefully designed computation filters, it avoids unnecessary distance calculation by examining the two kinds of bounds against the filtering conditions. The second is *heap optimization*, which helps avoid even more distance calculations by controlling the examination order of instances. Neither optimization alters the semantics or results of Citation KNN.

Experiments on seven datasets of various sizes and dimensions confirm that the FALCON algorithm produces the same results as the original Citation KNN does. At the same time, FALCON avoids 84–99.8% distance calculations, and consistently outperforms Citation KNN by one or two orders of magnitude in terms of speed, making it a promising drop-in replacement of Citation KNN for MIL studies and applications.

We organize the rest of this paper as following, in section 2 we give a brief introduction about key ideas in Citation KNN. Section 3 describes the optimization approaches in detail. Section 4 presents the experimental results and comparison with Citation KNN. The last section summarizes the work.

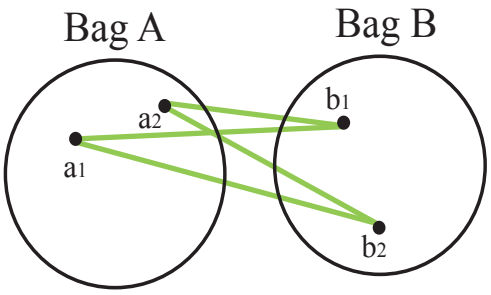

Figure 1: Maximal Hausdorff distance

2 BACKGROUND ON CITATION KNN

The scenarios in MIL problems differ from those in the classic KNN. In classic KNN, each instance corresponds a single point, but in a MIL problem, the distance is measured between bags which each contains a set of instances. Hausdorff distance is used to characterize the distance between two sets of instances.

2.1 Hausdorff Distance

In Citation KNN, two kinds of Hausdorff distance have been used [27]: minimal Hausdorff distance and maximal Hausdorff distance. They have a slight difference in definition, which leads to quite different optimization strategies in FALCON as we will show later.

Definition 2.1. Given two sets of points $A = \{a_1, ..., a_m\}$ and $B = \{b_1, ..., b_n\}$, the minimal Hausdorff distance is defined as:

$$H_{min}(A, B) = \max\{h_{min}(A, B), h_{min}(B, A)\} \tag{1}$$

where

$$h_{min}(A, B) = \min_i \min_j ||a_i - b_j|| \tag{2}$$

Definition 2.2. Given two sets of points $A = \{a_1, ..., a_m\}$ and $B = \{b_1, ..., b_n\}$, the maximal Hausdorff distance is defined as:

$$H_{max}(A, B) = \max\{h_{max}(A, B), h_{max}(B, A)\} \tag{3}$$

where

$$h_{max}(A, B) = \max_i \min_j ||a_i - b_j|| \tag{4}$$

Although in the definition, $H_{min}(A, B)$ equals to the larger one between $h_{min}(A, B)$ and $h_{min}(B, A)$, they actually always have the same value as h_{min} is symmetric. Both of them equal to the distance between two closest instances in two bags. Therefore, we only need to calculate one of them to get the minimal Hausdorff distance between two bags. However, this is not the case for maximal Hausdorff distance in which $h_{max}(A, B)$ and $h_{max}(B, A)$ may differ.

Let $d(a_i, b_j)$ represent the Euclidean distance between instance a_i and b_j and $d(a_i, B)$ is the smallest distance from instance a_i to bag B. As shown in Figure 1, $h_{max}(A, B) = \max_i d(a_i, B) = d(a_1, b_1)$ while $h_{max}(B, A) = \max_j d(b_j, A) = d(a_2, b_2)$. Clearly, $d(a_1, b_1)$ is not necessarily the same as $d(a_2, b_2)$. To get maximal Hausdorff distance, we have to calculate both $h_{max}(A, B)$ and $h_{max}(B, A)$.

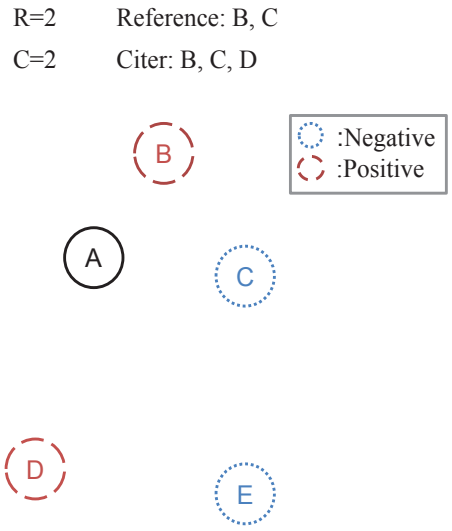

Figure 2: R-nearest references and C-nearest citers of an unknown bag A

2.2 Reference and Citer

Besides Hausdorff distance, the other important concepts in Citation KNN are *reference* and *citer*. Citation KNN borrows the notion of citation from library and information science: if a paper cites a previously published paper, the paper is deemed to be related to the reference. Similarly, if a paper is cited by a subsequent paper, the citer is also regarded relevant to the paper. When this idea comes to the context of MIL, it suggests that when labeling a bag, the algorithm should take into account not only its neighbors but also the bags that count the concerned bag as a neighbor. For example, as shown in Figure 2, we have an unknown bag A to label. Firstly, we find its R-nearest neighbors. Subsequently, we probe each bag other than bag A to see whether it has bag A in its C-nearest neighbors. If it does, that bag is a citer of bag A. In this example, it is easy to see that references of bag A are bags B and C while citers of bag A are bags B, C, D. Both references and citers will vote to determine the label of bag A. It is worth noting that bags B and C are both references and citers of bag A. They thereby each have two votes. In our example, bag B and bag D are positive while bag C is negative. Bag A is labeled positive due to 3 positive votes over 2 negative votes.

2.3 Citation KNN Algorithm

In Citation KNN, for a given unknown bag, the key step is to find its references and citers. References are the nearest neighbors of the concerned bag, while for a bag to be a citer of the concerned bag, its nearest neighbors must include the concerned bag. These two notions are closely related. If we have an algorithm that finds K nearest bags for a specific bag, the algorithm can be applied not only on the concerned bag to find out its references but also on other bags to find their K nearest bags to see whether the concerned bag is one of them. This is the essential idea underlying the de facto implementation of Citation KNN.

Algorithm 1 gives the pseudo code. It works for either min Hausdorff or max Hausdorff. It first calculates the distance between each pair of bags. Since Hausdorff distance is symmetric, for each pair of bags, the distance only needs to be computed once. For the concerned bag, the algorithm finds its R-nearest bags as its references and put them into the voter list. For any other bag, if the concerned bag is one of its C-nearest bags, it is considered as a citer of the concerned bag; it is also put into the voter list. Finally, all the bags in the voter list vote to determine the label of the concerned bag. Most time is taken by the calculations of the distances between bags. As Hausdorff distance of two bags is based on the distances between all instances in the two bags, the complexity is quadratic to the total number of instances.

Citation KNN
input : Bags from input dataset, bag A is the query bag to be labeled,the number of bags is N
output: Label for bag A
for $i \leftarrow 1$ **to** $N-1$ **do**
  for $j \leftarrow i+1$ **to** N **do**
    $H(bag_i, bag_j)$ = HausdorffDist(bag_i,bag_j);
    $H(bag_j, bag_i) = H(bag_i, bag_j)$;
  end
  if $i == A$ **then**
    find R smallest values in $H(bag_i, bag_t)$ (t=1,2,...,N);
    Insert the R bag_ts into *VoterList*;
  else
    find C smallest values in $H(bag_i, bag_t)$ (t=1,2,...,N);
    if *bag A is one of the C* bag_t *s* **then**
      Insert bag_i into *VoterList*;
    end
  end
end
return the majority bag label in $VoterList$;

Algorithm 1: Original Citation KNN algorithm

3 OPTIMIZATION APPROACHES

FALCON speeds up Citation KNN by avoiding most of the distance calculations. It has two key ideas, *multi-level triangle inequality-based distance filtering* and *heap optimization*.

3.1 Multi-Level Filtering

The first key idea tries to avoid unnecessary distance calculations through bounds-based filtering. FALCON does it at both bag level and instance level.

In the subsequent sections, we introduce filtering conditions and bounds of distances used by FALCON. To make used notations easily accessible, they are organized in Table 1.

Our optimization is based on the well-known triangle inequality and its definition is given as follows:

THEOREM 3.1. *The sum of the lengths of any two sides of a triangle must be greater than or equal to the length of the remaining side.*

Table 1: Notations Used in the Paper

Notations	Definitions
$d(a_i, b_j)$	Euclidean distance between instance a_i and instance b_j
$d(a_i, B)$	The smallest Euclidean distance between instance a_i and bag B, $d(a_i, B) = \min_j d(a_i, b_j)$
$d_{cur}(a_i, B)$	The currently smallest Euclidean distance between instance a_i and bag B
$h_{min}(A, B)$	The smallest Euclidean distance from bag A to bag B, $h_{min}(A, B) = \min_i d(a_i, B)$
$h_{max}(A, B)$	The largest one of all smallest Euclidean distances between instance a_i and bag B, $h_{max}(A, B) = \max_i d(a_i, B)$
$h_{cur-min}(A, B)$	The currently smallest Euclidean distance from bag A to bag B
$h_{cur-max}(A, B)$	The currently largest one of all smallest Euclidean distances between instance a_i and bag B
$H_{min}(A, B)$	Minimal Hausdorff distance, $H_{min}(A, B) = \max(h_{min}(A, B), h_{min}(B, A))$
$H_{max}(A, B)$	Maximal Hausdorff distance, $H_{max}(A, B) = \max(h_{max}(A, B), h_{max}(B, A))$
$H(A, B)$	Hausdorff distance between bag A and bag B, representing both $H_{min}(A, B)$ and $H_{max}(A, B)$
$UB/LB(A, B)$	Upper/ Lower bound of Hausdorff distance between bag A and bag B, can represent both minimal and maximal Hausdorff distance
$UB/LB(a_i, b_j)$	Upper/ Lower bound of Euclidean distance between a_i and b_j

In the context of Citation KNN, let $d(x, y)$ represent the Euclidean distances between instance x and y. For any three instances a, b, c, we have:

$$|d(a, b) - d(b, c)| \leqslant d(a, c) \leqslant d(a, b) + d(b, c) \tag{5}$$

As long as $d(a, b)$ and $d(b, c)$ are known, triangle inequality provides a lower bound and an upper bound for $d(a, c)$. These bounds will be further used to avoid unnecessary calculations. For example, we want to find the nearest neighbor for instance a, if the lower bound of $d(a, c)$ is larger than current shortest distance, there is no need to calculate the exact value of $d(a, c)$. This theorem can be easily extended to include four instances which form a quadrilateral (or skew quadrilateral if instances are not on the same plane) as shown in Figure 3. Similarly, the sum of lengths of any three edges of this quadrilateral/skew quadrilateral must be greater than or equal to the length of the remaining edge. Therefore, for instance a, b, c, d, we have:

$$\max\{d(a, b)-d(b, c)-d(c, d), 0\} \leqslant d(a, d) \leqslant d(a, b)+d(b, c)+d(c, d) \tag{6}$$

3.1.1 Bag-level Filtering. FALCON has two levels of filtering. Firstly, bag-level filtering is applied. If the filtering condition is satisfied, the whole bag is filtered out and won't be considered further. If it fails, FALCON needs to go inside the bag and calculate distances between instances. Then instance-level filtering applies.

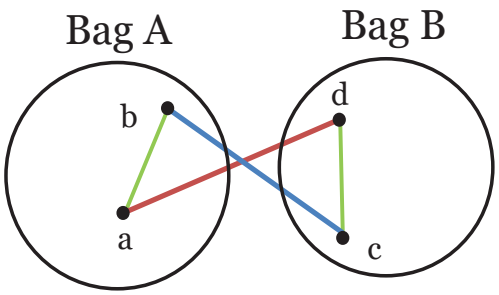

Figure 3: The extension of triangle inequality to quadrilateral / skew quadrilateral

Due to the difference between minimal Hausdorff distance and maximal Hausdorff distance, we derive different bounds.

Lemma 3.2. $\forall a_1 \in A, b_1 \in B$,

$$H_{min}(A, B) \geqslant d(a_1, b_1) - \max_t d(a_1, a_t) - \max_u d(b_1, b_u) \tag{7}$$

Proof. According to Definition 2.1, $H_{min}(A, B)$ equals to the distance between two closest instances in bag A and bag B. Let a_i and b_j be the two closest instances. Using the Formula (6), we have:

$$H_{min}(A, B) = d(a_i, b_j) \geqslant d(a_1, b_1) - d(a_1, a_i) - d(b_1, b_j)$$

Although the values of i and j are unknown, the distance between a_1 and a_i must be less than or equal to the longest distance from a_1 to all other instances in bag A: $d(a_1, a_i) \leqslant \max_t d(a_1, a_t)$. Similarly, $d(b_1, b_j) \leqslant \max_u d(b_1, b_u)$, thus:

$$\begin{aligned} H_{min}(A, B) = d(a_i, b_j) &\geqslant d(a_1, b_1) - d(a_1, a_i) - d(b_1, b_j) \\ &\geqslant d(a_1, b_1) - \max_t d(a_1, a_t) - \max_u d(b_1, b_u) \end{aligned}$$

□

Lemma 3.3. $\forall a_1 \in A, b_1 \in B$,

$$H_{max}(A, B) \geqslant d(a_1, b_1) - \min(\max_t d(a_1, a_t), \max_u d(b_1, b_u)) \tag{8}$$

Proof. Assume $H_{max}(A, B)$ equals $d(a_i, b_j)$ and the closest instance in bag B to a_1 is b_k. Thus, $d(a_i, b_j) \geqslant d(a_1, b_k)$. Instances a_1, b_k, b_1 form a triangle, as shown in Figure 4. Using Formula 5, we have:

$$\begin{aligned} H_{max}(A, B) = d(a_i, b_j) &\geqslant d(a_1, b_k) \geqslant d(a_1, b_1) - d(b_1, b_k) \\ &\geqslant d(a_1, b_1) - \max_u d(b_1, b_u) \end{aligned}$$

Assume the closest instance in bag A to b_1 is a_h. Using the same strategy to get another lower bound:

$$\begin{aligned} H_{max}(A, B) = d(a_i, b_j) &\geqslant d(a_h, b_1) \geqslant d(a_1, b_1) - d(a_1, a_h) \\ &\geqslant d(a_1, b_1) - \max_t d(a_1, a_t) \end{aligned}$$

Since bounds are supposed to be tight, we always use the larger one of the lower bounds, which is shown as Formula (8). □

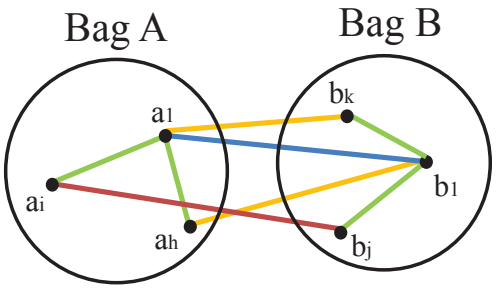

Figure 4: Bag-level filtering for maximal Hausdorff distance

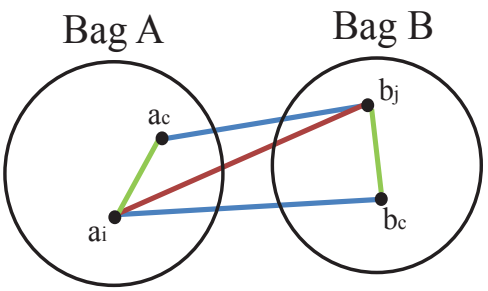

Figure 5: Instance-level filtering

Lemma 3.2 and 3.3 give the lower bounds of the Hausdorff distance between bag A and bag B. These lower bounds are used in filtering conditions.

***Filtering condition* 1:** Assume currently, bag T is the farthest bag among bag A's current K nearest neighbors in terms of Hausdorff distance (either min or max). If $LB(A, B) \geqslant H(A, T)$, bag B is not one of K nearest neighbors of bag A. ($H(A, T)$ is defined in Table 1)

Because we know bag B is not in bag A's K nearest neighbors, it's unnecessary to calculate the exact Hausdorff distance between bag A and bag B. This bag-level filtering brings benefits but also incurs overhead, because the distances from the chosen instance (a_1 or b_1) to all other instances in the same bag need to be calculated. For each bag, the incurred calculations are $O(t + 1)$, t is the number of instances in the bag, 1 represents the distance calculation between a_1 and b_1. If one bag is skipped, the avoided calculations are $O(t^2)$. Even though the overhead is much smaller than the benefits, it is always good to minimize the overhead. FALCON avoids the overhead by using *landmarks* as a_1 and b_1. A landmark is just a point in the data space that is used to form triangles with the points in the space. There could be multiple landmarks; each instance is often associated with the landmark that is closest to it. In our design, the landmarks are chosen such that the distances between them and other instances in the same bag are already known before we do this filtering; that helps minimize the overhead. More details are given in Section 3.1.3.

3.1.2 Instance-level Filtering. If bag-level filtering fails, we have to calculate the distance between two bags. Instance-level filtering is used to avoid unnecessary distance calculations for instances.

Instance-level filtering leverages landmarks to construct triangles. The landmarks and the instances form many triangles, based on which, the bounds of instance distances can be attained. The distance between a landmark and an instance can be reused for all the triangles involving that edge.

This part presents three instance-level filters used in FALCON. In the following discussion, we use a_c and b_c to represent the landmarks that a_i and b_i are associated with, respectively.

Lemma 3.4.

$$d(a_i, b_j) \geqslant \max(|d(a_c, b_j) - d(a_i, a_c)|, |d(a_i, b_c) - d(b_j, b_c)|) \quad (9)$$

$$d(a_i, b_j) \leqslant \min(d(a_c, b_j) + d(a_i, a_c), d(a_i, b_c) + d(b_j, b_c)) \quad (10)$$

Proof. a_i and b_j are any two instances in bag A and bag B. Let a_c and b_c be landmarks of a_i and b_j, respectively. As shown in Figure 5, there are two triangles: $\triangle a_i b_j a_c$ and $\triangle a_i b_c b_j$. For each of them, using Formula 5, we can get a lower bound and an upper bound for $d(a_i, b_j)$:

$$\triangle a_i b_j a_c : |d(a_c, b_j) - d(a_i, a_c)| \leqslant d(a_i, b_j) \leqslant d(a_c, b_j) + d(a_i, a_c)$$
$$\triangle a_i b_c b_j : |d(a_i, b_c) - d(b_j, b_c)| \leqslant d(a_i, b_j) \leqslant d(a_i, b_c) + d(b_j, b_c)$$

Since bounds are supposed to be tight, we use the larger one of lower bounds and the smaller one of upper bounds which lead to Formula (9) and (10). □

A series of filtering conditions have been carefully designed to leverage these bounds.

***Filtering condition* 2:** If $LB(a_i, b_j) \geqslant d_{cur}(a_i, B)$, there is no need to calculate $d(a_i, b_j)$.

This idea is straightforward. To calculate Hausdorff distance, the first step is to find out shortest distances from every instance in one bag to another bag. Since $LB(a_i, b_j) \geqslant d_{cur}(a_i, B)$, $d(a_i, b_j)$ will not be the shortest distance. Therefore, we do not need to know its exact value.

***Filtering condition* 3:** For minimal Hausdorff distance calculation, if $LB(a_i, b_j) \geqslant h_{cur-min}(A, B)$, there is no need to calculate $d(a_i, b_j)$.

The filtering condition works for minimal Hausdorff distance calculation. Since the goal is to find the shortest distance between bag A and bag B and there is already a distance $h_{cur-min}(A, B)$ less than or equal to $d(a_i, b_j)$, we know $d(a_i, b_j)$ cannot be the shortest distance. Therefore we can skip the calculation.

***Filtering condition* 4:** For maximal Hausdorff distance, if we have $UB(a_i, b_j) \leqslant h_{cur-max}(A, B)$, then we can skip a_i. Also, if $UB(a_i, b_j) > h_{cur-max}(A, B)$ but $d(a_i, b_j) \leqslant h_{cur-max}$, we still can skip subsequent calculations for a_i.

Maximal Hausdorff distance is the largest one of all shortest distances from instances in bag A to bag B. If $UB(a_i, b_j)$ or $d(a_i, b_j)$ is less than or equal to $h_{cur-max}(A, B)$, for both cases, we have $d(a_i, B) \leqslant d(a_i, b_j) \leqslant h_{cur-max}(A, B)$. It means $d(a_i, B)$ will not be the largest one among all shortest distances from instances in bag A to bag B. Thus, it's not necessary to calculate the exact value of $d(a_i, B)$.

3.1.3 Selecting Landmarks. In previous sections, we have mentioned how to capitalize on landmarks to construct triangles so as

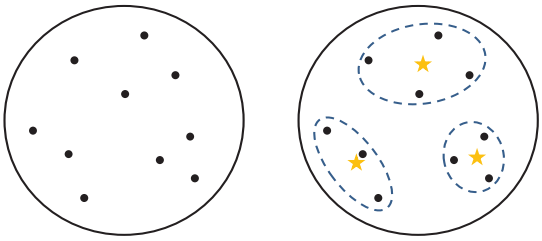

Figure 6: Dividing instances into groups

to apply filtering conditions. This part explains our strategies in setting these landmarks.

Good landmarks are essential for the tightness of the bounds. There are three criteria in setting landmarks: (1) They should be close to instances: Landmarks close to instances are helpful for constructing tighter bounds per triangle inequality; (2) Landmarks ideally should be sharable among multiple instances: Landmarks introduce extra calculations and the sharing lowers the overhead; (3) Landmarks should be fast to compute.

We draw on the ideas of clustering (e.g. KMeans) for selecting landmarks. Instances are divided into groups based on distances to group centers. Instances in one group share the group center as the landmark. As shown in Figure 6, instances inside one dot line cycle form a group. They all use the group center, shown as yellow star, as their landmark.

To compute the group center, classic KMeans runs for many iterations until convergence. That is often too costly for our usage. Though more iterations tend to give tighter bounds, it also incurs more overhead. We have analyzed cost-benefit trade-offs of different numbers of iterations. We find that using the initial centers chosen by KMeans++ [3] works well for us. What it does is: (1) choosing the first center randomly from all the instances; (2) After that, each subsequent center is chosen from remaining instances with probability proportional to its distance to the closest centers. The centers are directly taken as our landmarks. Another benefit this method brings is that since no further iterations are executed, all the centers are instances. The distance calculations, during instance-level filtering, between centers and instances are actual distances between instances. Therefore these results can be reused when calculating distance between instances.

It is easy to see that the filtering conditions described so far can be applied to the calculations of both references and citers. There are some filtering conditions specific to the calculations of citers. As they are related with *heap optimization*, we explain them in the next part.

3.2 Heap Optimization and Citer-Specific Filtering

Recall that the basic Citation KNN computes the K nearest neighbors of all bags. The second key optimization in FALCON is named *heap optimization*, which is based on the following observation:

> To get the references of a bag A, it is necessary to get the K nearest neighbors of A, but to determine whether a bag B is the citer of bag A, we do not have to compute all the K nearest neighbors of bag B; we only need to know whether bag A is one of the K nearest neighbors of bag B.

Based on this observation, we design a strategy which keeps track of the concerned bag in other bags' K nearest neighbor lists.

3.2.1 Heap Optimization. Let's assume the concerned bag is bag A and we would like to know whether bag B is A's citer. For bag B, we maintain a list of its current K nearest neighbors. Every time when the distance between bag B and another bag has been calculated, that bag is compared with the largest distance of the current list of K nearest neighbors; the list is updated accordingly. In our strategy, the distance between bag B and the concerned bag (i.e.,A) is calculated first such that the concerned bag is put into that list first. Then distances between bag B and other bags are calculated and bags are inserted into that list one by one. The bag with the largest distance will be removed when the list has more than K bags. If the bag removed is bag A, we can stop and skip all the subsequent calculations of nearest neighbors of bag B. Because it is certain that A cannot be in that list again, which means that bag B is not a citer of bag A.

We find that the idea can be effectively materialized through the use of heap data structure. Since we need to find the bag with largest distance and remove it from the list when there are more than K bags in that list, iterating through the whole list is inefficient. Maintaining a sorted list can help but is still not the optimal choice either. Because we are only interested in the one with the largest distance, keeping other bags in order causes unnecessary overhead. *Max-heap* has some properties that make it a desirable choice. On max-heap, finding the bag with the largest distance takes $O(1)$ time while inserting and deleting a bag take $O(\log(K))$ time. Figure 7 gives an example to show the process of checking whether bag B is a citer of bag A. Let K = 3. Concerned bag is calculated first, then distances to every other bag are calculated and bags are inserted into the max-heap. We get an early stop when bag A is removed.

3.2.2 Citer-specific Filtering. Based on *heap optimization*, we identify more opportunities of distance filtering for citer calculations.

According to *heap optimization*, during the process of finding citers, what we care about is whether the concerned bag is one of the K nearest neighbors of other bags. For example, we check bag B to see whether it's a citer of bag A. Due to heap optimization, bag A is the first one to be added to bag B's K nearest neighbor list. Thus, what interests us is whether bag A is removed from that list. Because if bag A is removed from that list, we know bag B is not the citer and thereby skip subsequent calculations for bag B.

Distance calculations fall into two categories: those that have no influence on the position of bag A in the K nearest neighbor list, and those that do. Distance calculations in the first category can be avoided, since it makes no difference on the position of bag A. We design some special filters for the second category.

***Filtering condition* 5:** Assume the concerned bag is bag A and we want to find out whether bag B is a citer of bag A. For minimal Hausdorff distance, if $LB(b_i, c_j) \geqslant H_{min}(A, B)$, we can skip the calculation of $d(b_i, c_j)$, where, c_j is an instance in another bag C.

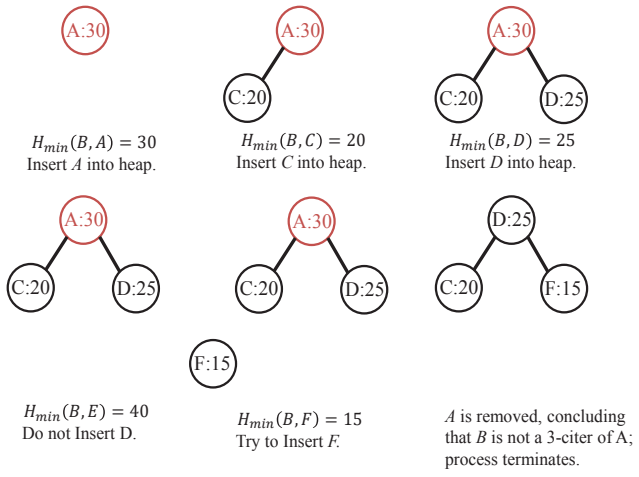

Figure 7: Process of inserting bags to max-heap for determining whether bag B is a 3-citer of bag A (concerned bag).

The correctness can be proved by examining all scenarios of $d(b_i, c_j)$. (1) If $d(b_i, c_j) \neq d(b_i, C)$, we do not need to calculate it, since what we are interested in is $d(b_i, C)$. (2) If $d(b_i, c_j) = d(b_i, C)$, then there are still two possible cases: (2.1) $d(b_i, c_j) = d(b_i, C) \neq H_{min}(B, C)$. Because what we are looking for is $H_{min}(B, C)$ and $d(b_i, C) \neq H_{min}(B, C)$, it's unnecessary to calculate $d(b_i, C)$; (2.2) $d(b_i, c_j) = d(b_i, C) = H_{min}(B, C)$. In this case, we have $H_{min}(B, C) = d(b_i, C) = d(b_i, c_j) \geqslant H_{min}(A, B)$. Thus, the position of bag A will not be affected, and the calculation of $d(b_i, c_j)$ can be avoided.

***Filtering condition* 6:** Assume the concerned bag is bag A and we want to find out whether bag B is a citer of bag A. For minimal Hausdorff distance, if $UB(b_i, c_j) \leqslant H_{min}(A, B)$ or $d(b_i, c_j) \leqslant H_{min}(A, B)$, we can skip all subsequent distance calculations between bag B and bag C and continue to the next bag (c_j is an instance in bag C).

This is because $H_{min}(B, C) \leqslant d(b_i, C) \leqslant d(b_i, c_j) \leqslant UB(b_i, c_j) \leqslant H_{min}(A, B)$. We can infer $H_{min}(B, C)$ is smaller than $H_{min}(A, B)$ without calculating its exact value. This time the position of bag A will be changed. Since what we care about is the position change of bag A and whether bag A is still in the list, we do not need to know the exact value of $H_{min}(B, C)$: As long as $H_{min}(B, C)$ is smaller than $H_{min}(A, B)$, the influence it brings to the position change of bag A is the same. The implementation can either explicitly change the position of bag A or implicitly change it by assigning $H_{min}(B, C)$ a very small distance value (e.g. 0) without calculating its exact value.

***Filtering condition* 7:** Assume the concerned bag is bag A and we want to find out whether bag B is a citer of bag A. For maximal Hausdorff distance, if $h_{cur-max}(B, C) \geqslant H_{max}(A, B)$, we can skip all subsequent distance calculations for bag C.

This filtering condition is straightforward. Since $H_{max}(B, C) \geqslant h_{max}(B, C) \geqslant h_{cur-max}(B, C) \geqslant H_{max}(A, B)$, therefore, the position of bag A will not be affected. Calculating the exact distance of $H_{max}(B, C)$ is not necessary. It's worth noting that if $H_{max}(B, C)$ is calculated, it may be reused in the future for labeling other bags.

Table 2: Datasets Used in Experiments

Dataset	Type	#bag	#inst	#dim
Musk2	mole. prediction	102	6598	166
Sival Apple	img. retrieval	1500	47414	30
Breast Cancer	img. classification	60	2002	708
Web Recom. 2	txt. classification	75	2219	6519
Harddrive	fail. prediction	369	68411	61
Protein	pro. prediction	193	26611	8
Corel African	img. classification	2000	7947	9

FALCON calculates $H_{max}(B, C)$ at that time only when it turns out to be necessary.

The two optimizations, *multi-level filtering* and *heap optimization*, together form a synergy that drastically reduces the amount of distance calculations needed in the original Citation KNN. They form the core of the FALCON algorithm.

4 EXPERIMENTS

We evaluate the efficacy of FALCON on a variety of MIL benchmark datasets as shown in Table 2. Musk2 [7] is for musky molecule prediction. Each bag is a molecule and each instance is a conformation of that molecule. SIVAL [20] is a dataset for image retrieval. Each image is a bag and every segment is an instance. UCSB breast cancer [12] is a medical image dataset in which bags represent images while instances are patches. Dataset of Web recommendation [34] is used for text classification. Webpages are considered as bags and links in the webpage are instances. Harddrive dataset [17] is used for hard drive failure prediction, in which every instance is measurements of harddrive at one time point. A Bag contains all instances in a period of time. In Protein dataset [23], a bag is a protein and an instance corresponds to the amino acid sequence. The goal is to predict whether a protein is a TrX-fold protein. Corel [6] is for image classification. Like other image classification datasets, bags represent images and instances correspond to patches.

The datasets are chosen to be diverse. The numbers of bags range from 60 to 2000, instances range from 2002 to 68411, and dimensions range from 8 to 6519.

Our experiments measure the efficacy of FALCON based on the number of distance calculations avoided as well as speedups it achieves. All the experiments are conducted in the following experimental environment: PowerEdge R620 equipped with 2 Xeon E5-2670 CPUs, 128GB memory, Ubuntu 14.04.

Experiments are conducted on original Citation KNN, Citation KNN + Heap Optimization and FALCON can be considered as Citation KNN + Heap Optimization + Multi-level Filtering. R and C are set to 2 and 4 in our experiments, which is the setting used by Wang et al. in their original Citation KNN paper [27]. In FALCON, the number of landmarks computed and used for each bag is $\lfloor Z/10 \rfloor$ where Z is the number of instances in a bag. We explored a spectrum of values, and found that that setting gave the best overall results. In all experiments, the optimized algorithms produce exactly the same results as the original Citation KNN does. We hence focus the following discussions on speedups. The times reported in this section include all runtime overhead.

Table 3: Distance Calculations and the Reductions by Proposed Optimizations

dataset	distance type	Distance Calculations			Avoided Calculations	
		Citation KNN	Citation KNN + Heap Optimization*	FALCON*	Citation KNN + Heap Optimization	FALCON
Musk2	min HD	19995014	5252878 (26.27%)	1622628 (1.7%)	73.8%	91.9%
	max HD	19995014	9462090 (47.32%)	857137 (1.09%)	52.7%	95.7%
Sival Apple	min HD	1123294157	78164942 (6.96%)	15178458 (0.1%)	93.0%	98.6%
	max HD	1123294157	37995169 (3.38%)	10156332 (0.41%)	96.6%	99.1%
Breast Cancer	min HD	1968726	569047 (28.9%)	119847 (1.61%)	71.1%	93.9%
	max HD	1968726	1104097 (56.08%)	301751 (7.15%)	43.9%	84.7%
Web recom.2	min HD	2400819	577808 (24.07%)	256519 (6.44%)	75.9%	89.3%
	max HD	2400819	782251 (32.58%)	180961 (3.58%)	67.4%	92.5%
Harddrive	min HD	2330996059	126960690(5.45%)	3919790 (0.01%)	94.6%	99.8%
	max HD	2330996059	87705823 (3.76%)	3649975 (0.03%)	96.2%	99.8%
Protein	min HD	352014663	72952148 (20.7%)	3537020 (0.03%)	79.3%	99.0%
	max HD	352014663	68280722 (19.4%)	2506547 (0.17%)	80.6%	99.3%
Corel African	min HD	31557751	1190722 (3.77%)	587124 (0.3%)	96.2%	98.1%
	max HD	31557751	1837272 (5.82%)	802156 (1.02%)	94.2%	97.5%

*: the distance calculations in columns 4 and 5 include both those left from original Citation KNN and those newly introduced for optimizations. The percentages in parentheses show the numbers of distance calculations left from original Citation KNN divided by column 3.

Table 4: Running Time Comparison For Different Algorithms

dataset	distance type	Running time (ms)			Speedups (X) over Citation KNN	
		Citation KNN	Citation KNN + Heap Optimization	FALCON	Citation KNN + Heap Optimization	FALCON
Musk2	min HD	53882	14075	6011	3.8	9.0
	max HD	64839	25937	2965	2.5	21.9
Sival Apple	min HD	674207	46865	31070	14.4	21.7
	max HD	760075	25593	10379	29.7	73.2
Breast Cancer	min HD	22571	6415	1622	3.5	14.0
	max HD	22878	12963	3755	1.8	6.1
Web recom.2	min HD	248650	61088	28329	4.1	8.8
	max HD	253788	82795	21164	3.1	12.0
Harddrive	min HD	2336980	127052	19741	18.4	118.4
	max HD	2416732	91120	7225	26.5	334.5
Protein	min HD	62669	12627	10765	5.0	5.8
	max HD	73666	13861	2169	5.3	34.0
Corel African	min HD	28238	943	1494	29.9	18.9
	max HD	38769	1739	1453	22.3	26.7

4.1 Avoided Calculations

Table 3 reports the numbers of distance calculations each of versions has on every dataset to label an unknown bag. For all datasets, the first bag is treated as the test bag, all other bags are used as training bags. Except for the numbers in the parentheses, all the numbers are based on the total distance calculations. For FALCON, it includes both the ones left from the original Citation KNN and those introduced by our optimizations. The percentages shown in parentheses of columns four and five show the fractions of distance calculations left from the original citation CNN.

According to column six, on average, 80% distance calculations are avoided by Heap Optimization. The benefits vary on these datasets. For example, when we use maximal Hausdorff distance (maxHD) in Breast Cancer dataset, it only reduces less than 50% distance calculations. But for Sival Apple dataset, more than 90% distance calculations can be avoided by this approach. FALCON manifests consistent superior performance. On average, FALCON avoids more than 94% distance calculations compared with the original Citation KNN. It shows significant improvements, especially on datasets where Heap Optimization does not perform very well. For the Breast Cancer dataset with maxHD, FALCON attains 84.7% reduction which is about 2 time more than what Heap Optimization gets. On Musk2 dataset with maxHD, FALCON improves the result from 52.7% to 95.7%. For some other datasets, the improvement is

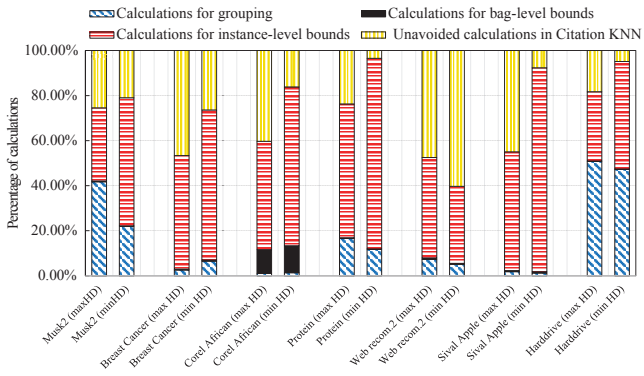

Figure 8: Breakdown of distance calculations of FALCON.

not that large. One important reason is that Heap Optimization has done a good job on those datasets. There is not much space left for multiple-level filtering to improve. It is worth noting that according to Table 3, the best results are on dataset Harddrive, Protein, and Sival Apple which happen to have the most bags and instances among all datasets. This is consistent with our intuition that more bags and instances lead to more space for improvement. With the growing size of dataset, FALCON is expected to have even better performance.

4.2 Overall Speedups

Table 4 reports the run times and the overall speedups that the heap optimization and FALCON bring. In the calculations of the speedups, the run times of FALCON include all the overhead incurred by all the filters and optimizations it uses. (We report a detailed analysis of the overhead in the next subsection.)

FALCON brings one or two orders magnitude speedups for almost all the datasets. On average, Heap Optimization achieves 11 times speedups over Citation KNN. Based on that, multi-level optimization further brings another four times speedups. One of results we would like to highlight is that FALCON attains 334.5X speedups in Harddrive dataset with maxHD. This attributes to FALCON's outstanding performance in reducing calculations. According to Table 3, FALCON successfully avoids 99.8% calculations on Harddrive dataset with maxHD.

Heap optimization alone leads to some substantial speedups (1.8–29.9X), upon which, multi-level filtering adds significantly more on most datasets. An exception is Corel African dataset with minHD, heap optimization excels in this case. From our understanding, this is because Corel African dataset has limited number of instances per bag. Many bags only has two instances. There is not enough space for multi-level filtering to take effect and extra calculations associated with filtering make FALCON more expensive choice.

On average, FALCON gives more speedups on maxHD than on minHD. It is due to Filtering condition 4, which helps remove a large number of calculations in the maxHD case. Another reason is that maxHD gets a much tighter bound at bag-level filtering. The cost for filtering a distance calculation at bag-level is much cheaper than that at instance level. That also explains why maxHD is faster than minHD even if the avoided amounts of calculations are almost the same.

Comparisons with Alternatives. Based on the relative speeds of Citation KNN and other MIL algorithms reported in previous literatures [18], the significant speedups over Citation KNN make FALCON the fastest MIL. For instance, on Musk2, one of the most popular datasets used in almost every MIL literature, the speedup from FALCON over Citation KNN (9X) makes it substantially outperform all the 10 MIL algorithms measured in the previous study [18] (e.g., 22X over mi-DS, 114X over mi-DD, and 55X over mi-EMDD). It is worth noting that the speedup of FALCON on Musk2 is actually one of the most modest ones; it shows much larger speedups over Citation KNN on other datasets as reported in Table 4.

4.3 Detailed Analysis

This part provides a breakdown of the distance calculations in FALCON. They fall into four categories: (1) Calculations for grouping; (2) Calculations for bag-level bounds; (3) Calculation for instance-level bounds; (4) Unavoided calculations in Citation KNN.

Calculations for grouping: When FALCON divides each instances into a group by KMeans++, the distance calculations take place between instances and group centers in the same bag. This is an extra overhead when compared with the original Citation KNN in which distances between instances in the same bag need not to be calculated. The amount of this type of calculations depends on the size of the bag. Since we use k = $\lfloor Z/10 \rfloor$, where k is the number of groups in the bag and Z is the number of instance in the bag, it's easy to see this overhead is $O(Z^2)$ for one bag. As Figure 8 shows, as only one iteration of KMeans++ take places, this category does not weigh heavily for most cases. On dataset Harddrive, grouping calculations weight relatively a large portion. That is because it has a much larger bag size than other datasets according to Table 2.

Calculations for bag-level bounds: This type of calculation is for bag-level filtering. For each pair of bags, only the distance between the first instances in two bags is calculated. In addition, distances between the first instance and all other instances in the same bag are calculated to find the longest one which is required by Formulas 8 and 9. The overhead is $O(N + M^2)$, where N is the total number of instances and M is the number of bags. It is negligible when compared with other types of distance calculations as shown in Figure 8.

Calculations for instance-level bounds: This part is the heaviest for most datasets in our experiments. When bag-level filtering fails, calculations come to the instance level. Instances-level bounds are much tighter than those at bag-level, but it also requires much more calculations to build. One group center needs to have distance to all instances in the other group so that the distance can be reused by its group members to construct triangles. In the worst case, the overhead is $O(Z^2)$ for each bag. Fortunately, FALCON sees a much smaller overhead than that: Since all the group centers are instances, all of these distance are reusable in the subsequent steps.

Unavoided calculations in Citation KNN: This part of distance calculations can not be avoided with any of the filters. Their exact values must be calculated. As shown in Figure 8, the percentage of remaining calculations in dataset *Web Recom.2* is relatively large

because the large ratio between its dimensionality and its number of bags makes distance filtering relatively more challenging.

Overall, the experiments indicate that FALCON is effective across the numbers of dimensions or dataset sizes. The benefits are the greatest when the dataset is large (with many data instances and bags), as the large number of distance calculations in such settings provide a large room for the filters to take effects.

5 CONCLUSIONS

This work studies Citation KNN and proposes FALCON as a practical replacement of Citation KNN with one or two orders of magnitude speedups. FALCON detects and avoids unnecessary calculations via carefully designed *multi-level filtering* and *heap optimization*. Experiment shows that, FALCON reduces 84–99.8% distance calculations and achieves 6–334X speedups without affecting the results of Citation KNN. FALCON shows the promise as a drop-in replacement of Citation KNN for practical MIL studies and applications. (The source code of FALCON is available in https://www.github.com/PICTureRG/FALCON)

ACKNOWLEDGMENTS

This material is based upon work supported by DOE Early Career Award (DE-SC0013700), the National Science Foundation (NSF) under Grant No. CCF-1455404, CCF-1525609, CNS-1717425, CCF-1703487. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of DOE or NSF.

REFERENCES

[1] Saad Ali and Mubarak Shah. 2010. Human action recognition in videos using kinematic features and multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence* 32, 2 (2010), 288–303.

[2] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. 2003. Support vector machines for multiple-instance learning. In *Advances in neural information processing systems*. 577–584.

[3] David Arthur and Sergei Vassilvitskii. 2007. k-means++: The advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 1027–1035.

[4] Peter Auer and Ronald Ortner. 2004. A boosting approach to multiple instance learning. In *European Conference on Machine Learning*. Springer, 63–74.

[5] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. 2011. Robust object tracking with online multiple instance learning. *IEEE transactions on pattern analysis and machine intelligence* 33, 8 (2011), 1619–1632.

[6] Yixin Chen, Jinbo Bi, and James Ze Wang. 2006. MILES: Multiple-instance learning via embedded instance selection. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28, 12 (2006), 1931–1947.

[7] Thomas G Dietterich, Richard H Lathrop, and Tomás Lozano-Pérez. 1997. Solving the multiple instance problem with axis-parallel rectangles. *Artificial intelligence* 89, 1-2 (1997), 31–71.

[8] Murat Dundar, Balaji Krishnapuram, RB Rao, and Glenn M Fung. 2007. Multiple instance learning for computer aided diagnosis. In *Advances in neural information processing systems*. 425–432.

[9] M Murat Dundar, Sunil Badve, Vikas C Raykar, Rohit K Jain, Olcay Sertel, and Metin N Gurcan. 2010. A multiple instance learning approach toward optimal classification of pathology slides. In *Pattern Recognition (ICPR), 2010 20th International Conference on*. IEEE, 2732–2735.

[10] Zach Jorgensen, Yan Zhou, and Meador Inge. 2008. A multiple instance learning strategy for combating good word attacks on spam filters. *Journal of Machine Learning Research* 9, Jun (2008), 1115–1146.

[11] Melih Kandemir and Fred A Hamprecht. 2015. Computer-aided diagnosis from weak supervision: A benchmarking study. *Computerized Medical Imaging and Graphics* 42 (2015), 44–50.

[12] Melih Kandemir, Chong Zhang, and Fred A Hamprecht. 2014. Empowering multiple instance histopathology cancer diagnosis by cell graphs. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 228–235.

[13] Chao Li, Chuqing Cao, and Yunfeng Gao. 2016. GM-Citation-KNN: Graph matching based multiple instance learning algorithm. In *Eighth International Conference on Digital Image Processing (ICDIP 2016)*, Vol. 10033. International Society for Optics and Photonics, 100334J.

[14] Fahira A Maken, Yaniv Gal, Darryl McClymont, and Andrew P Bradley. 2014. Multiple instance learning for breast cancer magnetic resonance imaging. In *Digital Image Computing: Techniques and Applications (DICTA), 2014 International Conference on*. IEEE, 1–8.

[15] Oded Maron and Tomás Lozano-Pérez. 1998. A framework for multiple-instance learning. In *Advances in neural information processing systems*. 570–576.

[16] Oded Maron and Aparna Lakshmi Ratan. 1998. Multiple-Instance Learning for Natural Scene Classification.. In *ICML*, Vol. 98. 341–349.

[17] Joseph F Murray, Gordon F Hughes, and Kenneth Kreutz-Delgado. 2005. Machine learning methods for predicting failures in hard drives: A multiple-instance application. *Journal of Machine Learning Research* 6, May (2005), 783–816.

[18] Dat T Nguyen, Cao D Nguyen, Rosalyn Hargraves, Lukasz A Kurgan, and Krzysztof J Cios. 2013. mi-DS: Multiple-instance learning algorithm. *IEEE transactions on cybernetics* 43, 1 (2013), 143–154.

[19] John C Platt. 1999. 12 fast training of support vector machines using sequential minimal optimization. *Advances in kernel methods* (1999), 185–208.

[20] Rouhollah Rahmani, Sally A Goldman, Hui Zhang, John Krettek, and Jason E Fritts. 2005. Localized content based image retrieval. In *Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*. ACM, 227–236.

[21] Soumya Ray and Mark Craven. 2005. Learning statistical models for annotating proteins with function information using biomedical text. *BMC bioinformatics* 6, 1 (2005), S18.

[22] Vikas C Raykar, Balaji Krishnapuram, Jinbo Bi, Murat Dundar, and R Bharat Rao. 2008. Bayesian multiple instance learning: automatic feature selection and inductive transfer. In *Proceedings of the 25th international conference on Machine learning*. ACM, 808–815.

[23] Qingping Tao, Stephen Scott, NV Vinodchandran, and Thomas Takeo Osugi. 2004. SVM-based generalized multiple-instance learning via approximate box counting. In *Proceedings of the twenty-first international conference on Machine learning*. ACM, 101.

[24] Sebastián Ubalde, Francisco Gómez-Fernández, Norberto A Goussies, and Marta Mejail. 2016. Skeleton-based action recognition using citation-knn on bags of time-stamped pose descriptors. In *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE, 3051–3055.

[25] Ranga Raju Vatsavai. 2013. Gaussian multiple instance learning approach for mapping the slums of the world using very high resolution imagery. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 1419–1426.

[26] Ranga Raju Vatsavai, Budhendra Bhaduri, and Jordan Graesser. 2013. Complex settlement pattern extraction with multi-instance learning. In *Urban Remote Sensing Event (JURSE), 2013 Joint*. IEEE, 246–249.

[27] Jun Wang and Jean-Daniel Zucker. 2000. Solving multiple-instance problem: A lazy learning approach. (2000).

[28] Jiajun Wu, Yinan Yu, Chang Huang, and Kai Yu. 2015. Deep multiple instance learning for image classification and auto-annotation. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, 3460–3469.

[29] Baohan Xu, Yanwei Fu, Yu-Gang Jiang, Boyang Li, and Leonid Sigal. 2016. Video emotion recognition with transferred deep feature encodings. In *Proceedings of the 2016 ACM on International Conference on Multimedia Retrieval*. ACM, 15–22.

[30] Cheng Yang and Tomas Lozano-Perez. 2000. Image database retrieval with multiple-instance learning techniques. In *Data Engineering, 2000. Proceedings. 16th International Conference on*. IEEE, 233–243.

[31] Cha Zhang, John C Platt, and Paul A Viola. 2006. Multiple instance boosting for object detection. In *Advances in neural information processing systems*. 1417–1424.

[32] Kaihua Zhang and Huihui Song. 2013. Real-time visual tracking via online weighted multiple instance learning. *Pattern Recognition* 46, 1 (2013), 397–411.

[33] Zhi-Hua Zhou. 2004. Multi-instance learning: A survey. *Department of Computer Science & Technology, Nanjing University, Tech. Rep* (2004).

[34] Zhi-Hua Zhou, Kai Jiang, and Ming Li. 2005. Multi-instance learning based web mining. *Applied Intelligence* 22, 2 (2005), 135–147.