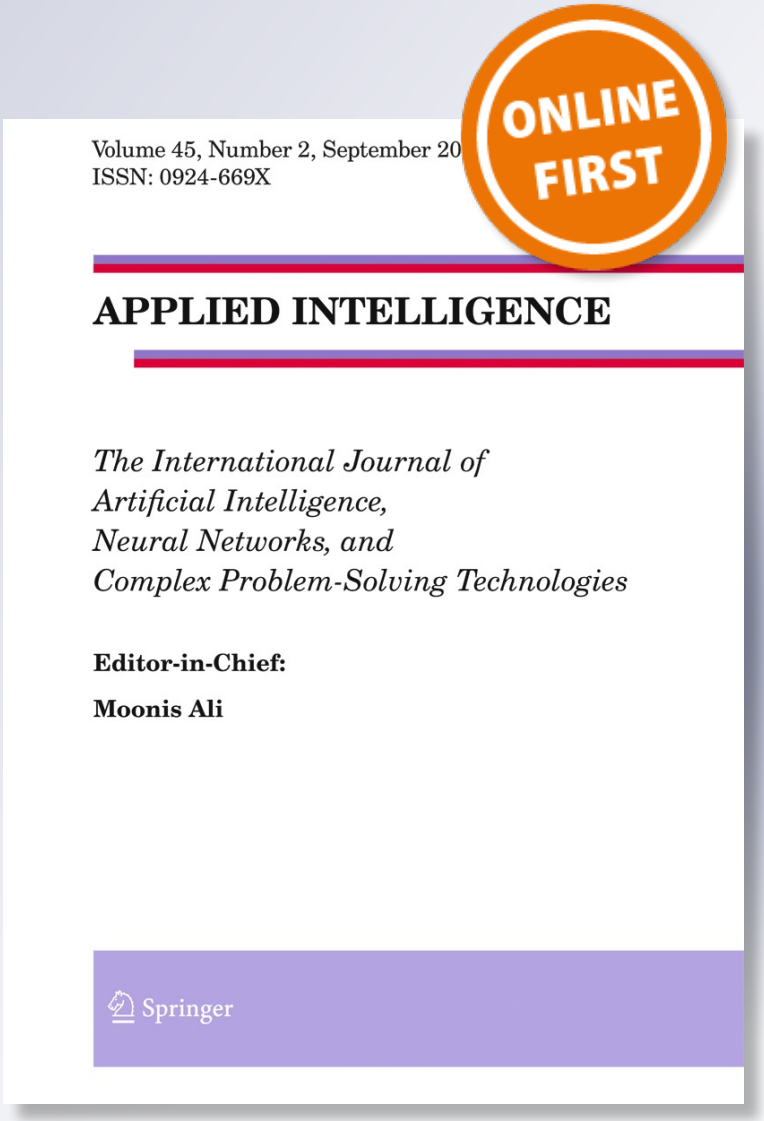
Learning weighted distance metric from group level information and its parallel implementation

Hamidreza Mohebbi, Yang Mu & Wei Ding

Applied Intelligence
The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies

ISSN 0924-669X

Appl Intell
DOI 10.1007/s10489-016-0826-7



Springer

Appl Intell
DOI 10.1007/s10489-016-0826-7

Learning weighted distance metric from group level information and its parallel implementation

Hamidreza Mohebbi[1] · **Yang Mu**[1] · **Wei Ding**[1]

Abstract The performance of many machine learning algorithms heavily relies on the distance metrics. Usually a distance metric is learned from a training set, while other valuable information, such as group structure, is not. Samples within a short distance form a group, which may contain several classes; each sample may have partial memberships to multiple groups. The group structure exists in both training and test sets. Additionally, outliers have negative effects on a distance metric. Increasing the number of noisy samples during the learning phase may increase the negative effects of outliers. Use of weights is one way to alleviate this problem when more similar samples are given more weight. This paper introduces a learning technique for weighted-distance metric. This semi-supervised method learns labeled information from training set and identifies groups among the samples from test set to form a metric space. In the experiments, the nearest neighbors algorithm is used as a classifier. The proposed weighted-distance metric improves the classification accuracy by more than 10 %. Furthermore, parallel computing with optimized CPU and GPU code is developed to speed up the computing time. Two parallel implementations with Matlab and CUDA are compared in this research. Parallel code that uses both CPU and the GPU achieves more than 3.7 times speedup compared to the traditional CPU code in the experiments.

Keywords Distance learning · Semi-supervised learning · Parallel computing · GPU accelaration

1 Introduction

Often samples of the same class have closer distances to each other than samples of other classes. This creates a group structure among the samples. Each sample contributes to a group based on its distance and its contribution, which is moderated as a weight. This group structure exists both in the training and test sets, the information derived from the group structure can improve the classification accuracy. This paper proposes a method to learn this group structure from both labeled training and unlabeled test sets. The improvements on classification accuracy are demonstrated with extensive experiments.

Machine learning algorithms often learn the distance metric from class-labeled information of training data only and miss the group-level information which may contain the group structure of the classes. Similar samples of a class create a group, and dissimilar ones belong to another group. A group can contain samples from more than one class, and each sample may contribute to more than one group. The proposed method assigns a weight to each sample to bring out the group and class structures. A sample may have different weights for different groups. A classifier may achieve higher accuracy by identifying groups of similar and dissimilar samples. This is because the performance of many classification and clustering algorithms, such as k-nearest-neighbors (KNN) and k-means, depends on distance metrics

✉ Hamidreza Mohebbi
Hamidreza.Mohebbi001@umb.edu

Yang Mu
yangmu@cs.umb.edu

Wei Ding
wei.ding@umb.edu

[1] Computer Science Department, University of Massachusetts Boston, Boston, MA 02125, USA

Published online: 03 August 2016

over input samples [24]. The classification may be more accurate if the distance metric expresses the statistical properties of input data more clearly. The proposed *weighted* distance metric (WDM) learns the metrics from both labeled training and unlabeled test data to create group structure, and then combines these two parts to project them into a new space where instances of the same class are brought together. This is the novelty and major difference between the WDM and other distance learning methods [2, 13, 14].

One way to search for groups in the samples is to limit search space to nearest neighbors. The proposed method looks among the nearest neighbors of each class to identify group structure. The sample with shortest distance is the most similar member of a class, and the one with the farthest distance among nearest neighbors often belongs to a different class. The ideal distance metric brings samples of a class together, and thus gives more weight to members of its own class than those of other classes. The WDM creates two weight vectors from training and test sets; these two vectors are then combined into one weight vector. Another advantage of a weighted-distance metric is that it reduces the sensitivity of KNN-based classifiers to the number of neighbors. It has been shown that the negative effect of scarcity and noise in data dramatically increases with a higher number of neighbors in KNN-based classifiers [6]. This article describes a new KNN-based classifier called KNN-WDM that uses the KNN classifier on the proposed weighted-distance metric. Fig. 1 shows an overview of the WDM distance learning process.

Time complexity of computing a distance metric among n samples is $O(n^2)$. Parallel computing divides tasks and executes them concurrently. Modern CPUs have several cores. GPU devices with thousands of cores can provide greater processing power than CPUs. However, GPUs may not be the best solution for all computations. One weakness of GPUs is the overhead of memory communication between the host CPU and the device. A GPU task with high communication overhead may be slower than good parallel implementation on a CPU. Our goal in this research is to integrate CPU and GPU computation to improve the overall execution time. Based on runtime profiling, we chose to do distance calculation on the GPU and rest of the computation on the CPU. Fig. 1 illustrates the parallel architecture of WDM.

Our contributions towards this research are:

– We introduce a weighted-distance metric that learns group information from both labeled training and unlabeled test sets. The technique reduces the effect of outliers and increases the classification performance.

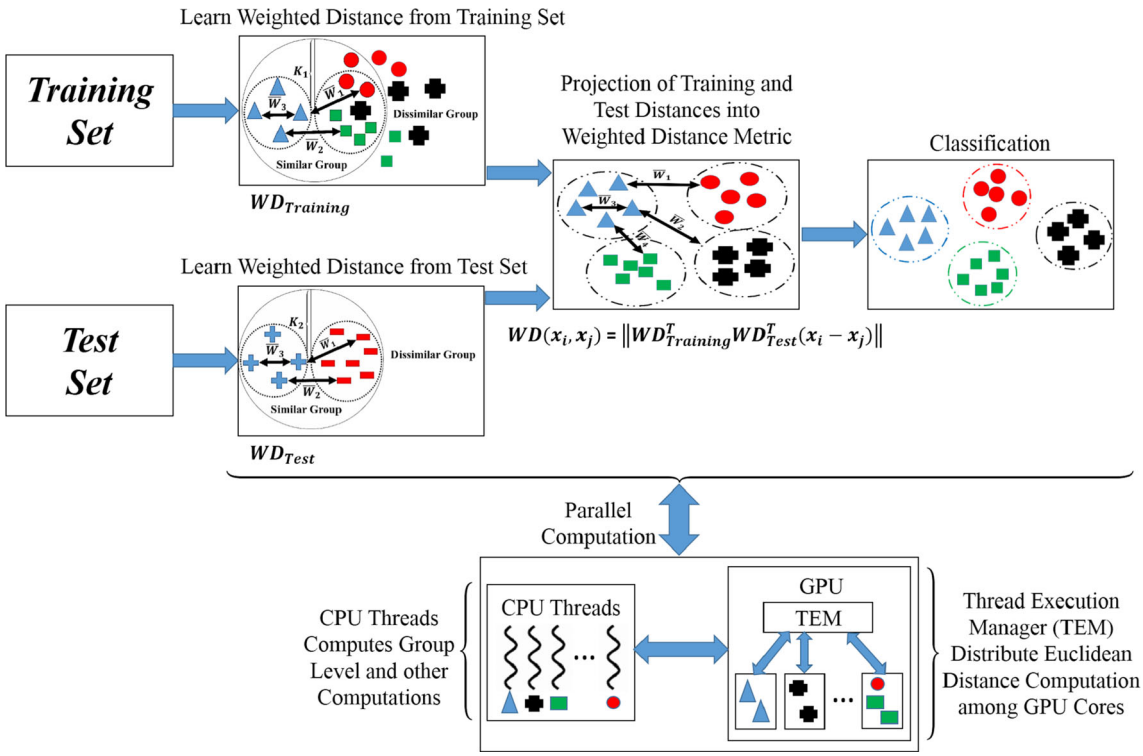


Fig. 1 Overview of the WDM architecture: The WDM learns a weighted-distance metric by combining class and group information from labeled training and unlabeled test data, respectively. $WD_{Training}$ and WD_{Test} are the weighted-distance metrics learned from the training and test sets. WD combines $WD_{Training}$ and WD_{Test}. Parallel computation by using both CPU and GPU greatly speeds up the computation

The maximum increase of classification accuracy is more than 10 %.
- We reduce the sensitivity of a KNN classifier to the number of neighbors.
- We improve the execution time by combining CPU and GPU parallel computations. The maximum, minimum, and average of speedup achieved by the parallel implementation of WDM is more than 3.7, 1.6, and 2.5 times than serial CPU code, respectively.

2 Related work

The related work of this study categorizes into two parts: distance metric learning, and the GPU acceleration of machine learning algorithms.

2.1 Distance metric learning

The distance metric learning category of related work divides into three subcategories. The first subcategory contains local classification methods like the SVM method. The second subcategory includes the algorithms that use similarity and dissimilarity constraints to build a global-general metric over all data set. The third subcategory comprises semi-supervised methods that use both labeled and unlabeled information from both training and test sets.

Many efforts [18, 20] have been made to define or learn either local or global metrics for supervised classification. While these methods often created good metrics for classification, it is less clear whether they can be used as general metrics for other algorithms such as K-means. Especially if the information is less structured than the traditional, homogeneous-labeled training sets expected by the algorithms. The WDM method addressed this problem by learning group-level information from both labeled training and unlabeled test sets. The distance metric produced by this method would be more general because it uses different data sources.

The Xings method [24] is one of the related approaches that tried to satisfy both similarity and dissimilarity constraints simultaneously by building constraints globally over the entire data set. The opposite of global are local constraint methods like large margin nearest neighbor (LMNN) [21] and local Fisher discriminant analysis (LFDA) [19]. Local constraint methods outperformed global approaches in multimodal distribution environments, because local methods utilized only neighborhood constraints rather than all constraints in the data set to learn the distance [12]. The WDM followed the local constraint approach.

The proposed method builds a weighted-distance metric from the labeled training set and combines it with group-level information from the unlabeled test set. This differs from the related approaches that use only labeled training data in order to build the model. The WDM is a semi-supervised approach, similar to Semi-supervised Discriminant Analysis (SDA) [2] and Bipart [14] that used labeled and unlabeled data. The difference between WDM and methods like SDA is that it learns group-level information from both training and test sets. The proposed method combines the information from these two sources with a projection matrix. In addition, similar and dissimilar samples contribute to this process based on their similarity and dissimilarity to the query sample. In order to reach this goal, WDM assigns a weight to each sample.

Table 1 compares WDM method with the other state-of-the-art algorithms using criteria of Category, Detect Group Structure, Sensitivity to Outliers, and Weight Learning features. Category indicates whether the algorithms use unsupervised, supervised, and semi-supervised approaches. Similar classes create a group and WDM detects this structure. Sensitivity to Outliers represents the relationship between the accuracy of a classifier and search space. Weight Learning indicates whether the methods use weighted samples when computing distance metrics. From each category of the related methods, the most similar method to WDM selected for this comparison. The list includes Weighted-KNN because WDM uses a KNN classifier in this study. SVM, LMNN, and Bipart methods represent the choice for the first, second, and third groups of the related methods, respectively. This table shows the novelties of the proposed method, including detecting group

Table 1 The feature comparison of the wdm method and the related approaches

Distance Metric Algorithm	Category	Detect Group Structure	Sensitivity To Outliers	Weight Learning
WDM	Semi-Supervised	✓	Low	✓
Weighted-KNN	Supervised	×	Low	✓
SVM	Supervised	×	High	×
LMNN	Supervised	×	High	×
Bipart	Semi-Supervised	×	High	×

structure, reducing the sensitivity to outliers, and learning weights based on the similarity and dissimilarity of constraints.

2.2 GPU acceleration of machine learning algorithms

Data scientists in industry and academia have used GPUs for machine learning across a variety of applications including image classification, video analytics, speech recognition, and natural language processing. Researchers with sophisticated, multi-level, deep neural networks have performed feature detection from massive amounts of training data [16]. The communication overhead between components has been one of the challenges of a distributed learning system. Coates *et al.* [4] proposed an approach with high speed communication infrastructure along with GPU processing power to scale up a deep learning algorithm.

Many libraries and frameworks such as cuBLAS [5] and Matlab Parallel Computing Toolbox [11] used GPU for high performance computing. These libraries used optimized parallel matrix operations to provide good performance. Another approach is to implement specific operations directly with CUDA [15], which is the hardest method in terms of implementation and debugging. In order to achieve high performance in the WDM, the bottlenecks in the code identified and vectorization techniques applied to reduce the number of instructions in the code.

Table 2 Definition of symbols in the manuscript

Notation	Description
X	Data set of n samples with d dimensions
x_i	Data sample i
Y	Labels for each element in X
y_i	Label for data sample x_i
G_i	i-th group of samples, $\{x_1^{G_i}, \ldots, x_{k_i}^{G_i}\}$
y^{G_i}	Label of group G_i
k_i	Number of elements in group G_i
A	The unified objective distance metric
A_1	The unified objective distance metric from training set
A_2	The unified objective distance metric from test set
d_A	Distance metric defined by matrix A
L	Alignment matrix
W	Weighted distance metric learned from both training and test sets
W_1	Weighted distance metric learned from training set
W_2	Weighted distance metric learned from test set
w_t	Weight of t-th nearest neighbor of sample x_i
G_i^s	Group nearest G_i with same class label
G_i^d	Group nearest G_i with different class label
k_i^s	Number of elements in group G_i^s with same class label
k_i^d	Number of elements in group G_i^d with different class label
$x_p^{G_i^s}$	p-th element from group G_i^s
$x_q^{G_i^d}$	q-th element from group G_i^d
w_s	Weight vector for group G_i^s with same class label
w_d	Weight vector for group G_i^d with different class label
w_i	Weight vector for sample x_i learned from both training and test sets
β	Balancing parameter for training and test metrics
n_1	number of samples in test set

3 The WDM method

This section explains the weighted-distance metric learning and its parallel implementation. Table 2 shows the definition of symbols in this paper.

3.1 Weighted distance metric learning

The WDM method is a semi-supervised learning approach that learns distance metric from both labeled and unlabeled data. There are two components of the WDM metric. One metric is learned from the similar and dissimilar groups of the training set, and the other is learned from those of the test set. These two are combined into a single metric, which the data is projected for classification. Some may think that using unlabeled information from the test set is cheating. However, its been shown that the use of unlabeled data in conjunction with labeled information can produce considerable improvements in learning accuracy. The semi-supervised learning is useful when there are relatively few labeled points and a large number of unlabeled points. It is directly relevant to a lot of practical problems where it is expensive to produce labeled data, e.g., the automatic classification of web pages [3]. This class of learning methods falls between unsupervised (without any labeled training data) and supervised learning (with completely labeled training data).

The major difference between the approach of this paper and typical classification is in using group structure in classification. Let a data set be $(X, Y) = \{(x_1, y_1), \ldots, (x_i, y_i), \ldots, (x_n, y_n)\}$, where $x_i \in \mathbb{R}^n$, and y_i is the label of x_i. The goal is to generate a model to classify unknown samples. Instead of classifying unknown samples directly, groups are first identified as $G_i = \{x_1^{G_i}, \ldots, x_{k_i}^{G_i}\}$, where k_i is the number of samples in group G_i. The corresponding label y^{G_i} of G_i is defined as $y^{G_i} = y_1^{G_i} =, \ldots, = y_{k_i}^{G_i}$.

Consider the task of predicting the label of query sample x_i. The Mahalanobis distance (d_A) between two

samples (x_i, x_j) can be defined by (1), where A is positive semi-definite. Note that when A is the identity matrix, this simplifies to Euclidean distance.

$$d_A(x_i, x_j) = \sqrt{(x_i - x_j)^T A (x_i - x_j)} \tag{1}$$

Equation (1) expressed as (2) with the Cholesky decomposition [23]. This replaces A with WW^T, where W is the weighted-distance metric.

$$d_A(x_i, x_j) = \sqrt{(x_i - x_j)^T WW^T (x_i - x_j)} = \|W^T (x_i - x_j)\| \tag{2}$$

The WDM method learned distance metric from unlabeled test W_1 and the labeled training W_2 sets by projecting to W_1, then to W_2, as in (3).

$$\begin{aligned} d_A(x_i, x_j) &= \sqrt{(x_i - x_j)^T W_2 W_2{}^T W_1 W_1{}^T (x_i - x_j)} \\ &= \|W_2{}^T W_1{}^T (x_i - x_j)\| \end{aligned} \tag{3}$$

In order to learn the projection matrices W_1 and W_2, it is necessary to find a metric space that keeps similar samples in a group and dissimilar ones in another group. It means that, for samples in a group, the distances among samples of the same class should be minimized and those among samples of different classes should be maximized.

In the KNN-based classifiers, when the size of training set approaches infinity, the error rate approximates the optimal Bayesian error rate [6]. In other words, the accuracy of the KNN classifier decreases with increasing values of k. Also, to find the right value of k is one of the challenges in KNN, which can have a significant impact on the performance of KNN-based classifiers. If k is very small, the local estimate tends to be very poor for sparse and noisy data. If k is very large, the accuracy of classification decreases due to outliers from other classes [25]. The idea behind the WDM method is to reduces the sensitivity of KNN-based classifier to k and improves the accuracy of the classification. In the WDM method a weight assigns to each sample, which shows the level of similarity or dissimilarity of the sample in the group. The WDM creates a more general distance metric with higher values of k.

The first way to assign the weights is to normalize the distance between the nearest and farthest elements. (4) shows this method. The nearest element gets weight of one, and the farthest element gets weight of zero. The weight of other elements are scaled linearly by their distances. In (4), w_t is the weight of t-th nearest neighbor of query x_i, and the k is the number of neighbors in the KNN algorithm.

$$w_t = \begin{cases} \frac{d(x_i,x_k) - d(x_i,x_t)}{d(x_i,x_k) - d(x_i,x_1)} & \text{if } d(x_i, x_k) \neq d(x_i, x_1) \\ 1 & \text{if } d(x_i, x_k) = d(x_i, x_1) \end{cases} \tag{4}$$

With the weight vector $w = \{w_1, \ldots, w_k\}$, the KNN becomes less sensitive to the choices of k than the related methods. However, it is still not robust to large values of k, and it still suffers from the issue of outliers. The WDM uses a dual weight approach to make it more robust to the change of k. (5) represents normalization of the weight in (4). With this change the weight of each nearest neighbor reduces, except for the first nearest and the k-th nearest neighbor. It avoids giving too much weight to the outliers, and therefore increases classification accuracy.

$$w_t = \begin{cases} \frac{d(x_i,x_k) - d(x_i,x_t)}{d(x_i,x_k) - d(x_i,x_1)} \times \frac{d(x_i,x_k) + d(x_i,x_1)}{d(x_i,x_k) + d(x_i,x_t)} & \text{if } d(x_i, x_k) \neq d(x_i, x_1) \\ 1 & \text{if } d(x_i, x_k) = d(x_i, x_1) \end{cases} \tag{5}$$

Previous studies [19, 21] showed that building local constraints from neighbors leads to higher accuracy than the global constraint method in the multimodal distributions. For any sample x_i in group G_i^s, similarity constraints are formed from other elements in the same group. In the same way the dissimilarity constraints are built from groups with different class labels denoted as G_i^d. Based on (5), the WDM method uses a dual weighted-distance metric, which creates a distance vector for similar and dissimilar classes. The weights are between one and zero such that the nearest element has weight one, and the farthest element has a weight close to zero.

The distance metric is more general when more samples are incorporated. In this study the value for the number of neighbors is twice $(2k)$ of Bipart (k). In the WDM, the nearest k samples are close to the sample point that they are good enough with existing weights. However, the farther k samples have longer distances, and their weights are adjusted according to (5).

Let n_1 be the number of elements in training set for sample x_i, and let k_i^s be the number of elements in the group G_i^s. Let k_i^s be vectors in the nearest group with the same class label $x_p^{G_i^s} \in G_i^s$. Let $w^s = [w_1^s, \ldots, w_{k_i^s}^s]$ be the weight vector of the same class and k_i^d be the number of elements in dissimilar group (G_i^d). The following shows the derivation of W_1. Similar derivation works for W_2. (6) minimizes the similarity constraints, where d_{A_1} is Euclidean distance from the training set.

$$argmin_{A_1} \sum_{i=1}^{n_1} \sum_{p=1}^{k_i^s} w_p^s . d_{A_1}^2 \left(x_i, x_p^{G_i^s}\right) \tag{6}$$

In the same way, (7) maximizes dissimilarity constraints.

$$argmax_{A_1} \sum_{i=1}^{n_1} \sum_{q=1}^{k_i^d} w_q^d . d_{A_1}^2 \left(x_i, x_q^{G_i^d}\right) \tag{7}$$

Equations (6) and (7) combine into one equation using the scaling parameter β:

$$argmin_{A_1} \sum_{i=1}^{n_1} \left(\sum_{p=1}^{k_i^s} w_p^s . d_{A_1}^2 \left(x_i, x_p^{G_i^s}\right) - \beta \sum_{q=1}^{k_i^d} w_q^d . d_{A_1}^2 \left(x_i, x_q^{G_i^d}\right) \right) \tag{8}$$

In the above equations, $A_1 = W_1^T W_1$ is the distance metric learned from the training set. Similar equations are obtained for the labeled test set $A_2 = W_2^T W_2$. The weights vector with the balancing parameter defined as this:

$$w_i = \left[w_1^s, \ldots, w_{k_i^s}^s, -\beta.w_1^d, \ldots, -\beta.w_{k_i^d}^d \right] \tag{9}$$

Equation 8 is rewritten below where the i-th row of matrix X is $X_i = [x_i, G_i^s, G_i^d] = [x_i, x_1^{G_i^s}, \ldots, x_{k_i^s}^{G_i^s}, x_1^{G_i^d}, \ldots, x_{k_i^d}^{G_i^d}]$.

$$\begin{aligned} & argmin_{A_1} \sum_{i=1}^{n_1} \left(\sum_{j=1}^{k_i^s + k_i^d} d_{A_1}^2 \left(X_i\{1\}, X_i\{j+1\} \right) (w_i)_j \right) \\ & = argmin_{A_1} \sum_{i=1}^{n_1} \left(\sum_{j=1}^{k_i^s + k_i^d} \| W_1 \left(X_i\{1\} - X_i\{j+1\} \right) \|_2^2 (w_i)_j \right) \\ & = argmin_{A_1} \sum_{i=1}^{n_1} tr \left(W_1^T X_i L_i X_i^T W_1 \right) \end{aligned} \tag{10}$$

where $X_i\{j\}$ is the j-th column of matrix X_i, $(w_i)_j$ is the j-th element of w_i, and L is the alignment matrix computed by following equation:

$$L = \sum_{i=1}^{n_1} L(X_i, X_i) + L_i \tag{11}$$

The L_i matrix is defined by (12) such that $L_i \in \mathbb{R}^{(k_i^s + k_i^d + 1) \times (k_i^s + k_i^d + 1)}$.

$$L_i = \begin{bmatrix} \sum_{j=1}^{k_i^s + k_i^d} (w_i)_j & -w_i^T \\ -w_i & diag(w_i) \end{bmatrix} \tag{12}$$

3.2 Parallel computation

Several ways explored to parallelize the computation, including writing CUDA [15] code directly, calling CUDA libraries such as cuBLAS [5], and using other tools such as Matlab Parallel Computing Toolbox [11]. Writing CUDA code is the most involved, because programmers need to understand the hardware architecture. For instance, GPU devices handle the floating point calculations differently from CPUs, which raise some issues like accuracy and precision [22]. However, efficient CUDA code can provide good performance. Many matrix operations implemented and are available in CUDA libraries. Using existing library routines saves time in code development. However, programmers need to integrate the CUDA code with other programming languages and development tools, which reduce readability and maintainability due to heterogeneous code. Matlab [10] is a solution to these issues, and its Parallel Computing Toolbox delivers high performance along with simplicity, ease of debugging, maintainability, and readability. Each of these methods have its own advantages and disadvantages. In this experiment two versions of the parallel code with CUDA and Matlab Parallel Computing Toolbox are implemented and compared.

The first step to create parallel code from a serial one is to identify the parts of the code that take the most of computation time. These parts are bottlenecks in achieving faster execution. One of the bottlenecks is the loop. There are two big loops in the WDM method. The first one is based on (10) that calculates the weighted-distance metric. The algorithm (1) shows the pseudo code for (10). The second one runs the KNN classifier on sample data (3) – algorithm (2) shows the pseudo code. The main operation on the KNN classifier is the Euclidean distance calculation between the samples, and it is the most time consuming part of the calculation in (3). An explicit Matlab loop can be removed by apply a function to each element of an array (`arrayfun`), which uses a fast, implicit loop to run the elements of the array through the function. This mechanism applied to the algorithms (1) and (2) and the optimized algorithms are shown by algorithms (3) and (4), respectively. Using `arrayfun` has significant impact on the total execution time. Other improvements like the use of vectorized operations and the removal of the `if-else` statements also lead to faster computation.

The next step is to convert the efficient CPU code into the parallel version on the GPU platform. This step is implemented with Matlab Parallel Computing Toolbox and CUDA code.

The Matlab Parallel Computing Toolbox provides options for element-wise `arrayfun` functions on GPU. Each element of the array is processed by a GPU thread. The ideal parallel solution is to run `arrayfun` of (3) and (10) on the GPU. However the `arrayfun` method has some limitations; for instance all the input elements must have same size and same type [17]. But (3) and (10) have inputs

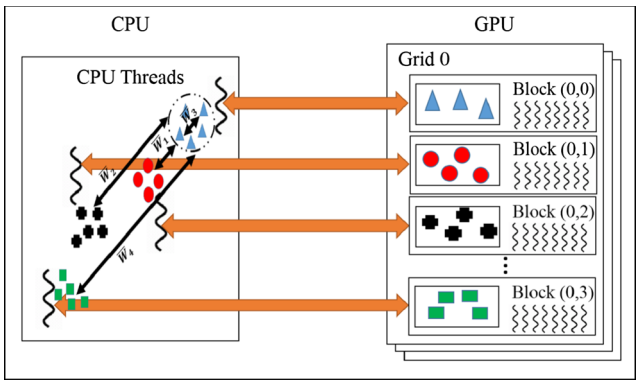


Fig. 2 Overview of the parallel implementation of WDM. The CPU threads send each part of the job to a group of the GPU cores, which organized as a block in the GPU. Each block calculates a part of output matrix. The blocks in the GPU are organized as a grid

Algorithm 1 Weighted Distance Metric Calculation

Input: fea and gnd are features and ground truth with n records
Output: A weighted-distance metric Output

$X = \text{fea}^T$;
for $i = 1 : n$ **do**

$$w_i = \left[w_1^s, \ldots, w_{k_i^s}^s, -\beta.w_1^d, \ldots, -\beta.w_{k_i^s}^d \right];$$

$$L_i = \begin{bmatrix} \sum_{j=1}^{k_i^s + k_i^d} (w_i)_j & -w_i^T \\ -w_i & diag(w_i) \end{bmatrix};$$

$$L = \sum_{i=1}^{n_1} L(X_i, X_i) + L_i;$$

$$\text{Output(i)} = argmin_w tr\left(W^T X L X^T W\right);$$

end
return Output;

Algorithm 2 The KNN Classifier

Input: fea , gnd and W are features , ground truth and weighted-distance metric with n records
Output: Predicted ground truth Output

for $i = 1 : n$ **do**

$$W_1 = \text{fea} * W(:, 1:i); \quad W_2 = \text{gnd} * W(:, 1:i);$$

$$[\text{Output(:,i)}] = \sqrt{(x_i - x_j)^T W_2 W_2{}^T W_1 W_1{}^T (x_i - x_j)} = \|W_2{}^T W_1{}^T (x_i - x_j)\|;$$

end
return Output;

Algorithm 3 Optimized Weighted-Distance Metric Calculation

Input: fea and gnd are features and ground truth with n records
Output: A weighted-distance metric Output

Initializing the handle object h = handle;
h.fea = fea;
h.gnd = gnd;
h.Output = zeros(n);
$i = [1..n]^T$;
arrayfun(@weighted_distance_calculations,h,i);
return h.Output;

Algorithm 4 Optimized KNN Classifier

Input: fea , gnd and W are features , ground truth and weighted-distance metric with n records
Output: Predicted ground truth Output

```
Initializing the handle object h = handle;
h.fea = fea;
h.gnd = gnd;
h.W = W;
h.Output = zeros(n);
i = [1..n]^T;
arrayfun(@KNN_classifier,h,i);
return Output;
```

with different sizes. There are two solutions for this problem. The first solution is the use of global variables and the second, includes the handle class which provides a shared reference to the object. However, Matlab does not send shared variables or references to the GPU [17], because of consistency issues among copies of shared variables and references. Thus for this study, the ideal solution is not feasible with the Matlab Parallel Computing Toolbox. The next best solution is a hybrid approach that uses optimized CPU code and GPU together. Fig. 2 shows an overview of this parallel approach. (3) and (10) are implemented by optimized CPU code and the Euclidean distance calculation in (3) is executed on the GPU. In other words, a group of CPU threads are running the code, and each CPU thread sends a job to the GPU for Euclidean distance calculation. In the GPU, a group of threads called a block is responsible for executing a job, and blocks are grouped into a grid. Threads of a single block will be executed on a multiprocessor. Threads are synchronized and they use shared data cache in a block. A function is executed on the GPU as a grid of blocks of threads. The GPU devices run different functions as multiple grids simultaneously. Fig. 2 represents thread, block and grid organization in the GPU.

Many general functions are implemented as part of Matlab Parallel Computing Toolbox and there are built-in functions used for general matrix operations. These functions are used in the GPU implementation of the Euclidean distance calculation of (3). In particular, the `gpuArray` function creates an array data structure in the GPU memory. Each index of the array represents a part of the data in the host memory. After array creation, the element-wise operations are executed on the GPU. It means that each GPU core is responsible for providing the results for a chunk of elements in the array. Another function named `gather` retrieves a `gpuArray` from the GPU memory into the Matlab workspace [11].

The CUDA implementation follows the structure of the hybrid approach which is explained in Fig. 2. Parts of the code that can not be implemented efficiently with CPU or parallel Matlab code are implemented by the CUDA. The weighted-distance metric calculation in algorithm (1) is selected for CUDA implementation. Based on runtime profiling, this part is one of the time consuming parts of the code. Algorithm (5) represents the CUDA pseudo code for weighted-distance metric. The CUDA code calculates the L matrix represented by (11). In this code, each thread is responsible for reading one record of data from the Euclidean distance matrix (D) and ground truth (gnd) inputs. Each thread fills a part of output (L) matrix. The L matrix is the projection of similar and dissimilar samples to the i-th sample. The similar and dissimilar samples are different for each sample. Therefore each thread writes to different places of the output matrix. Then the memory access pattern is irregular. This irregular access slows down the computation. In general, GPU devices are good in computation but weak in memory access.

Data transfer time is another consideration in using GPU. The total time is the sum of the execution time on the GPU plus the transfer time of the data and results between the host memory and the GPU memory. When the size of data is small, the CPU is faster than the GPU, because of the communication overhead for GPU. However, when the data is big enough, the total time on GPU is less than CPU computation time. Based on this, the size of intermediate variables must taken into the account, and only large chunks of data should sent to the GPU for processing.

4 Distance metric evaluation

Four data sets are used for distance metric evaluation. The criteria are accuracy, precision, and recall. The first data set

Algorithm 5 The CUDA Algorithm For Weighted-Distance Metric Calculation

Input: D and gnd are the Euclidean distance and ground truth with n records
Output: A projection matrix Output

1 i = thread_index;

2
$$D^{index} = \left[D_1^s, \ldots, D_{k_i^s}, D_1^d, \ldots, D_{k_i^s}^d \right];$$

3
$$w_i = \left[w_1^s, \ldots, w_{k_i^s}, -\beta.w_1^d, \ldots, -\beta.w_{k_i^s}^d \right];$$

4
$$L_i = \begin{bmatrix} \sum_{j=1}^{k_i^s+k_i^d} (w_i)_j & -w_i^T \\ -w_i & diag(w_i) \end{bmatrix};$$

5
$$\text{Output(i)} = \sum_{i=1}^{k_i^s+k_i^d} Output(D_i^{index}, D_i^{index}) + L_i;$$

6 **return** Output;

is the Diffuse Large B-cell lymphoma (DLBCL) [9]. The DLBCL data set is classified into breast cancer subtypes. The task is to identify the common subtypes in two independent data sets, which makes this data set good candidate for evaluation of the WDM method. The data set contains genomic expression profiles of 129 patients. The data was generated with an one-channel oligonucleotide microarray and 2-channel custom cDNA microarrays. There are four category of breast cancer subtypes, D 1, D 2, D 3, and D 4. Each patient record has 3,795 features. In this evaluation, the number of features is reduced to 400 with the PCA method.

The second data set is the Madelon [7]. This is an artificial data set from the NIPS 2003 feature selection challenge. This is a two-class classification problem with continuous input variables. It has 4,400 instances, and the number of features is 500, which is reduced to 10 with the PCA approach. The difficulty in classifying this data set is that the problem is multivariate and highly non-linear.

The third data set is the Heart disease (Hungary) [1]. It contains the information of 294 patients, and the classes refer to the presence of heart disease in the patient. Class 0 means the patient has no heart disease, and Classes 1 to 4 identify different heart diseases. The original data set contains 76 features, but a subset of 14 features selected for evaluation of the WDM.

The fourth data set is the Mice Protein Expression [8]. The data set consists of the expression levels of 77 proteins/protein modifications that produced detectable signals in the nuclear fraction of cortex. There are 38 control mice and 34 trisomic mice (Down syndrome), for a total of 72 mice. In the experiments, 15 measurements were registered of each protein per sample/mouse. Therefore, for control mice, there are $38 \times 15 = 570$ measurements, and for trisomic mice, there are $34 \times 15 = 510$ measurements. The data set contains a total of 1,080 measurements per protein. Each measurement can be considered as an independent sample/mouse. The eight classes of mice described based on features such as genotype, behavior and treatment. According to genotype, mice can be control or trisomic. Based on behavior, some mice have been stimulated to learn (context-shock) and others have not (shock-context). In order to assess the effect of the drug memantine in recovering the ability to learn in trisomic mice, some mice have been injected with the drug and others have not. The aim is to identify subsets of proteins that are discriminant between the classes. The Classes of Mice Protein Expression data set are:

- c-CS-s: control mice, stimulated to learn, injected with saline (9 mice).
- c-CS-m: control mice, stimulated to learn, injected with memantine (10 mice).

- c-SC-s: control mice, not stimulated to learn, injected with saline (9 mice).
- c-SC-m: control mice, not stimulated to learn, injected with memantine (10 mice).
- t-CS-s: trisomy mice, stimulated to learn, injected with saline (7 mice).
- t-CS-m: trisomy mice, stimulated to learn, injected with memantine (9 mice).
- t-SC-s: trisomy mice, not stimulated to learn, injected with saline (9 mice).
- t-SC-m: trisomy mice, not stimulated to learn, injected with memantine (9 mice).

The results of the WDM method are compared with five related distance metric methods. These methods includes the related approaches in Table 1. The classic KNN and Weighted KNN classifiers are selected to compare the proposed distance metric with Euclidean and weighted Euclidean distances, respectively. The weights of Weighted KNN classifier have inverse relationship with the square of distance. In what follows, KNN stands for the classic KNN, Weighted KNN stands for KNN classifier with weights for distance, Bipart stands for the Bipart metric in conjunction with KNN, and KNN-WDM for KNN with the WDM metric.

KNN-WDM and these other methods are compared based on accuracy, precision, and recall. Accuracy is the ratio of correct classifications over the total number of samples. The reported accuracy is the mean of accuracies from 10 independent runs. Precision is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved. Recall is the ratio of the number of relevant records retrieved to the number of relevant records. 10-fold cross validation (CV) and random splitting (RS) are used to prevent overfitting. $RS\_1, \ldots, RS\_9$ represent splitting the data into labeled training and test sets incrementally from 10 to 90 percent, respectively.

The performance of parallel and GPU implementations are compared to the serial implementation. The input for this evaluation is random data, and runtime is measured in minutes. Different numbers of records in the input data are used to compare performance. The KNN-WDM and other methods are implemented with the 64-bit Matlab R2011b installed on CentOS v6.6. The server has two 3.5 GHz Intel Xeon E5-2643 v2 CPUs with 128 GB RAM and 4.0 TB hard drive. The GPU device used in this study is Nvidia Tesla K40 with 2,880 CUDA cores and 12 GB RAM.

4.1 Classification results

In this section, the classification results of KNN-WDM are compared with the accuracy of related classifiers for DLBCL, Madelon, Heart disease, and Mice Protein Expression data sets.

Table 3 shows the accuracy of multiple classification for the DLBCL data set. The KNN-WDM method is more accurate than other methods in most of the cases for the DLBCL data set. The best results are obtained by using RS_5, RS_6, RS_7, RS_8, and CV methods. It is observed that when the size of the training set is relatively small to the test set ($RS\_1, \ldots, RS\_4$), SVM performs better than other approaches. One explanation to this behavior of WDM is that when the size of training set is relatively small to the size of test set, the algorithm can not detect the group structures properly. This is because WDM at first builds the groups structure based on the training set and then uses the test set to make the group structure clearer. This also explains the increase in accuracy of KNN-WDM when the size of training set is greater than or equal to the size of test set. Other factors such as use of weights and distinguishing similar and dissimilar groups also affect accuracy of KNN-WDM.

Table 4 shows the results of the precision evaluation for the DLBCL data set. Precision is measured with cross validation and it reveals that KNN-WDM can distinguish more relevant samples to all the records for $D2$ and $D4$ subtypes than the other methods. Precision of SVM is higher than the other methods for $D1$ and $D3$ subtypes. Table 5 represents the values of recall for cross

Table 3 The multiple classification accuracy (%) comparison of knn-wdm method and the related classifiers fOR dlbcl data set

	SVM	LMNN	KNN	Weighted KNN	Bipart	KNN-WDM
CV	73.7973	73.6927	64.0852	71.0903	76.8263	78.3746
RS_1	53.4483	44.3499	49.569	36.8966	45.9483	46.8966
RS_2	64.0777	59.6511	52.3301	45.9223	59.6117	60.2913
RS_3	71.1111	62.4444	58.3333	52.4444	66.6667	65.4444
RS_4	71.4286	68.7012	62.4675	61.1688	70.6494	71.039
RS_5	73.4375	72.0312	63.5938	58.9063	71.5625	73.5938
RS_6	69.2308	72.5	58.6538	63.6538	74.8077	82.3077
RS_7	69.2308	72.8205	60.5128	63.0769	76.4103	82.3077
RS_8	73.0769	69.6153	63.0769	63.0769	76.5385	77.3077
RS_9	76.9231	66.1538	62.3077	70.7692	78.4615	76.9231

Table 4 The precision (%) comparison of knn-wdm method and the related classifiers fOR dlbcl data set. the precision measured fOR each category and only cv experiments were considered

SubType	SVM	LMNN	KNN	Weighted KNN	Bipart	KNN-WDM
$D1$	**63.3333**	30.6968	22.9147	22.5556	40.509	43.5238
$D2$	60.45	53.5107	41.729	50.9532	59.219	61.608
$D3$	64.4167	38.3	22.46	28.231	45.454	50.5
$D4$	64.1242	61.229	52.615	58.3998	66.696	67.750

validation. It shows that KNN-WDM can predict more relevant records for $D1$ and $D4$ subtypes than other methods, and for $D2$ and $D3$ subtypes, it is not far behind Bipart. The main reason for these results is the use of weights and information in similar and dissimilar groups in KNN-WDM.

The same evaluations are repeated for Madelon data set. Table 6 contains the accuracy evaluation of this experiment. KNN-WDM has higher accuracy than the related methods for cross validation and RS_2, RS_4, RS_5, RS_6, RS_8, RS_9. Bipart is the second best method. This shows the importance of using information in the test set in learning distance metric. Also, Weighted KNN is more accurate than KNN; this shows the use of weights is effective in this data set. KNN-WDM uses weights with a semi-supervised approach and therefore can achieve higher classification accuracy than other approaches. Another point in the results of both Madelon and DLBCL data sets is that the accuracy of KNN-WDM increases with sizes of the training sets.

The Madelon data set contains two classes and Table 7 shows the precision of the related methods by cross validation. KNN-WDM can distinguish more relevant samples to all records for class 1. KNN has the highest precision for class 2. Table 8 compares recall of the proposed method with other approaches. KNN and KNN-WDM achieve the highest recall for class 1 and 2, respectively. However, in KNN the difference between the values of precision and recall for class 1 and 2 are high – this reflects the unbalanced classification between two classes. In the Madelon data set the number of elements in class 1 is equal to class 2. Therefore, precision and recall for two classes should be close. The precision and recall values for class 1 and 2 is close together for proposed method and this shows it provides a more balanced and clear classification compare to other approaches. One explanation to this, is the use of weights and learn group structure to distinguish similar and dissimilar samples more clearly.

Table 9 compares accuracy of KNN-WDM with the related approaches for the Heart disease data set. The accuracy of KNN-WDM is higher than other methods for cross validation. Table 10 and 11 contain the precision and recall measures for KNN-WDM and the related methods for 10-CV. The precision and recall of KNN-WDM are higher than other methods for class 0 and 3.

Comparison of accuracy for the Mice Protein Expression data set shown by Table 12. Bipart and KNN-WDM have higher accuracy than other methods. This is because both of these methods are semi-supervised approaches. However, KNN-WDM detects group structure and uses more samples with appropriate weights. The highest accuracy achieved by KNN-WDM for CV, RS_1, RS_2, RS_3, RS_7, RS_8 and RS_9 splitting approaches. The same pattern is observed for precision (Table 13) and recall (Table 14). Precision of KNN-WDM is the highest for $c-CS-s$, $c-CS-m$, $t-CS-s$ and $t-CS-m$ classes. Bipart has the highest precision for other classes. KNN-WDM has the highest recall for all classes except the $c-SC-m$ class, for which Weighted KNN has the highest recall. It observed that the accuracy, precision and recall measures for all the methods are close together.

4.2 Sensitivity to the number of neighbors

This section compares the accuracy of the KNN-WDM with Bipart according to the number of neighbors for DLBCL, Madelon, Heart disease and Mice Protein Expression data sets. One of the goals of this study is to, by using weights; reduce the sensitivity of the Bipart method to the number of neighbors. There are two numbers of neighbors, one for

Table 5 The recall (%) comparison of knn-wdm method and the related classifiers fOR dlbcl data set. the recall measured fOR each category and only cv experiments were considered

SubType	SVM	LMNN	KNN	Weighted KNN	Bipart	KNN-WDM
$D1$	30.3812	60.5	58.5	42.5	62.5	70.5
$D2$	49.9024	68.8333	52.25	79.9167	74.5	73.583
$D3$	33.6786	55.6667	39.5	39.6667	67.833	65.833
$D4$	79.094	90.4	86.1	89.6	88.1	91.05

Table 6 The multiple classification accuracy (%) comparison of knn-wdm method and the related classifiers fOR madelon data set

	SVM	LMNN	KNN	Weighted KNN	Bipart	KNN-WDM
CV	50.0833	73.35	70.0833	74.3167	68.5167	75.2333
*RS*_1	48.3333	67.1667	67.6667	76.5	68.8333	73.833
*RS*_2	53.3333	70.8333	73.0833	74.8333	71.75	78
*RS*_3	47.2222	71.7778	76.4444	76.1111	80.3333	79.889
*RS*_4	49.5833	74.25	76.2083	78.5	79.5833	80.042
*RS*_5	49.3333	74.7667	77.1667	79.9333	82.3	83.033
*RS*_6	49.4444	72.5833	77.75	79.8889	83.1111	83.167
*RS*_7	50.2381	73.1905	78.6905	79.4524	83.381	83.357
*RS*_8	50.4167	73.2708	79.5833	80	83.0208	83.313
*RS*_9	50.7407	73.8148	79.9444	79.5	83.5926	83.704

Table 7 The precision (%) comparison of knn-wdm method and the related classifiers fOR madelon data set. the precision measured fOR each category and only cv experiments were considered

Class	SVM	LMNN	KNN	Weighted KNN	Bipart	KNN-WDM
1	22.7614	72.9767	64.8119	73.1804	68.0489	74.0317
2	30.5498	74.7937	83.2901	74.3882	69.6996	75.1441

Table 8 The recall (%) comparison of knn-wdm method and the related classifiers fOR madelon data set. the recall measured fOR each category and only cv experiments were considered

Class	SVM	LMNN	KNN	Weighted KNN	Bipart	KNN-WDM
1	39.8667	75.4	89.2	76.9667	70.6	75.4333
2	60.3	71.3	50.9667	73.2	66.4333	74.0333

Table 9 The multiple classification accuracy (%) comparison of knn-wdm method and the related classifiers fOR heart disease data set and hungary database

	SVM	LMNN	KNN	Weighted KNN	Bipart	KNN-WDM
CV	57.3036	52.088	49.7593	49.9038	56.6762	59.6848
*RS*_1	54.0377	46.7427	47.8491	48.1132	53.3585	53.5094
*RS*_2	52.3404	46.4219	49.0213	47.9574	52.2979	53.5745
*RS*_3	53.3398	47.6484	50.4369	48.2039	54.8544	56.4078
*RS*_4	54.375	45.9045	49.4886	49.0909	55.2273	56.5909
*RS*_5	55.932	47.8592	50.8163	49.0476	53.9456	57.551
*RS*_6	54.2373	49.5915	48.7288	50.9322	54.5763	54.8305
*RS*_7	55.5455	51.8772	49.4318	52.9545	56.25	56.4773
*RS*_8	54.8305	52.193	48.8136	51.8644	56.7797	53.3898
*RS*_9	55.5517	53.4386	50	52.069	53.7931	56.2069

Table 10 The precision (%) comparison of the knn-wdm method and the related classifiers fOR the heart disease data set and hungary database. the precision measured fOR each category and only cv experiments were considered

Class	SVM	LMNN	KNN	Weighted KNN	Bipart	KNN-WDM
0	50	60.1176	58.6408	58.78	63.0994	64.4037
1	7.5112	5.9828	4.9819	4.322	6.0462	7.1691
2	0.4811	2.0607	1.8091	1.6061	4.4405	4.1948
3	2.6841	3.208	1.6962	1.8057	4.8956	6.263
4	0.2245	0.0714	0.9064	0.6436	2.8234	1.5305

Table 11 Comparison the recall (%) of knn-wdm method fOR each category with the related classifiers fOR heart disease data set and hungary database (only cv experiments were considered)

Class	SVM	LMNN	KNN	Weighted KNN	Bipart	KNN-WDM
0	70.3036	73.9211	71.538	72.0848	77.8918	82.5643
1	56.3333	19.6667	16.0833	14.8333	18.1667	19.0833
2	14.3333	9.6667	8.8333	8	17.6667	17
3	16.1667	13.8333	7.8333	8.6667	19.1667	24.5
4	3.5	0.5	7.5	6	23	13

Table 12 The multiple classification accuracy (%) comparison of knn-wdm method and the related classifiers fOR mice protein expression data set

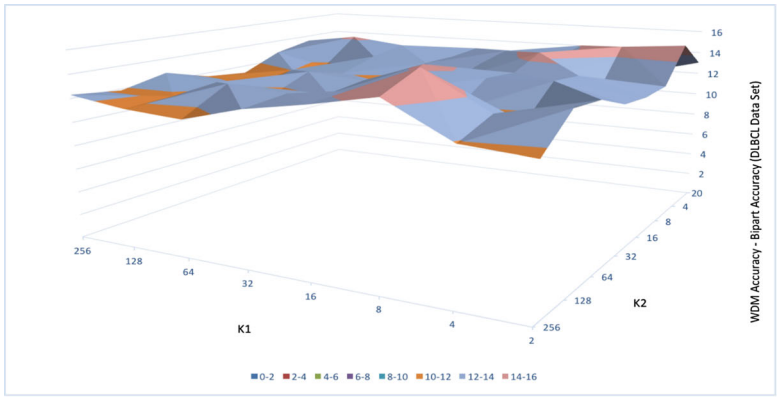
	SVM	LMNN	KNN	Weighted KNN	Bipart	KNN-WDM
CV	96.2368	98.5367	96.214	98.4058	98.0846	99.7309
RS_1	82.5412	80.1325	56.5844	69.177	81.7181	82.6132
RS_2	82.9282	85.3604	69.3519	80.9375	93.75	94.838
RS_3	93.0159	95.3227	76.3889	87.381	97.4339	97.8704
RS_4	93.4259	96.2891	82.7932	91.3272	98.5957	98.5031
RS_5	93.463	98.9795	86.7778	94.1481	99.3333	99.2963
RS_6	95.162	99.48	89.3056	96.8519	99.4213	99.3519
RS_7	95.6173	98.4215	92.4383	97.3148	99.5531	99.6605
RS_8	96.8981	99.6324	93.7963	98.1944	99.7685	99.7685
RS_9	97.8704	98.8681	95.8333	98.9815	99.6296	99.7222

Table 13 The precision (%) comparison of knn-wdm method and the related classifiers fOR mice protein expression data set. the precision measured fOR each category and only cv experiments were considered

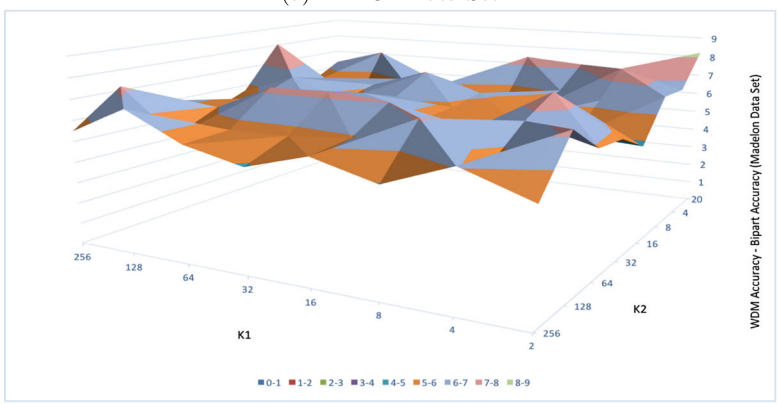
Class	SVM	LMNN	KNN	Weighted KNN	Bipart	KNN-WDM
$c-CS-s$	90.5916	91.1325	79.7327	95.8835	98.1088	99.1604
$c-CS-m$	92.4815	91.3601	81.9505	90.5047	97.9353	98.2647
$c-SC-s$	85.897	91.3227	77.8197	89.4772	99.2945	98.0734
$c-SC-m$	91.3156	92.2891	79.8715	90.6943	98.9615	98.8782
$t-CS-s$	90.2368	88.9795	74.7095	87.1282	96.0432	97.5682
$t-CS-m$	90.6577	90.48	82.4742	90.2144	97.9292	98.1437
$t-SC-s$	90.7824	90.4215	78.2103	89.9182	98.6095	98.0976
$t-SC-m$	92.9892	91.6324	79.9151	90.3315	98.4171	98.0976

Table 14 The recall (%) comparison of knn-wdm method and the related classifiers fOR mice protein expression data set. the recall measured fOR each category and only cv experiments were considered

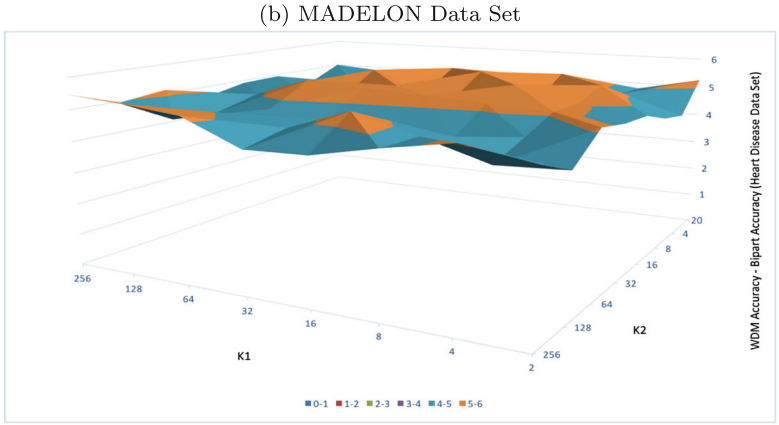
Class	SVM	LMNN	KNN	Weighted KNN	Bipart	KNN-WDM
$c-CS-s$	93.4505	98.456	95.2418	91.4231	88.3177	98.7418
$c-CS-m$	94.8	99.3333	94.8	99.7333	89.9095	100
$c-SC-s$	95.044	98.0824	100	100	87.5029	100
$c-SC-m$	96.8667	98.0667	99.2	99.5333	88.9932	99.2667
$t-CS-s$	90	97.9091	95.6364	99.9091	87.0589	100
$t-CS-m$	93.8846	99.1758	89.8297	98.7418	88.8538	99.9286
$t-SC-s$	94.6099	99.2527	99.1099	99.2473	88.2942	100
$t-SC-m$	100	97.8681	95.5549	98.7308	88.2898	100



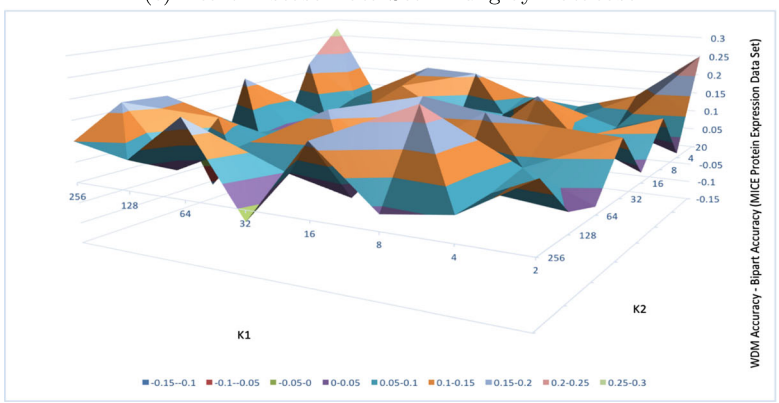
(a) DLBCL Data Set



(b) MADELON Data Set



(c) Heart Disease Data Set - Hungray Database



(d) Mice Protein Expression Data Set

Fig. 3 Difference between the accuracies of KNN-WDM and Bipart ($\text{KNN-WDM}_{\text{Accuracy}} - \text{Bipart}_{\text{Accuracy}}$) with changing the value of k_1 and k_2 for DLBCL (**a**), MADELON (**b**), Heart Disease (**c**) and Mice Protein Expression (**d**) data sets. This figure shows that KNN-WDM is more accurate than Bipart when the k_1 and k_2 have high values. However when the value of k_1 and k_2 reduces, the value of $|\text{KNN-WDM}_{\text{Accuracy}} - \text{Bipart}_{\text{Accuracy}}|$ decrease too in considered data sets

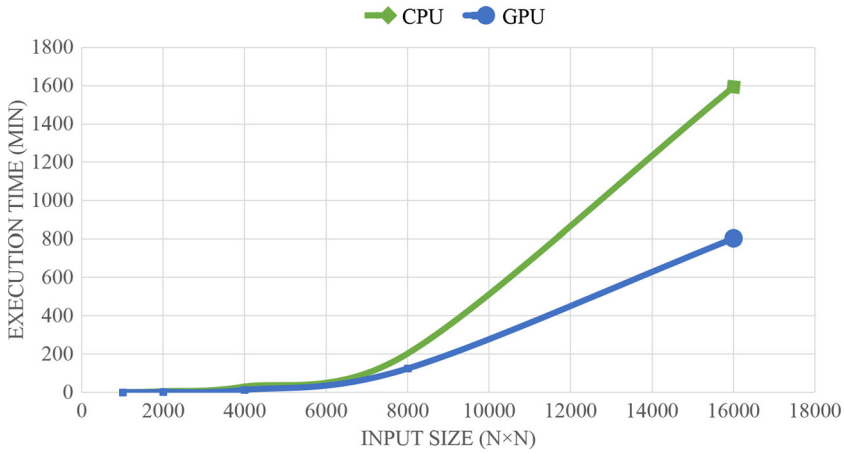


Fig. 4 The comparison of execution time of CPU and Matlab implementation on GPU for computing the Euclidean distance. The execution time includes the computation time plus the transfer time of data between the host and GPU device. This experiment showed that GPU is faster than CPU for larger input size

the labeled training set (k_1) and another for the test set (k_2). Fig. 3 shows the difference between KNN-WDM and Bipart accuracies with changing values of k_1 and k_2 in the range from 2 to 256.

Figure 3 reveals that KNN-WDM has higher accuracy than Bipart for all of the data sets. The performance of KNN and Bipart depend on good choices of the numbers of neighbors. In contrast, KNN-WDM is less sensitive to the number of neighbors than Bipart, and thus it produces good results even when the number of neighbors is not at the optimal setting. It is observed in results that when the values of k_1 and k_2 are both small, the value of $|\text{KNN-WDM}_{\text{Accuracy}} - \text{Bipart}_{\text{Accuracy}}|$ is higher than the situation when k_1 and k_2 are close to 256. This observed for DLBCL, Mdelon and Heart disease data sets clearly. The accuracy of KNN-WDM is close to Bipart for Mice Expression data set and this phenomenon is not clear for this data set. WDM assigns higher weights to the similar samples and lower weights to the dissimilar ones. WDM distinguishes more relevant samples, with more neighbors than Bipart. Therefore, the classifier performs better with increasing numbers of neighbors and this is the reason behind these results.

5 Speedup evaluation

According to runtime profiling, distance calculation in (3) is chosen for parallelization by the GPU with Matlab. Fig. 4 shows the execution time of distance calculation between two random square matrices. The input data are random numbers between zero and one. The execution time includes the time of doing the calculation 1,000 times, and for the data transfer time between the host and the GPU. This experiment is designed to evaluate WDM distance calculation time for large input sizes. Fig. 4 shows that GPU with Matlab is faster than CPU, and the advantage of GPU becomes higher for input with larger sizes.

Figure 5 compares different implementations of WDM on the GPU device. The comparion is based on the execution time (log of minutes) of the original code, the optimized version for CPU, GPU implementation with Matlab, and CUDA. The GPU methods use both optimized CPU code and GPU. The GPU-MATLAB approach calculates Euclidean distance on GPU. The GPU-CUDA implements weighted-distance metric with CUDA. This experiment measures the effect on the execution time by increasing the

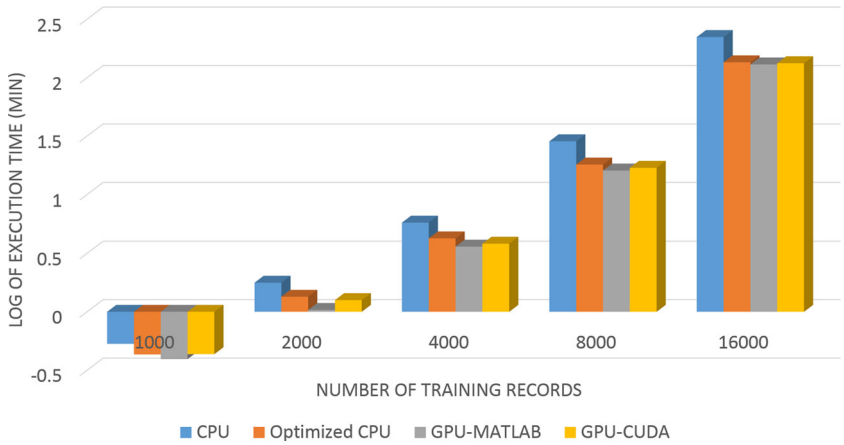


Fig. 5 Comparison of *log* execution time (minute) of CPU, Optimized CPU, and two hybrid implementations of WDM. The first hybrid approach uses Matlab Parallel Computing toolbox (GPU-MATLAB) and the second calls a CUDA kernel for parallel implementation (GPU-CUDA). This figure shows that GPU implementation with Matlab is the fastest approach in this experiment

number of records in the training data. Other factors are kept constant during the experiment. The number of features are 400, the test set contains 1000 records, the number of classes is 4, and all the input are random numbers.

The results in the Fig. 5 show the GPU-MATLAB approach is the fastest, and GPU-CUDA is the second fastest method. However, the execution time of optimized CPU version and GPU implementations are close together and it is against the initial expectation. The CUDA implementation does not reach its potential, because of irregular memory access patterns by each thread. Many threads access to different parts of the memory at the same time. This leads to serialization of memory access. The small size of intermediate variables that sent to the GPU explains the results of GPU-MATLAB approach. In addition, the transfer time of data between the host and the GPU increases with the numbers of records. These results show that communication between the host and the GPU is the bottleneck.

6 Future directions and conclusion

Possible future directions include improving the classification accuracy and parallel implementation. In this study, WDM is used in conjunction with KNN classification, and this may limit the accuracy of the WDM method. One way to improve the results is to use more sophisticated classifiers such as SVMs or neural networks, as WDM allows for any classification method to be used. The Matlab Parallel Computing Toolbox has some limitation on the used GPU devices. The performance of the GPU implementation of the WDM method may improve with SIMD instructions to provide data-level parallelism. Many of the latest integrated core architecture such as Intel Xeon Phi processors provide data-level parallelism. Future work may use other software and hardware parallel mechanisms to achieve high performance.

The proposed WDM method assigns weight to each sample based on similarity and dissimilarity to query sample using labeled training data and unlabeled test data. WDM is a new semi-supervised method that learns a weighted-distance metric in order to takes advantage of potential group structure in data. The main goals of using the weights are to improve the accuracy and to reduce the sensibility of the classifier to the number of neighbors. Experiments show that WDM performs favorably compared to other classifiers and distance metrics. We provided in-depth analysis of the parallel GPU implementations of the WDM method. Our empirical studies showed significant speedup when we compared the parallel GPU implementation to the CPU code. The Matlab GPU implementation is faster than CPU; however, its execution time is close to the optimized CPU version. The small size of intermediate variables and the time to transfer data between the host and the GPU are main reasons behind this observation. Also, the execution time of CUDA implementation is slower than expected due to irregular memory access. In order to maximize performance, WDM is implemented by a hybrid approach that use both CPU and GPU. The parallel implementation of the proposed method is fast for large input, and it can predict the pattern accurately when the number of classes is large.

Acknowledgments This study could not completed with the effort and co-operation of Professor Ming Ouyang from Computer Science Department of the University of Massachusetts Boston. His comments greatly improved the manuscript.

References

1. Blake C, Merz CJ (1998) {UCI} repository of machine learning databases
2. Cai D, He X, Han J (2007) Semi-supervised discriminant analysis. In: Computer vision, 2007. ICCV 2007. IEEE 11th international conference on, pp. 1–7. IEEE
3. Chapelle O, Schölkopf B, Zien A et al. (2006) Semi-supervised learning
4. Coates A, Huval B, Wang T, Wu D, Catanzaro B, Andrew N (2013) Deep learning with cots hpc systems. In: Proceedings of the 30th international conference on machine learning, pp 1337–1345
5. cuBLAS (2015) cuBLAS the nvidia cuda basic linear algebra subroutines (cublas) library @ONLINE. https://developer.nvidia.com/cuBLAS
6. Gou J, Du L, Zhang Y, Xiong T (2012) A new distance-weighted k-nearest neighbor classifier. J Inf Comput Sci 9:1429–1436
7. Guyon I, Gunn S, Ben-Hur A, Dror G (2004) Result analysis of the nips 2003 feature selection challenge. In: Advances in neural information processing systems, pp 545–552
8. Higuera C, Gardiner KJ, Cios KJ (2015) Self-organizing feature maps identify proteins critical to learning in a mouse model of down syndrome. PloS one 10(6):e0129,126
9. Hoshida Y, Brunet JP, Tamayo P, Golub TR, Mesirov JP (2007) Subclass mapping: identifying common subtypes in independent disease data sets
10. Mathworks (2015) Matlab @ONLINE. http://www.mathworks.com/products/matlab/
11. Mathworks (2015) Matlab parallel toolbox @ONLINE. https://www.mathworks.com/parallel-computing
12. Mu Y, Ding W, Tao D (2013) Local discriminative distance metrics ensemble learning. Pattern Recogn 46(8):2337–2349
13. Mu Y, Lo H, Ding W, Tao D (2014) Face recognition from multiple images per subject. In: Proceedings of the ACM international conference on multimedia, pp. 889–892. ACM
14. Mu Y, Lo HZ, Ding W, Amaral K, Crouter SE (2014) Bipart: Learning block structure for activity detection. IEEE Trans Knowl Data Eng 26(10):2397–2409
15. NVIDIA (2015) CUDA cuda instructions @ONLINE. https://developer.nvidia.com/cuda-zone
16. NVIDIA (2015) cuDNN nvidia cudnn - gpu accelerated deep learning @ONLINE. https://developer.nvidia.com/cuDNN
17. Reese J, Zaranek S (2012) Gpu programming in matlab. Mathworks News&Notes Natick, MA: The MathWorks Inc pp. 22–5

18. Schölkopf B, Smola AJ (2002) Learning with kernels: Support vector machines, regularization, optimization, and beyond MIT press
19. Sugiyama M (2007) Dimensionality reduction of multimodal labeled data by local fisher discriminant analysis. J Mach Learn Res 8:1027–1061
20. Tishby N, Pereira FC, Bialek W (2000) The information bottleneck method arXiv preprint physics/0004057
21. Weinberger KQ, Saul LK (2009) Distance metric learning for large margin nearest neighbor classification. J Mach Learn Res 10:207–244
22. Whitehead N, Fit-Florea A (2011) Precision & performance: Floating point and ieee 754 compliance for nvidia gpus. rn (A+ B) 21:1–1874919,424
23. Wilkinson JH, Wilkinson JH, Wilkinson JH (1965) The algebraic eigenvalue problem, vol 87. Clarendon Press, Oxford
24. Xing EP, Jordan MI, Russell S, Ng AY (2002) Distance metric learning with application to clustering with side-information. In: Advances in neural information processing systems, pp 505–512
25. Zavrel J (1997) An empirical re-examination of weighted voting for k-nn. In: Proceedings of the 7th belgian-dutch conference on machine learning, pp. 139–148. Citeseer

Hamidreza Mohebbi received his M.S degree from Iran University of Science and Technology in 2011. He is currently pursuing his Ph.D. degree of Computer Science at the University of Massachusetts Boston working on developing parallel algorithms for fields like Machine Learning, Markov Chains and etc. Hamidreza has worked as a lecturer in the Payamnour University of Iran and the University of Massachusetts Boston. His research interests include parallel programming, machine learning, artificial intelligence and markov chains.

Yang Mu received his B.S. and Ph.D degree from Jilin University and University of Massachusetts Boston in 2008 and 2015 respectively. Prior to his PhD study, he worked at Nanyang Technological University as a research assistant. His research interests include machine learning and data mining. His papers have been published on many top venues such as Pattern Recognition, IEEE TKDE, IEEE T-SMC part B, ACM SIGKDD, ACM MM and IEEE ICDM.

Wei Ding received her Ph.D. degree in Computer Science from the University of Houston in 2008. She is an Associate Professor of Computer Science in the University of Massachusetts Boston. Her research interests include data mining, machine learning, artificial intelligence, computational semantics, and with applications to astronomy, geosciences, and environmental sciences. She has published more than 105 referred research papers, 1 book, and has 2 patents. She is an Associate Editor of Knowledge and Information Systems (KAIS) and an editorial board member of the Journal of Information System Education (JISE), the Journal of Big Data, and the Social Network Analysis and Mining Journal. She is the recipient of a Best Paper Award at the 2011 IEEE International Conference on Tools with Artificial Intelligence (ICTAI), a Best Paper Award at the 2010 IEEE International Conference on Cognitive Informatics (ICCI), a Best Poster Presentation award at the 2008 ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPAITAL GIS), and a Best PhD Work Award between 2007 and 2010 from the University of Houston. Her research projects are sponsored by NIH, NASA, and DOE. She is an IEEE senior member and an ACM senior member.