The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Doubly Approximate Nearest Neighbor Classification

Weiwei Liu,[†] Zhuanghua Liu,[‡] Ivor W.Tsang,[‡] Wenjie Zhang,[†] Xuemin Lin[†]
[†]School of Computer Science and Engineering, The University of New South Wales
[‡]Center for Artificial Intelligence, University of Technology Sydney
{liuweiwei863, liuzhuanghua1991}@gmail.com, ivor.tsang@uts.edu.au,
wenjie.zhang@unsw.edu.au, lxue@cse.unsw.edu.au

Abstract

Nonparametric classification models, such as K-Nearest Neighbor (KNN), have become particularly powerful tools in machine learning and data mining, due to their simplicity and flexibility. However, the testing time of the KNN classifier becomes unacceptable and the KNN's performance deteriorates significantly when applied to data sets with millions of dimensions. We observe that state-of-the-art approximate nearest neighbor (ANN) methods aim to either reduce the number of distance comparisons based on tree structure or decrease the cost of distance computation by dimension reduction methods. In this paper, we propose a doubly approximate nearest neighbor classification strategy, which marries the two branches which compress the dimensions for decreasing distance computation cost as well as reduce the number of distance comparison instead of full scan. Under this strategy, we build a compressed dimensional tree (CD-Tree) to avoid unnecessary distance calculations. In each decision node, we propose a novel feature selection paradigm by optimizing the feature selection vector as well as the separator (indicator variables for splitting instances) with the maximum margin. An efficient algorithm is then developed to find the globally optimal solution with convergence guarantee. Furthermore, we also provide a data-dependent generalization error bound for our model, which reveals a new insight for the design of ANN classification algorithms. Our empirical studies show that our algorithm consistently obtains competitive or better classification results on all data sets, yet we can also achieve three orders of magnitude faster than state-of-the-art libraries on very high dimensions.

Introduction

Nonparametric models have played a vital role in machine learning and data mining as a result of their simplicity and flexibility. One of the most popular nonparametric classification methods is called K-Nearest Neighbor (KNN), and is identified as one of the top 10 data mining algorithms. KNN classification has shown promising results to various real world applications, such as face recognition, document annotation and image processing. However, its linear dependence on the number of instances and features hinders its use as a practical algorithm for large scale high dimensional settings.

Recently, the emerging trends of ultrahigh dimensionality have been analyzed in (Tan, Wang, and Tsang 2010; Zhai, Ong, and Tsang 2014; Tan, Tsang, and Wang 2014; Liu and Tsang 2016; 2017; Liu, Shen, and Tsang 2017). For example, the Web continues to generate quintillion bytes of data daily, leading to a great challenge for KNN classification on the WEBSPAM data set with 16,609,143 features, collected from (Wang, Irani, and Pu 2012). The testing time of KNN on this data set is unacceptable. Furthermore, KNN achieves degenerated performance on the WEBSPAM data set, because all vectors are almost equidistant to the testing instance in 10 millions of dimensions. Therefore, KNN suffers from the curse of dimensionality (Beyer et al. 1999), and how to precisely and efficiently scale the KNN classifier to very high dimensional settings poses a serious challenge.

It is desirable that an approximate nearest neighbor (ANN) classification algorithm, designed for very high dimensional settings, should meet two principles: 1) The number of distance computations should be reduced. 2) The cost of each distance computation should be decreased. Unfortunately, the state-of-the-art ANN techniques fall short in either of these two principles. Many tree-based algorithms, such as KD-Tree and K-Means Tree (Muja and Lowe 2014), have been developed to reduce the number of distance computations. However, such approaches still depend on the original feature dimensions, which lead to high distance computational cost and degenerated performance for very high dimensions; the authors in (Weber, Schek, and Blott 1998) have already demonstrated the underperformance of tree-based methods on high dimensions. In addition, hashing (Gionis, Indyk, and Motwani 1999; Charikar 2002; Andoni et al. 2015; Shen et al. 2017) is one of the widely-studied dimension reduction methods for decreasing the cost of distance computation by mapping high-dimensional data into a short binary code and using Hamming distance as the distance metric. Product quantization (PQ) (Jégou, Douze, and Schmid 2011) is then proposed to obtain more precise distance than hashing and reduce the quantization error. However, due to the non-convexity of binary embedding and orthogonal constraints, the main drawback of most learning-based hashing and PQ is that they usually get stuck in local minimum for their optimization problem.

This paper proposes a doubly approximation strategy for nearest neighbor classification, which marries the ad-

Table 1: Comparison between the existing methods and ours. (r: maximum number of points to examine for K-Means Tree and KD-Tree. b: # bits used for KBE. ϑ: # dictionaries used for KPQ. ς: the length of each dictionary used for KPQ. $\mathcal{T}$: the total selected feature set for CD-Tree. The distance computation is abbreviated to DC.)

Method	K-Means Tree (Muja and Lowe)	KD-Tree (Muja and Lowe)	KBE (Zhang et al.)	KPQ (Zhang et al.)	CD-Tree (ours)
Reduce # of DC	✓	✓	×	✓	✓
Reduce DC cost	×	×	✓	×	✓
Space cost	$\mathcal{O}(nm)$	$\mathcal{O}(nm)$	$\mathcal{O}(nb)$	$\mathcal{O}(n\vartheta log(\varsigma) + m\varsigma)$	$\mathcal{O}(n\|\mathcal{T}\|)$
Training cost	$\mathcal{O}(nmlogn)$	$\mathcal{O}(nlogn)$	$\mathcal{O}(nmlog^2m)$	$\mathcal{O}(nmlog^2m)$	$\mathcal{O}(mlogB + nlogn + t\varpi B)$
Testing cost	$\mathcal{O}(rmlogn)$	$\mathcal{O}((logn + r)m)$	$\mathcal{O}(nb)$	$\mathcal{O}(n\vartheta + m\varsigma)$	$\mathcal{O}(\log(\frac{n}{N_{min}})\varpi B + N_{min}\varpi B)$

vantages of the dimension reduction and tree-based methods, and simultaneously remedies the defect of these approaches. Table 1 demonstrates the comparison between the existing methods and ours. Specifically, KD-Tree has been an efficient space-partitioning data structure for organizing data in low dimensions. However, KD-Tree only chooses a single feature with the maximum variance in each node as the hyperplane to split the data. On the other hand, Ram *et al.* (Ram, Lee, and Gray 2012) have proposed the max-margin tree (MMT) to learn a better hyperplane, and demonstrated that large margin partitions can improve tree search performance. Unfortunately, the distance computational cost of these methods on very high dimensions is prohibitive. To break the bottleneck of tree-based methods, motivated by hashing and PQ, we build a compressed dimensional tree (CD-Tree), which is inspired by the name of KD-Tree, to learn an informative and compact feature subset as well as the sparse hyperplane for splitting the data with the maximum margin (Yang et al. 2017; Liu, Tsang, and Müller 2017). To avoid being stuck in local minimum, we employ the tight convex relaxation technique (Li et al. 2013) and develop an efficient algorithm to find a globally optimal solution with convergence guarantee. Lastly, we analyze generalization error bound for the proposed doubly approximate nearest neighbor classification. Experiments on a wide spectrum of data sets show that our proposed method excels in all data sets, yet we obtain three orders of magnitude faster than state-of-the-art libraries on very high dimensions.

Compressed Dimensional Tree

This section first introduces the notion of the compressed dimensional node and then presents the CD-Tree.

We denote the transpose of the vector/matrix by the superscript $'$ and the logarithms to base 2 by log. Let $\odot$ represent the elementwise product sign and N_{min} be the constant. If $\mathcal{Q}$ is a set, $|\mathcal{Q}|$ denotes its cardinality. Let $\mathbb{R}$ represent real number and $||\cdot||_2$ denote the l_2-norm. Given a set of instances $\{x_i\}_{i=1}^n$, where $x_i \in \mathbb{R}^{m\times 1}$. $\mathbf{w} \in \mathbb{R}^{m\times 1}$ denotes the weight vector. A vector with all entries equal to one is represented as $\mathbf{1} \in \mathbb{R}^{n\times 1}$. We define $\{s_i\}_{i=1}^n, s_i \in \{\pm 1\}$, as the separator variables, which are used as the indicators for splitting instances. Let $\mathcal{S} = \{\mathbf{s} = [s_1, \cdots, s_n]' \in \{\pm 1\}^n | -\beta \leq \sum_{i=1}^n s_i \leq \beta\}$ be the domain of $\mathbf{s}$, where β is a small constant controlling the imbalance of partitions.

Algorithm 1 CD-Tree

Input: A set of instances $\{x_i\}_{i=1}^n$, where $x_i \in \mathbb{R}^{m\times 1}$. **Output:** CD-Tree.

```
1: while CDN contains more than $N_{min}$ instances do
2:     Learn the classifier for this CDN.
3:     Split the instances of this node into left child if these
       instances are learned to be positive and call Algorithm 1 with split data the input.
4:     Split the remaining instances of this node into right
       child and call Algorithm 1 with the remaining instances the input.
5: end while
6: Return the complete CD-Tree.
```

Before deriving the CD-Tree, we first introduce the notion of the compressed dimensional node (CDN):

Definition 1 (Compressed Dimensional Node (CDN)). *In order to learn a sparse hyperplane, we introduce a feature selection vector* $\mathbf{d} = [d_1, \cdots, d_m]' \in \mathcal{D}$*, where* $\mathcal{D} = \{\mathbf{d} = [d_1, \cdots, d_m]' \in [0,1]^m | \sum_{j=1}^m d_j \leq B\}$ *is the domain of* $\mathbf{d}$ *and* B *controls the sparsity of* $\mathbf{d}$*. A compressed dimensional node contains a linear decision function* $\mathfrak{F}(x) = \mathbf{w}'(x_i \odot \sqrt{\mathbf{d}} + b)$*, where* $\sqrt{\mathbf{d}} = [\sqrt{d_1}, \cdots, \sqrt{d_m}]'$ *and the parameters are obtained by solving the following problem:*

$$\begin{aligned} &\min_{\mathbf{d}\in\mathcal{D}} \min_{\mathbf{s}\in\mathcal{S}} \min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||_2^2 + \frac{C}{2}\sum_{i=1}^n \xi_i^2 \\ &s.t. \quad s_i\mathbf{w}'(x_i \odot \sqrt{\mathbf{d}} + b) \geq 1 - \xi_i, i = 1, \cdots, n \end{aligned} \tag{1}$$

Remark. The task of Problem (1) is to find the optimal sparse hyperplane as well as the optimal separator with the maximum margin.

We build a compressed dimensional tree in a top-down approach. Starting from the root, we solve Problem (1) on each CDN and then use the learned separator to split the instances into two child nodes. This process continues until the remaining data at the node cannot be further split by the induced classifier. The details are stated in Algorithm 1. We follow (Ram, Lee, and Gray 2012) to do prediction.

Learning the CDN

We develop the technique for learning the classifier at CDN in this section. For the sake of clarity, we consider the function without an offset, although the algorithm can be extended to the function with offset. Given $\mathbf{d}$ and $\mathbf{s}$, the inner minimization problem w.r.t. $\mathbf{w}$ and ξ is a standard SVM problem: $\min_{\mathbf{w},\xi} \frac{1}{2}||\mathbf{w}||_2^2 + \frac{C}{2}\sum_{i=1}^n \xi_i^2 \quad : \quad s.t. \quad s_i\mathbf{w}'(x_i \odot \sqrt{\mathbf{d}}) \geq 1 - \xi_i, i = 1, \cdots, n$. This problem can be solved in its dual form: $\max_{\alpha\in\mathcal{A}} -\frac{1}{2}||\sum_{i=1}^n \alpha_i s_i(x_i \odot \sqrt{\mathbf{d}})||_2^2 - \frac{1}{2C}\alpha'\alpha + \alpha'\mathbf{1}$, where $\mathcal{A} = \{\alpha | 0 \leq \alpha_i, i = 1, \cdots, n\}$.

The separator variables and feature selection variables in Problem (1) are optimized by our proposed separator generation and feature generation strategies in the following subsections. Moreover, we provide the convergence analysis for proposed algorithms.

Separator Generation

Given $\mathbf{d}$, we define $f(\alpha, \mathbf{s}) = -\frac{1}{2}||\sum_{i=1}^n \alpha_i s_i(x_i \odot \sqrt{\mathbf{d}})||_2^2 - \frac{1}{2C}\alpha'\alpha + \alpha'\mathbf{1}$. Following the minimax inequality theorem (Kim and Boyd 2008), the inner minimization problem w.r.t. $\mathbf{s}$ and α of Problem (1) can be lower-bounded by

$$\max_{\alpha\in\mathcal{A}} \min_{\mathbf{s}\in\mathcal{S}} f(\alpha, \mathbf{s}) \tag{2}$$

By introducing variable $\theta \in \mathbb{R}$, Problem (2) becomes

$$\max_{\alpha\in\mathcal{A},\theta\in\mathbb{R}} -\theta \quad : \quad s.t. \quad -f(\alpha, \mathbf{s}) \leq \theta, \forall \mathbf{s} \in \mathcal{S} \tag{3}$$

(Li et al. 2013) have already shown that problem (3) is the tight convex relaxation of the inner minimization problem w.r.t. $\mathbf{s}$, $\mathbf{w}$ and ξ of Problem (1). Since problem (3) involves an exponential number of constraints, the cutting plane algorithm can be used here to solve the above problem. Specifically, given α_{t-1} and $\mathbf{d}$, the worst-case analysis is to infer the most-violated $\mathbf{s}_t$, and add it into the active constraint set $\mathcal{C}_t$. Then, we update α_t by solving the following problem:

$$\max_{\alpha\in\mathcal{A},\theta\in\mathbb{R}} -\theta \quad : \quad s.t. \quad -f(\alpha, \mathbf{s}) \leq \theta, \forall \mathbf{s} \in \mathcal{C}_t \tag{4}$$

By introducing dual variable μ_h for each constraint defined by $\mathbf{s}_h$, we can transform problem (4) to an multiple kernel learning (MKL) problem (Liu et al. 2015; Liu and Tsang 2017; Gong et al. 2017) as follows:

$$\max_{\alpha\in\mathcal{A}} \min_{\mu\in\mathcal{M}} \sum_{\mathbf{s}_h\in\mathcal{C}_t} \mu_h f(\alpha, \mathbf{s}_h) = \min_{\mu\in\mathcal{M}} \max_{\alpha\in\mathcal{A}} \sum_{\mathbf{s}_h\in\mathcal{C}_t} \mu_h f(\alpha, \mathbf{s}_h) \tag{5}$$

where $\mathcal{M} = \{\mu | \sum \mu_h = 1, \mu_h \geq 0\}$. The details of the separator generation are shown in Algorithm 2.

Algorithm 2 Separator Generation

```
Input: A set of instances $\{x_i\}_{i=1}^n$, where $x_i \in \mathbb{R}^{m\times 1}$. Initialize $\alpha$ and $\mathbf{d}$.
1: $t = 0, \mathcal{C}_t = \emptyset$.
2: repeat
3:     $t = t + 1$.
4:     Separator inference: Finding the most violated $\mathbf{s}_t$ and set $\mathcal{C}_t = \mathcal{C}_{t-1} \cup \{\mathbf{s}_t\}$ .
5:     Master problem optimization: Solving problem (5) over $\mathcal{C}_t$.
6: until $\epsilon$-optimal.
```

Algorithm 3 Feature Generation

```
Input: A set of instances $\{x_i\}_{i=1}^n$, where $x_i \in \mathbb{R}^{m\times 1}$. Initialize $\alpha_0$ and $\mathbf{s}_0$.
1: $z = 0, \mathcal{U}_z = \emptyset$.
2: repeat
3:     $z = z + 1$.
4:     Feature inference: Generating a feature selection vector $\mathbf{d}_z$ by solving problem (11) based on $\alpha_{z-1}$ and $\mathbf{s}_{z-1}$and set $\mathcal{U}_z = \mathcal{U}_{z-1} \cup \{\mathbf{d}_z\}$ .
5:     Call Algorithm 2 with $(\{x_i\}_{i=1}^n, \mathcal{U}_z)$ the input and obtain the output: $(\alpha_z, \mathbf{s}_z)$.
6: until $\epsilon$-optimal.
```

We have the following convergence rate theorem:

Theorem 1. *Let O^t be the objective value of Eq.(5) at the t-th iteration and O^* be the optimal objective value. Given $-f(\alpha, \mathbf{s})$ is λ-strongly convex function and L is the Lipschitz constant of $-f(\alpha, \mathbf{s})$, algorithm 2 converges in no more than $\frac{O^1 - O^*}{\varphi}$ iterations, where $\varphi = \left(\frac{-c+\sqrt{c^2+4\epsilon}}{2}\right)^2$, and $c = L\sqrt{\frac{2}{\lambda}}$.*

The proof can be adapted from (Li et al. 2013). We use the method in (Li et al. 2013) to find the most violated $\mathbf{s}_t$, which takes $\mathcal{O}(n \log n)$ time complexity. To efficiently solve problem (5), we first transform problem (5) to the equivalent $l_{2,1}^2$-regularized problem and then adapt the fast accelerated proximal gradient (APG) method (Beck and Teboulle 2009; Toh and Yun 2009) to solve the reduced problem.

Let $\bar{x}_i = x_i \odot \sqrt{\mathbf{d}}, i = 1, \cdots, n$. Suppose that after t-th iterations, we generate the separator sequence: $\mathbf{s}_1, \cdots, \mathbf{s}_t$. Let $\mathbf{w}_t$ denote the weight vector w.r.t. $\bar{x}_i$ for each iteration and $\mathcal{W} = [\mathbf{w}_1, \cdots, \mathbf{w}_t]$. We define $P(\mathcal{W}) = \frac{C}{2}\sum_{i=1}^n \xi_i^2$, where $\xi_i = \max\{0, 1 - \sum_{h=1}^t s_{ih}\mathbf{w}_h'\bar{x}_i\}$. Then, we present the following theorem:

Theorem 2. *The MKL problem (5) can be equivalently transformed to the following primal problem:*

$$\min_{\mathcal{W}} \frac{1}{2}\Big(\sum_{h=1}^t ||\mathbf{w}_h||_2\Big)^2 + P(\mathcal{W}) \tag{6}$$

where dual optimal solution α^* can be recovered from the optimal solution ξ^*: $\alpha^* = C\xi^*$.

Proof. Let $q_h = ||\mathbf{w}_h||_2$ and $q = \sum_{h=1}^t q_h$, we have $\frac{1}{2}\Big(\sum_{h=1}^t ||\mathbf{w}_h||_2\Big)^2 = \frac{1}{2}q^2$. Apparently, we have $q_h \geq 0$ and $q \geq 0$. We define the cone $\mathscr{C} = \{(\mathbf{u}_h, v) \in \mathbb{R}^{m+1}, ||\mathbf{u}_h||_2 \leq v\}$. Then, Eq.(6) can be transformed to the following problem:

$$\min_{\mathcal{W}} \frac{1}{2}q^2 + P(\mathcal{W}) \quad : \quad s.t. \quad \sum_{h=1}^t q_h \leq q, (\mathbf{w}_h, q_h) \in \mathscr{C} \tag{7}$$

By introducing the positive Lagrangian dual variables $\alpha, \delta, \phi, \varrho$ to the corresponding constraints, we find the

Lagrangian function: $\mathcal{L}(q, \mathcal{W}, \xi; \alpha, \delta, \phi, \varrho) = \frac{1}{2}q^2 + \frac{C}{2}\sum_{i=1}^n \xi_i^2 - \sum_{i=1}^n \alpha_i(\sum_{h=1}^t s_{ih}\mathbf{w}_h'\bar{x}_i - 1 + \xi_i) + \delta(\sum_{h=1}^t q_h - q) - \sum_{h=1}^t(\phi_h'\mathbf{w}_h + \varrho_h q_h)$. The derivatives of the Lagrangian w.r.t. the primal variables have to vanish which leads to the following KKT conditions: $q = \delta, \varrho_h = \delta, \phi_h = -\sum_{i=1}^n \alpha_i s_{ih}\bar{x}_i, \xi_i = \frac{\alpha_i}{C}, ||\phi_h||_2 \leq \delta$. By substituting the above results into the Lagrangian function, we have

$$\mathcal{L}(q, \mathcal{W}, \xi; \alpha, \delta, \phi, \varrho) = -\frac{1}{2}\delta^2 - \frac{1}{2C}\alpha'\alpha + \mathbf{1}'\alpha$$

Therefore, Eq.(6) can be formulated as:

$$\begin{aligned} &\max_{\alpha,\delta} -\frac{1}{2}\delta^2 - \frac{1}{2C}\alpha'\alpha + \mathbf{1}'\alpha \\ &s.t. \quad ||\sum_{i=1}^n \alpha_i s_{ih}\bar{x}_i||_2 \leq \delta, h = 1, \cdots, t; \\ &\quad\quad \alpha_i \geq 0, i = 1, \cdots, n \end{aligned} \tag{8}$$

Let $\theta = \frac{1}{2}\delta^2 + \frac{1}{2C}\alpha'\alpha - \mathbf{1}'\alpha$ and $f(\alpha, \mathbf{s}_h) = -\frac{1}{2}||\sum_{i=1}^n \alpha_i s_{ih}\bar{x}_i||_2^2 - \frac{1}{2C}\alpha'\alpha + \mathbf{1}'\alpha$, we have

$$\begin{aligned} &\max_{\alpha,\theta} -\theta \\ &s.t. \quad -f(\alpha, \mathbf{s}_h) \leq \theta, h = 1, \cdots, t; \\ &\quad\quad \alpha_i \geq 0, i = 1, \cdots, n \end{aligned} \tag{9}$$

By setting $\mathcal{A} = \{\alpha | 0 \leq \alpha_i, i = 1, \cdots, n\}$, problem (9) is indeed in the form of problem (4). Lastly, we produce the connection between the primal formulation and dual formulation and complete the proof. □

Following Theorem 2, we address problem (6) rather than directly solving problem (5), which has great advantages for fast optimization. We modify the APG method for solving Problem (6). Lastly, we use $\mathfrak{F}(\bar{x}) = \sum_{h=1}^t \mathbf{w}_h'\bar{x}$ as the prediction function.

Feature Generation

Next, we focus on feature generation based on the solutions of Algorithm 2. After finding $\mathbf{s}$ and α, we solve the following problem to obtain the optimal $\mathbf{d}$.

$$\max_{\mathbf{d}\in\mathcal{D}} \frac{1}{2}||\sum_{i=1}^n \alpha_i s_i(x_i \odot \sqrt{\mathbf{d}})||_2^2 \tag{10}$$

Let $c_j = (\sum_{i=1}^n \alpha_i s_i x_{i,j})^2, j \in \{1, \cdots, m\}$, then problem (10) can be formulated as a linear programming problem:

$$\max_{\mathbf{d}\in\mathcal{D}} \frac{1}{2}\sum_{j=1}^m c_j d_j \tag{11}$$

This problem can be quickly solved by the sorting method, which takes only $\mathcal{O}(m \log B)$ time, making it computationally efficient. The feature generation algorithm is presented in Algorithm 3.

Convergence Analysis

Problem (1) involves separator variables, which are the binary discrete variables. Therefore, it is non-trivial to provide the globally convergence guarantee for Problem (1). This subsection aims to analyze the convergence of Algorithm 3 to the optimum of Problem (1). Let $\mathcal{F}_{\mathbf{s},\alpha}(\mathbf{d}) = \frac{1}{2}||\sum_{i=1}^n \alpha_i s_i(x_i \odot \sqrt{\mathbf{d}})||_2^2 + \frac{1}{2C}\alpha'\alpha - \alpha'\mathbf{1}$. Recall that both $\mathcal{D}$ and $\mathcal{A}$ are convex compact sets, according to the minimax saddle-point theorem (Sion 1958), we have $\min_{\mathbf{d}\in\mathcal{D}}\max_{\alpha\in\mathcal{A}} = \max_{\alpha\in\mathcal{A}}\min_{\mathbf{d}\in\mathcal{D}}$. Following the minimax inequality theorem, we have $\min_{\mathbf{s}\in\mathcal{S}}\max_{\alpha\in\mathcal{A}} \geq \max_{\alpha\in\mathcal{A}}\min_{\mathbf{s}\in\mathcal{S}}$. Thus, Problem (1) can be reformulated as:

$$\begin{aligned} \min_{\mathbf{d}\in\mathcal{D}}\min_{\mathbf{s}\in\mathcal{S}}\max_{\alpha\in\mathcal{A}} -\mathcal{F}_{\mathbf{s},\alpha}(\mathbf{d}) &= \min_{\mathbf{s}\in\mathcal{S}}\max_{\alpha\in\mathcal{A}}\min_{\mathbf{d}\in\mathcal{D}} -\mathcal{F}_{\mathbf{s},\alpha}(\mathbf{d}) \\ &\geq \max_{\alpha\in\mathcal{A}}\min_{\mathbf{s}\in\mathcal{S}}\min_{\mathbf{d}\in\mathcal{D}} -\mathcal{F}_{\mathbf{s},\alpha}(\mathbf{d}) \end{aligned} \tag{12}$$

Then, we focus on the following problem:

$$\min_{\alpha\in\mathcal{A}}\max_{\mathbf{s}\in\mathcal{S}}\max_{\mathbf{d}\in\mathcal{D}} \mathcal{F}_{\mathbf{s},\alpha}(\mathbf{d}) \tag{13}$$

Assume that Algorithm 3 generates a sequence of $\mathbf{d}$: $\mathbf{d}_1, \cdots, \mathbf{d}_z$, after z iterations. In the $z+1$-th iteration, $\mathbf{d}_{z+1}$ is found in terms of $\mathbf{s}_z$ and α_z. We define $\underline{\gamma}_z = \max_{1\leq i\leq z} \mathcal{F}_{\mathbf{s}_z,\alpha_z}(\mathbf{d}_i) = \min_{\alpha\in\mathcal{A}}\max_{\mathbf{s}\in\mathcal{S}}\max_{1\leq i\leq z} \mathcal{F}_{\mathbf{s},\alpha}(\mathbf{d}_i)$ and $\bar{\gamma}_z = \min_{1\leq j\leq z} \mathcal{F}_{\mathbf{s}_j,\alpha_j}(\mathbf{d}_{j+1}) = \min_{1\leq j\leq z}\max_{\mathbf{d}\in\mathcal{D}} \mathcal{F}_{\mathbf{s}_j,\alpha_j}(\mathbf{d})$. The following convergence theorem indicates that Algorithm 3 gradually approaches to the optimal solution:

Theorem 3. *Let $\gamma* = \min_{\alpha\in\mathcal{A}}\max_{\mathbf{s}\in\mathcal{S}}\max_{\mathbf{d}\in\mathcal{D}}\mathcal{F}_{\mathbf{s},\alpha}(\mathbf{d})$ be the optimal objective value of Problem (13), then sequences $\{\underline{\gamma}_z\}$ and $\{\bar{\gamma}_z\}$ have the following property:*

$$\underline{\gamma}_z \leq \gamma* \leq \bar{\gamma}_z \tag{14}$$

As the number of iterations z increases, $\{\underline{\gamma}_z\}$ is monotonically increasing and $\{\bar{\gamma}_z\}$ is monotonically decreasing.

We further derive the following lemma:

Lemma 1. *Algorithm 3 stops after a finite number of steps with the global solution of Problem (1).*

Generalization Error Bound

To the best of our knowledge, very few work study the ANN classification error bound by considering the margin, the capacity of the kernel matrix (Zhuang, Tsang, and Hoi 2011) and the training error loss simultaneously. This section aims to provide the generalization error bound for ANN classification using our learning model from aforementioned perspectives, and the theoretical justification for learning a sparse linear hyperplane in each decision node for ANN classification. Let $\mathbb{f}$ be a class of functions mapping from input space to $\{\pm 1\}$. We follow the definitions in (Bartlett and Mendelson 2002) and use $G_n(\mathbb{f})$ to denote the Gaussian complexity of $\mathbb{f}$. The generalization error bound for ANN classification using CD-Tree:

Theorem 4. *Let T be a CD-Tree, where the Euclidean norm of the linear functions in T is at most 1. Suppose that the* *depth of T is no more than ν, and every leaf node in T retains one instance. Let $(x_1, y_1), \cdots, (x_n, y_n)$ be the n training instances and $y_i \in \{\pm 1\}$, which are generated independently according to a probability distribution D. Then, we can bound the generalization error of ANN classification using T with probability greater than $1 - \delta$ to be less than*

$$\sum_l \sum_{i=1}^{\nu_l} \mathcal{E}_i^l + \frac{5c\zeta\nu G_n(T)}{2} + \big((3\nu+1)\sqrt{n}\zeta + 1\big)\sqrt{\frac{ln(6/\delta)}{2n}}$$

where c and ζ are the constants, $\nu_l(\nu_l \leq \nu)$ is the depth of leaf l, $\mathcal{E}_i^l = \frac{1}{n_l\gamma_i^l}\sum_{j=1}^{n_l}\xi_{i,j}^l + \frac{4}{n_l\gamma_i^l}\sqrt{tr(K_l)}$, n_l denotes the number of instances $(x_1^l, y_1^l), \cdots, (x_{n_l}^l, y_{n_l}^l)$ reaching leaf l, K_l is the kernel matrix for $\{x_1^l, \cdots, x_{n_l}^l\}$, $\xi_{i,j}^l = \max\big(0, \gamma_i^l - y_{i,j}^l g_i^l(x_j^l)\big)$, where $y_{i,j}^l$ represents the correct output for input x_j^l in decision node i and γ_i^l denotes the corresponding margin.

Remark. The data dependency of our bound comes through the training error loss, the margin and the capacity of the kernel matrix defined on the training data which can be transformed to a budget constraint on the weight coefficients. Specifically, our results indicate that decreasing training error loss and the capacity of the kernel matrix , and enlarging the margin can yield better generalization performance, which is equivalent to optimizing Problem (1). Thus, our results provide the theoretical justification to learn a CDN, and CD-Tree is able to tighten the generalization error bound. Furthermore, our analysis reveals a new insight for the design of ANN classification algorithms.

Computational and Practical Issues

Computational Cost

Let us equally split the instances of each node into left and right child in Algorithm 1. CD-Tree has a depth of $\left\lceil \log \frac{n}{N_{min}} \right\rceil$+1, where $\lceil \Upsilon \rceil$ is the smallest integer not less than Υ. Assume in each decision node, we select ϖB features. Our proposed CD-Tree takes $\mathcal{O}(\log(\frac{n}{N_{min}})\varpi B + N_{min}\varpi B)$ testing time, which is dominated by only a few selected features. Thus, our algorithm is expected to be much faster for testing. Following the computational cost analysis in (Muja and Lowe 2014) and (Zhang et al. 2015), we make the comparisons shown in Table 1. From this table, we can see that 1) The space cost of CD-Tree is lower than other methods and comparable with KBE; 2) The training cost of CD-Tree is lower than K-Means Tree, KBE and KPQ.

Practical Issues

Accessing features in sparse format is very expensive for high dimensional data sets. This paper proposes two efficient auxiliary data structures to reduce the computation cost of indexing and accessing features. We first propose a modified direct indexing method for small and medium-sized data sets. Let $\mathcal{N}_{|\mathcal{T}|}$ represent the natural number set which contains $|\mathcal{T}|$ numbers from 0 to $|\mathcal{T}| - 1$. Then, we build an injection table: $\mathcal{I} : \mathcal{T} \mapsto \mathcal{N}_{|\mathcal{T}|}$ to compress the whole selected feature set to a continuous index set. To support direct indexing for every instance, we maintain an array with size $|\mathcal{T}|$. For each feature that belongs to $\mathcal{T}$, we use $\mathcal{I}$ to find out the index number, and store the feature value at that index in the array. To handle very high dimensional data sets, in which the data is usually stored in sparse format, we maintain a hash table to store the feature ID as the key and the feature value for each instance. The method takes $\mathcal{O}(1)$ time to access the features.

Experiment

Data Sets and Baselines

This section evaluates the performance of various methods on a variety of real world data sets, which fall into two categories. The first category contains two small data sets and two medium-sized data sets. The second category contains three large-scale data sets with millions of dimensions. The URL data set is prepared by (Ma et al. 2009) and other data sets are collected from the LIBSVM website[1](Du et al. 2017; Wang et al. 2015). The training/testing partition is either predefined or the data is randomly split into 80% training and 20% testing (Wu et al. 2014). Table 3 shows the statistics of these data sets. We compare CD-Tree with several state-of-the-art methods: 1) RP+KD-Tree: Bingham *et al.* (Bingham and Mannila 2001) have already shown that projecting the data into a random lower-dimensional subspace yields results comparable to conventional dimensionality reduction methods such as principal component analysis (PCA), and using random projection (RP) is computationally significantly less expensive than using PCA. Thus, in this experiment, we first randomly project the data into a lower dimension, then build a KD-Tree on the embedding space. We run this method three times and take the average results. 2) FLANN (Muja and Lowe 2014): is a state-of-the-art library for performing fast KNN in high dimensional space. It contains a collection of algorithms, such as brute-force KNN (BF-KNN), KD-Tree and a modified K-Means Tree. 3) KBE and KPQ (Zhang et al. 2015): the Kronecker product is an up-to-date technique for hashing (KBE) and product quantization (KPQ) proposed in (Zhang et al. 2015). 4) MMT (Ram, Lee, and Gray 2012): an advanced tree-based algorithm. We use solvers provided by the respective authors.

For RP+KD-Tree, we project the data into $0.1m$ dimension for small and medium-sized data sets and 400 dimension for very high dimensional data sets. Following the parameter settings in (Li et al. 2013), β is set as $0.3n$. The parameter B is selected using 5-fold cross validation over the range $\{20, 50, 100, 200, 400\}$ for small and medium-sized data sets, and we set $B = 400$ for very high dimensional data sets. C is fixed to 5. We set r (the maximum number of points to examine for K-Means Tree and KD-Tree) as $\{5, 10, 30\}$ and we achieve similar prediction performance for different r. We therefore simply fix $r = 5$ for fast testing time. Following similar parameter settings to those in (Zhang et al. 2015) for KBE and KPQ, b is selected over the

[1]https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets

Table 2: Testing error rate (in %) of various methods on three of small and medium-sized data sets using 1NN and 5NN.

DATA SET	MMT	KD-TREE		K-MEANS TREE		KBE($b = 256$)		KPQ		CD-TREE	
		1NN	5NN	1NN	5NN	1NN	5NN	1NN	5NN	1NN	5NN
A7A	22.18	22.80	20.22	22.21	19.84	22.42	20.48	21.03	22.80	21.74	19.66
W8A	2.74	3	3	3	3	3	3	2.95	2.21	2.41	2.12
REAL-SIM	10.83	12.04	10.29	8.21	7.62	14.52	15.92	11.00	13.69	8.75	9.54

Table 3: Data sets.

DATA SET	# FEATURES	# TRAINING	# TESTING
A7A	123	16,100	16,461
W8A	300	45,546	13,699
REAL-SIM	20,958	57,848	14,461
RCV1	47,236	20,242	677,399
NEWS20	1,355,191	15,997	3,999
URL	3,231,961	12,800	3,200
WEBSPAM	16,609,143	28,000	7,000

range $\{256, 512, 1024\}$ for the first category data sets and we fix $b = 256$ for large-scale data sets with fast testing time, $\varsigma = 256$ and $\vartheta = 32$. The experiments are performed on a server with a 3.4GHZ Intel CPU and 94.5GB main memory running on a Linux platform.

Results on Small and Medium-Sized Data Sets

This subsection studies the performance of various methods on A7A, W8A, REAL-SIM and RCV1 data sets. To verify the generalization error analysis in Section , we first compare the classification performance of 1NN and 5NN using CD-Tree and other main baselines on three data sets. We set N_{min} to one and ten for 1NN and 5NN, respectively. Table 2 shows that 1) CD-Tree achieves comparable or better accuracy than state-of-the-art methods, which verifies our theoretical analysis. 2) All the main methods obtain similar classification performance for 1NN and 5NN. Thus, we fix $K = 5$ in the following experiment.

We compare our method with MMT on A7A, W8A, REAL-SIM data sets. The results are shown in Table 2. From this table, we can see that our method outperforms MMT, which verify the effectiveness of our proposed method. Because MMT runs slowly on most data sets, we do not report their results on other data sets. In our experiment, some methods run out of memory on the RCV1 data set, which contains a large number of testing instances. We randomly sample 1%, 5%, 8% and 10% testing instances for comparison. The testing error rate of various methods is listed in Table 4. From this table, we observe that: 1) With the increasing value of b, the performance of KBE rises, which is consistent with the empirical results in (Zhang et al. 2015). 2) RP+KD-Tree and KD-Tree achieve similar results, and KPQ outperforms KBE, RP+KD-Tree and KD-Tree. 3) CD-Tree improves the results of KD-Tree and RP+KD-Tree by more than 10% on medium-sized data sets and is comparable to the best results on all data sets.

Table 5 shows the testing time of various methods spent on all testing instances per small and medium-sized data sets. From this table, we observe that: 1) With the increasing value of b, the testing time of KBE rises. 2) KD-Tree is faster than BF-KNN, K-Means Tree and KPQ, while KBE and RP+KD-Tree are faster than other baseline methods, because the testing time of KBE and RP+KD-Tree does not depend on the original high dimensions. CD-Tree obtains at least four and seven times speedup over RP+KD-Tree and KBE, respectively. The results verify our computational cost analysis. 3) CD-Tree improves the classification results of KD-Tree by more than 10%, while being at least 11 times faster than the KD-Tree. Furthermore, CD-Tree is at least 100 times faster than KPQ and is three orders of magnitude faster than BF-KNN.

We next use the RCV1 data set for training time comparison. KD-Tree and K-Means Tree takes 95s and 887s for training, respectively. KBE and KPQ both take 2642s for training, while CD-Tree takes 550s for training. Thus, the training of CD-Tree is faster than that of K-Means Tree, KBE and KPQ, which is also consistent with our computational cost analysis.

Results on Very High Dimensional Data Sets

In this subsection, we evaluate the performance of various methods on three large scale data sets: NEWS20, URL and WEBSPAM. Some methods run out of memory on these three data sets, thus, we randomly sample the data for comparison. The testing error rate and testing time of various methods on three large scale data sets are listed in Figure 1. From Figure 1, we can see that: 1) CD-Tree is several times faster than KBE and RP+KD-Tree. Moreover, CD-Tree is much more accurate than KBE and RP+KD-Tree. 2) CD-Tree is three orders of magnitude faster than other baselines and achieves superior accuracy on the data sets with millions of dimensions. Finally, we conclude that CD-Tree is effective and efficient on very high dimensional settings.

Conclusion

This paper proposes a doubly approximation strategy to subsume hashing and tree-based methods, and simultaneously remedy the defect of these approaches. We also provide a data-dependent generalization error bound for our model, which reveals a new insight for the design of ANN classification algorithms on very high dimensional settings. Our empirical studies verify our theoretical studies and CD-Tree is able to precisely and efficiently scale to very high dimensional settings.

Table 4: Testing error rate (in %) of various methods on small and medium-sized data sets. "-" denotes out of memory problem of baselines.

				KBE					
Data Set	BF-KNN	KD-Tree	K-Means Tree	$b = 256$	$b = 512$	$b = 1024$	KPQ	RP+KD-Tree	CD-Tree
A7A	19.87	20.22	19.84	20.48	20.38	20.07	22.80	21.86	19.66
W8A	3	3	3	3	3	3	2.21	3	2.12
REAL-SIM	7.3	10.29	7.62	15.92	15.68	14.82	13.69	24.49	9.54
RCV1(1%)	6.72	18.88	8	20.24	18.31	13.06	10.57	21.99	7.59
RCV1(5%)	6.79	21.74	7.47	20.12	17.84	12.19	10.22	21.8	7.68
RCV1(8%)	6.93	23.37	6.68	20.21	17.74	12.24	10.31	22	7.71
RCV1(10%)	6.94	19.24	6.82	22.57	20.61	16.07	10.84	22.21	7.79
RCV1	-	-	-	22.79	21.32	16.38	-	23.10	7.88

Table 5: Testing time (in seconds) of various methods on small and medium-sized data sets. "-" denotes out of memory problem of baselines. Numbers in parentheses indicate speedup ($\times$) of CD-Tree over baselines.

				KBE					
Data Set	BF-KNN	KD-Tree	K-Means Tree	$b = 256$	$b = 512$	$b = 1024$	KPQ	RP+KD-Tree	CD-Tree
A7A	63.97 (400)	17.27 (108)	17.71 (111)	17.29 (108)	18.03(113)	22.90 (143)	17.91 (112)	17.56 (110)	0.16
W8A	242.36 (505)	16.25 (34)	16.32 (34)	41.02 (85)	42.58 (89)	59.58 (124)	47.60 (99)	14.32 (30)	0.48
REAL-SIM	23210.72 (6173)	41.29 (11)	296.72 (79)	57.15 (15)	65.64 (17)	74.00 (20)	414.85 (110)	15.51 (4)	3.76
RCV1(1%)	10743.09 (4495)	25.27 (11)	92.31 (39)	17.77 (7)	22.11 (9)	44.66 (19)	343.02 (144)	9.06 (4)	2.39
RCV1(5%)	37322.88 (4000)	103.10 (11)	552.17 (59)	73.74 (8)	93.59 (10)	169.66 (18)	1667.2 (179)	48.30 (5)	9.33
RCV1(8%)	58496.60 (4009)	187.84 (13)	880.37 (60)	100.93 (7)	149.42 (10)	267.49 (18)	2608 (179)	77.56 (5)	14.59
RCV1(10%)	70352.37 (3722)	350.30 (19)	1905.17 (101)	150.45 (8)	208.10 (11)	372.58 (20)	3260.1 (172)	85.44 (5)	18.90
RCV1	-	-	-	1838.65 (9)	2174.97 (11)	4549.34 (22)	-	882.31 (4)	204.82

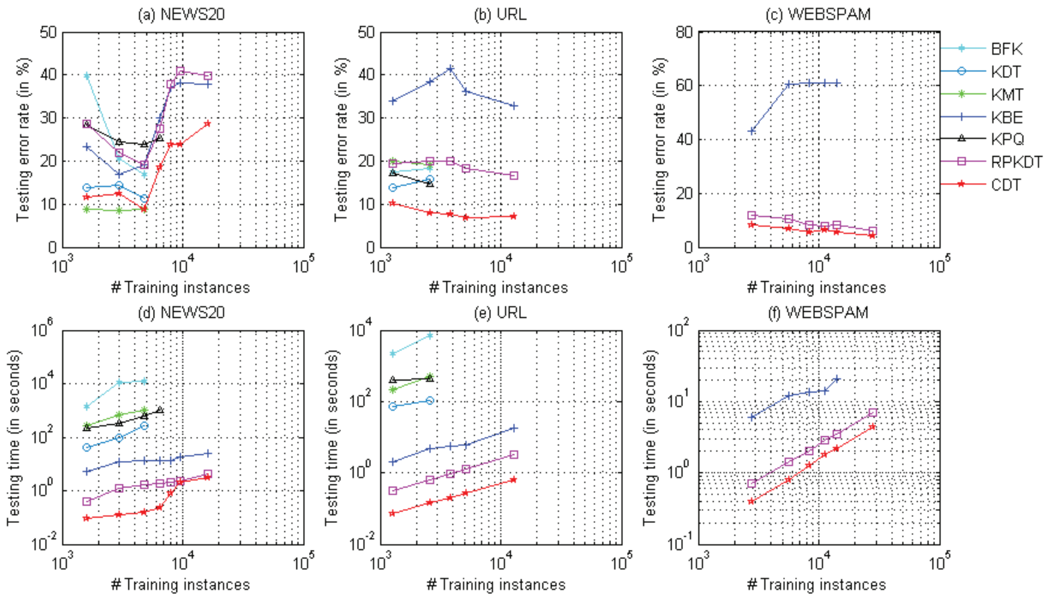

Figure 1: Testing error rate (in %) and testing time (in seconds) of various methods on 3 very high dimensional data sets. X and Y axes are in log scale. BF-KNN, KD-Tree, K-Means Tree, RP+KD-Tree and CD-Tree are abbreviated to BFK, KDT, KMT, RPKDT and CDT, respectively.

Acknowledgments

This project is supported by the DP170101628, DP150102728, DP150103071, DP180103096, DP180100106, NSFC 61232006, NSFC 61672235, FT130100746 and LP150100671.

References

Andoni, A.; Indyk, P.; Laarhoven, T.; Razenshteyn, I. P.; and Schmidt, L. 2015. Practical and optimal LSH for angular distance. In *NIPS*.

Bartlett, P. L., and Mendelson, S. 2002. Rademacher and Gaussian complexities: Risk bounds and structural results. *JMLR* 3:463–482.

Beck, A., and Teboulle, M. 2009. A fast iterative shrinkage-thresholding algorithm for linear inverse problems. *SIAM J. Imaging Sciences* 2(1):183–202.

Beyer, K. S.; Goldstein, J.; Ramakrishnan, R.; and Shaft, U. 1999. When is "nearest neighbor" meaningful? In *7th International Conference Database Theory*, 217–235.

Bingham, E., and Mannila, H. 2001. Random projection in dimensionality reduction: Applications to image and text data. In *SIGKDD*, 245–250.

Charikar, M. 2002. Similarity estimation techniques from rounding algorithms. In *Proceedings on 34th Annual ACM Symposium on Theory of Computing*, 380–388.

Du, B.; Zhang, M.; Zhang, L.; Hu, R.; and Tao, D. 2017. PLTD: patch-based low-rank tensor decomposition for hyperspectral images. *IEEE Trans. Multimedia* 19(1):67–79.

Gionis, A.; Indyk, P.; and Motwani, R. 1999. Similarity search in high dimensions via hashing. In *VLDB*, 518–529.

Gong, C.; Tao, D.; Liu, W.; Liu, L.; and Yang, J. 2017. Label propagation via teaching-to-learn and learning-to-teach. *IEEE Trans. Neural Netw. Learning Syst.* 28(6):1452–1465.

Jégou, H.; Douze, M.; and Schmid, C. 2011. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.* 33(1):117–128.

Kim, S.-J., and Boyd, S. P. 2008. A minimax theorem with applications to machine learning, signal processing, and finance. *SIAM Journal on Optimization* 19(3):1344–1367.

Li, Y.-F.; Tsang, I. W.; Kwok, J. T.; and Zhou, Z.-H. 2013. Convex and scalable weakly labeled SVMs. *JMLR* 14(1):2151–2188.

Liu, W., and Tsang, I. W. 2016. Sparse perceptron decision tree for millions of dimensions. In *AAAI*, 1881–1887.

Liu, W., and Tsang, I. W. 2017. Making decision trees feasible in ultrahigh feature and label dimensions. *JMLR* 18(81):1–36.

Liu, X.; Wang, L.; Yin, J.; Dou, Y.; and Zhang, J. 2015. Absent multiple kernel learning. In *AAAI*, 2807–2813.

Liu, W.; Shen, X.; and Tsang, I. W. 2017. Sparse embedded k-means clustering. In *NIPS*, 3280–3288.

Liu, W.; Tsang, I. W.; and Müller, K.-R. 2017. An easy-to-hard learning paradigm for multiple classes and multiple labels. *JMLR* 18(94):1–38.

Ma, J.; Saul, L. K.; Savage, S.; and Voelker, G. M. 2009. Identifying suspicious urls: an application of large-scale online learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 681–688.

Muja, M., and Lowe, D. G. 2014. Scalable nearest neighbor algorithms for high dimensional data. *IEEE Trans. Pattern Anal. Mach. Intell.* 36(11):2227–2240.

Ram, P.; Lee, D.; and Gray, A. G. 2012. Nearest-neighbor search on a time budget via max-margin trees. In *SDM*, 1011–1022.

Shen, X.; Liu, W.; Tsang, I. W.; Sun, Q.-S.; and Ong, Y.-S. 2017. Multilabel prediction via cross-view search. *TNNLS* 1–15.

Sion, M. 1958. On general minimax theorems. *Pacific Journal of Mathematics* 8(1):171–176.

Tan, M.; Tsang, I. W.; and Wang, L. 2014. Towards ultrahigh dimensional feature selection for big data. *JMLR* 15(1):1371–1429.

Tan, M.; Wang, L.; and Tsang, I. W. 2010. Learning sparse svm for feature selection on very high dimensional datasets. In *ICML*, 1047–1054.

Toh, K.-C., and Yun, S. 2009. An accelerated proximal gradient algorithm for nuclear norm regularized least squares problems. Technical report.

Wang, R.; Nie, F.; Yang, X.; Gao, F.; and Yao, M. 2015. Robust 2DPCA with non-greedy l1-norm maximization for image analysis. *IEEE Trans. Cybernetics* 45(5):1108–1112.

Wang, D.; Irani, D.; and Pu, C. 2012. Evolutionary study of web spam: Webb spam corpus 2011 versus webb spam corpus 2006. In *8th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 40–49.

Weber, R.; Schek, H.; and Blott, S. 1998. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, 194–205.

Wu, J.; Hong, Z.; Pan, S.; Zhu, X.; Cai, Z.; and Zhang, C. 2014. Multi-graph-view learning for graph classification. In *ICDM*, 590–599.

Yang, Y.; Deng, C.; Tao, D.; Zhang, S.; Liu, W.; and Gao, X. 2017. Latent max-margin multitask learning with skelets for 3-d action recognition. *IEEE Trans. Cybernetics* 47(2):1108–1112.

Zhai, Y.; Ong, Y.-S.; and Tsang, I. W. 2014. The emerging "big dimensionality". *IEEE Computational Intelligence Magazine* 9(3):14–26.

Zhang, X.; Yu, F. X.; Guo, R.; Kumar, S.; Wang, S.; and Chang, S.-F. 2015. Fast orthogonal projection based on Kronecker product. In *ICCV*, 2929–2937.

Zhuang, J.; Tsang, I. W.; and Hoi, S. C. H. 2011. A family of simple non-parametric kernel learning algorithms. *JMLR* 12:1313–1347.