國立臺灣大學電機資訊學院資訊工程研究所
碩士論文
Department of Computer Science and Information Engineering
College of Electrical Engineering and Computer Science
National Taiwan University
Master Thesis

利用正交轉換在分散式環境中尋找 K 個最近鄰居
kNN-Search with Orthogonal Transformation in the Distributed Environment

王瑞斌
Jui-Pin Wang

指導教授：林守德博士
Advisor: Shou-De Lin, Ph.D.

中華民國 103 年 7 月
July, 2014

國立臺灣大學
資訊工程研究所

碩士論文

利用正交轉換在分散式環境中尋找K個最近鄰居

王瑞斌 撰

1037

國立臺灣大學（碩）博士學位論文
口試委員會審定書

論文中文題目

論文英文題目

本論文係○○○君（○學號○）在國立臺灣大學○○學系、所完成之碩（博）士學位論文，於民國○○年○○月○○日承下列考試委員審查通過及口試及格，特此證明

口試委員：

（簽名）

（指導教授）

系主任、所長 （簽名）

（是否須簽章依各院系所規定）

致謝

這裡將簡單介紹如何利用 LaTeX 來編輯你的畢業論文，若不知道 LaTeX 是什麼或是沒有概念的話，建議你可以簡單看過放在此資料夾裡的李果正 -大家來學 LaTeX前四章內容，在下載適合的 LaTeX 整合發行套件之後（請看第 III.項），可以嘗試用剛安裝好的 LaTeX 編輯器來編譯thesis.tex這份文件，編譯的方法可以看下面第 V.項的介紹，若編譯成功，所編譯出來的 thesis.pdf 文件的應該會跟此 demo.pdf 文件一模一樣，而且沒有任何問號符號，走到這一步的話，就差不多可以開始邊學習 LaTeX 邊編輯你的畢業論文了！基本上會使用到的指令都包含在論文的的各章節裡，怎麼在論文裡寫公式或是放圖之類的就自行看 tex 檔學吧。如果有任何問題或建議可以來信與我討論，我的信箱是dran31545@gmail.com，或是到此範本Google Project裡面的Issues貼上你的問題與建議，我會盡我所能更新此範本，也歡迎大家自行重製、改良此範本並散布給他人，祝大家順利畢業！

要編輯致謝請打開acknowledgementsCH.tex

I. 此範本參考並修改自下列網站的資料：

- 如何用 LaTeX 排版臺灣大學碩士論文
  —台灣大學論文 LaTeX 樣版原創者黃子桓的教學網頁
- LaTeX 常用語法及論文範本
  —Hitripod所修改的範本，這裡參考了許多他所寫的格式和內容
- 使用 LaTeX 做出精美的論文
- XeTeX：解決 LaTeX 惱人的中文字型問題
- 台灣大學碩士、博士論文的 Latex 模板

II. 幾個有用的參考資料及網路資源：

- 李果正 -大家來學 LaTeX—建議先看完前四章
- WIKIBOOKS-LaTeX—好用的線上工具書
- Working with a .bib file using JabRef
- Using BibDesk - A short tutorial
- LaTeX for Physicists

III. 下載 LaTeX 整合發行套件，可參考TeX Collection：

1. MacTeX: For **MacOSX**，下載MacTeX.pkg
2. ProTeXt: For **Windows**，下載ISO file
3. TeX Live: For **GNU/Linux** and **MacOSX**, and **Windows**，下載ISO file
4. CTAN: The Comprehensive TeX Archive Network.

IV. 好用的程式：

- 文獻管理系統：
  1. JabRef
     可參考Working with a .bib file using JabRef或是Google及YouTube
  2. BibDesk (For Mac)
     可參考Using BibDesk - A short tutorial或是Google及YouTube
- 方程式編輯器：Daum Equation Editor (Chrome App，必須使用 Google 瀏覽器)

V. 編譯流程：

1. `xelatex thesis`
   對 thesis.tex 進行第一次 XeLaTeX 編譯，產生 thesis.pdf 以其他檔案
2. `bibtex thesis`
   對 thesis.tex 進行 BibTeX 編譯，產生 bbl 檔以及 blg 檔
3. `xelatex thesis`
   對 thesis.tex 進行第二次 XeLaTeX 編譯，產生目錄、圖表連結及參考文獻
4. `xelatex thesis`
   對 thesis.tex 進行第三次 XeLaTeX 編譯，產生參考文獻連結，完成編譯

注意！此範本使用 cite 套件，可依據你利用文獻管理系統所整理好的thesisbib.bib檔在論文最後產生參考文獻頁面，若你的系所規定要在每個章節的後面產生參考文獻，則可以用 chapterbib 套件，來對每個有附參考文獻的章節 tex 檔進行一次 BibTeX 編譯產生 bbl 檔，如範例的introduction.tex、THM.tex和EXP.tex，如果有這需要請把thesis.tex檔裡使用 cite 套件的指令利用註解符號% 來取消使用 cite 套件，並刪去出現在使用 chapterbib 套件指令前面的註解符號% 來啟動使用 chapterbib 套件

```
\usepackage{cite}
%\usepackage{chapterbib}
```

改成

```
%\usepackage{cite}
\usepackage{chapterbib}
```

再來利用註解符號% 取消會把參考文獻放在論文最後的指令

```
\bibliographystyle{unsrt}
\addcontentsline{toc}{chapter}{\bibname}
\bibliography{thesisbib}
```

改成

```
%\bibliographystyle{unsrt}
%\addcontentsline{toc}{chapter}{\bibname}
%\bibliography{thesisbib}
```

再把用來輸入章節檔案的 \input 指令改成 \include 指令

```
\input{introduction}   =>   \include{introduction}
\input{THM}            =>   \include{THM}
\input{EXP}            =>   \include{EXP}
```

最後記得在每個有附參考文獻的章節加上產生參考文獻的指令，即在introduction.tex、THM.tex和EXP.tex三個檔案裡最後啟動下面兩行指令

```
%\bibliographystyle{unsrt} => \bibliographystyle{unsrt}
%\bibliography{thesisbib}  =>  \bibliography{thesisbib}
```

而編譯時則需要對有附參考文獻的introduction.tex、THM.tex和EXP.tex各做一次 BibTeX 編譯，編譯流程如下

1. `xelatex thesis`
   對 thesis.tex 進行第一次 XeLaTeX 編譯，產生 thesis.pdf 及其他檔案
2. `bibtex introduction`
   對 introduction.tex 進行 BibTeX 編譯，產生 bbl 檔以及 blg 檔
3. `bibtex THM`
   對 THM.tex 進行 BibTeX 編譯，產生 bbl 檔以及 blg 檔
4. `bibtex EXP`
   對 EXP.tex 進行 BibTeX 編譯，產生 bbl 檔以及 blg 檔
5. `xelatex thesis`
   對 thesis.tex 進行第二次 XeLaTeX 編譯，產生目錄、圖表連結及參考文獻

6. `xelatex thesis`
   對 thesis.tex 進行第三次 XeLaTeX 編譯，產生參考文獻連結，完成編譯

VI. 補充說明與注意事項：

- 口試委員會審定書：
  請到台大圖書館網頁的電子論文服務下載論文格式範本，並修改成正確的格式，也可到此範本所在資料夾的cert.doc修改。當然你也可以利用 LaTeX 來編輯，你只要填好ntuvars.tex檔的資料，並去除在 thesis.tex 裡下面這行的註解符號%

  ```
  %\makecertification
  ```

  編譯完後就可以產生審定書格式。口試通過後，請把已經簽名的審定書掃描成 pdf 檔，再取代原本的cert.pdf，即可放上已簽名的審定書。處理審定書出現的指令在 thesis.tex 裡

  ```
  %----------- generate the certification ...
  %\makecertification
  %----------- includepdf by using package ...
  \addcontentsline{toc}{chapter}{口試委員會審定書}
  \includepdf[pages={1}]{cert.pdf}
  ```

- 浮水印：
  資料夾已經附上浮水印檔案了，若學校有更改，到請到台大圖書館網頁的電子論文服務下載pdf 格式的浮水印到此範本所在資料夾。若要開啟關閉浮水印功能，即自行刪去或加上下面位於thesis.tex指令的註解符號%

  ```
  %\CenterWallPaper{0.174}{watermark.pdf}
  %\setlength{\wpXoffset}{6.1725cm}
  %\setlength{\wpYoffset}{10.5225cm}
  ```

- 單面印刷與雙面印刷：
  此範本為單面印刷，若論文頁數超過 80 頁，依規定需要用雙面印刷，此時只需把 thesis.tex 裡的

  ```
  \documentclass[a4paper, 12pt, oneside]{book}
  ```

  改成

  ```
  \documentclass[a4paper, 12pt, twoside]{book}
  ```

- 如何加入附錄?

在thesis.tex裡，依需求選擇 input 或 include，刪去% 符號來輸入附錄章節

```
%----------- Input your appendix here  -----------
%\input{AppendixA}
%or %chapter cite  == \include
%\include{AppendixA}
```

在章節檔 AppendixA.tex 裡，開頭打

```
\chapter{First appendix title}
```

即可，以此類推。

- 系上規定論文圖表須全部放到最後獨立出來的章節，且章節不出現在目錄中：
  在thesis.tex裡，依需求選擇 input 或 include，刪去% 符號來輸入圖表章節

```
%----------- Input your Figure chapter here  -----------
%\input{EndFigTab}
%chapter cite  == \include
%\include{EndFigTab}
```

在章節檔EndFigTab.tex裡有範例和說明可供參考，要注意正文的圖表和附錄的圖表要分清楚，即在EndFigTab.tex內

```
\renewcommand{\thefigure}{\arabic{chapter}.
\arabic{figure}}
\renewcommand{\thetable}{\arabic{chapter}.
\arabic{table}}
%--- Input your main figures and tables here  ---
```

這幾行之後章節計數器格式已切換為 1...9，放正文的圖表，

```
\renewcommand{\thefigure}{\Alph{chapter}.
\arabic{figure}}
\renewcommand{\thetable}{\Alph{chapter}.
\arabic{table}}
%--- Input your appendix figures and tables here  ---
```

這幾行之後章節計數器格式已切換為A...Z，放附錄的圖表。另外要取消圖表的浮動功能，才能讓圖表按照指令出現順序排好，即把平常使用的圖表指令

```
\begin{figure}[htb]
...
\begin{table}[htb]
```

改成

```
\begin{figure}[!]
...
\begin{table}[!]
```

剩下的只要注意章節圖表的計數器設定即可。`\ref` 和 `\label` 指令可以在此圖表章節與正文章節使用。

- 如果我想要修改 margin(文字邊界) 的話，可以從哪裡下手呢?
  請打開ntu.sty修改下面這行的上下左右參數即可：

```
\RequirePackage[top=3cm,left=3cm,bottom=2cm,right=3cm]
{geometry}
```

- 我想引用 Twomey (1974): Pollution and planetary albedo 這篇論文，如何用 `\cite` 引用它的時候在內文顯示 Twomey (1974) [編號] ？
  建議使用 natbib 套件，參考資料如下：
  LaTeX/Bibliography Management
  Overview of Bibtex-Styles
  Reference sheet for natbib usage
- XeTeX ：
  此範本中文字體使用XeTeX 轉換，細節請參考Hitripod寫的XeTeX：解決 LaTeX 惱人的中文字型問題。
- 如何輸入英文 ‘單引號’ 和 “雙引號” 以及不同長度的破折號？
  可以參考李果正 -大家來學 LaTeX第 17 頁針對標點符號的遊戲規則，範例如下，輸入以下指令：

```
`單引號'\\
``雙引號''\\
-hyphen\\
--en-dash\\
---em-dash\\
```

則顯示：
‘單引號’
“雙引號”

-hyphen
–en-dash
—em-dash

中文摘要

請打開並編輯abstractCH.tex

關鍵字：壹、貳、參、肆、伍、陸、柒

Abstract

Open and edit abstractEN.tex

Key words:A, B, C, D, E, F, G

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Due to the advance of technology, query by similarity for various kinds of datasets among distributed machines like mobile devices becomes an increasingly common and important task. In this distributed environment where a large amount of local machines are involved in computation and storage, our goal in most case is to minimize the amount of transmission cost during finding the answer from these machines. Therefore, this paper aims to generalize the current state-of-the-art on distributed pattern matching from only workable for "time series" datasets to more types of dataset such as "images" and "music" while significantly reducing the communication cost.

Consider the first scenario in which, through a program installed on mobile devices, a company about data mining could obtain some unusual and potentially profitable images and want to know quickly whether a similar image exsists in someone's cell phone or tabelet. To do so, it is required to find the similar images among a large amount of distributed devices for the final answer. Moreover, this company could construct a platform which allows users to search among the images of other cell phones as long as they also provide their images. With this platform, each user could use their images to find the similar images from other cell phones. On the other hand, beside of the images, this platform could also deal with other type of data like music.

Since this paper is generalized from the work [1] called LeeWave and our work [2] called MsWave, our model could also apply to those scenarios such as the abnormal detection and work on the time series datasets. Although our discussions in the following

chapters would focus on the problem of handling a single query and finding the k nearest neighbors among these distribued devices, we could deal with multiple queries and the k farthest neighbors like MsWave only with a little modifications.

However, to design a framework which could handle these scenarios would requires addressing some challanges. First, our goal is to find the *exact* k nearest neighbors instead of approximate k nearest neighbors. So, we find the answer according to their exact similarity. Second, the number of distributed devices could be very large. It would cause huge communication cost to get our answers from them. Third, since there could be a large amount of users on that platform, we also have to handle a extremely long sequence of queries. Forth, to achieve the cost saving in communication, we have to extract valuable information from queries to send to machines instead of the whole query. It is a important question to define the valuable information and then discard impossible candidates only with the help of these information. Finally, since in many situations there are bandwidth limitations and concerns of energy consumption as well as communication cost, it is crucial to design an framework that needs as little communication cost among distributed machines as possible. This paper would overcome these problems on the basis of the ideas proposed by LeeWave.

Although the state-of-the-art method, LeeWave in [1] could solve some of the challenges above, there are still some problems for it to deal with. The most important problem is that the way LeeWave to prune the candidates is not effective when the datasets are not time series. Since the bounds of LeeWave comes from the Haar wavelet transformation, which is a kind of transformation usually applied on time series, these bounds would be no longer tight when the data we deal with is not time series. Another problem is that the cost of the pruning procedure in LeeWave grows linearly with the number of total instances in these distrubited devices. When the number of instances is large, its communication cost would be too expensive.

We summarize our main contributions as follows:

1. We propose a general communication-efficient framework which could find kNN and kFN instances given multiple queries for various types of datasets which are

distributed in a large amount of devices.

2. In the part of methodology, based on the ideas of upper bounds and lower bounds of similarity in [1], we derive new bounds with the help of the orthogonal transformation. And these new bounds enhance the power of pruning is the main source to achieve the goal of cost saving in communication.

3. Since it would be expensive to find the threshold used in the pruning procedure if the number of total instances in these machines is large, we give a method that could make this cost be indenpendent of this number.

4. We propose a method to estimate the communication cost which allows us to dynamically adjust how much information we need to send for the current query according to the history of the past queries.

5. We conduct extensive experiments for various types of datasets to demonstrate that our model could achieve the goal of saving communication cost.

1.1 Thesis Overview

We organize this paper as follows: In the chapter 2, we discuss about the related papers and their limitations. Then, in the chapter 3, we propose the details of our framework. In the chapter 4, we provide the results of the experiments for the datasets of images, music, and time series. Finally, in the chapter 5, we discuss the results and our future work.

Chapter 2

Related Works

In this chapter, we talk about the works which are related to these scenarios.

2.1 Concurrent Processing

Concurrent Processing (CP) [3] is the baseline of this problem. First, the server would send the whole query to every local machine. Each machine calculates the distance between this query and every instance on it. Then, every machine return the top k instances with the lowest distance back to the server. By these $m \times k$ instances, the server could know which k instances are the k nearest neighbors of the query. We could easily notice that much communication cost is waste in this framework.

2.2 Probabilistic Processing

There is another method named Probabilistic Processing (PRP) in [3] that we could take it as an improvement of CP. The most important characteristic of PRP is that it guarantee to find the answers in two rounds with less communication cost than CP. In the first round, the server also sends the whole query to every local machines. But, instead of return the top k instances like CP does, each machine only return the top $\left\lfloor \frac{k}{m} \right\rfloor + 1$ instances back to the server where m is the number of total local machines. With these

$m \times (\lfloor \frac{k}{m} \rfloor + 1) = k + m$ instances, the server could prune some machines which are impossible to obtain the final k answers and then ask the other machines for the final answers in the second rounds. Although PRP might able to prune some machines in the first round, it still spends too much cost in the second round.

2.3 LeeWave

Now let's talk about the state-of-the-art method, LeeWave [1], which is the starting point of our framework. The spirit of LeeWave is to iteratively pruning impossible candidaites until only k instances left by transforming a raw feature vector into an error tree like TODO with the help of the Haar wavelet transformation. Although the total number of coefficients in an error tree would be equal to length of the raw feature vector, the coefficients at the upper levels would be more important than those in the lower levels. The importance defined here is the chance to contibute more to the final Euclidean distance, And it could also be observed from the way to calculate the Euclidean distance from the error trees, the higher level the coefficient is, the heavier weight it has to multiply.

Once we have the importance of the coefficients, LeeWave sends coefficents according to their levels in the error tree transformed from the query q, from upside to down. In each round, LeeWave would send those coefficients in one level of the tree to each candidate machines. Then, these machines would return some information that allows the server to compute the bounds between q and the instances in these machines. With the help of these bounds, the server could prune some instances that they are impossible to be the final answers. If there are exactly k instances left after pruning, then we just achive our goal to find the kNN/kFN. Otherwise, the server would send the next level and repeat the pruning process until finding the answers or sending every level of this tree.

2.4 MsWave

MsWave [2] is our previous work which is also extended from LeeWave but toward a different direction. While the main contibution of this paper is to generalize the ideas

of LeeWave to various types of datasets, the contribution of MsWave focus on how to modify the bounds in LeeWave for multiple queries and finding k farthest neighbors. We could say that LeeWave is a special case of MsWave for a single query only with some slightly difference. Since the foundation of MsWave and this paper is the same, we could easily apply our all improvements in this paper to the scenarios in MsWave. That is, we could also solve the cases of multiple queries and find k farthest neighbors with our new bounds. Because there are detailed discussions about the situation of multiple queries and differences between finding kNN and kFN, we only conduct the experiments for the case of a single query and finding kNN.

2.5 Others

Due to the population of the P2P paradigm [4, 5], there are some methods which use the distributed computing to do similarity search over a set of machines. For example, [6–8] are P2P approaches proposed for similarity search, but they are designed for one dimensional data, not high dimensional data. SWAM [9] is a family of *Small World Access Methods*, whose goal is to build a network topology which could collect peers with similar content. However, in this framework, each peer could only obtain a single data, which is not suitable with our problem for a large amount of data. VBI-tree and SkipIndex both rely on tree-based approaches that could not scale when the dimensions of data are high. [10] leverage on LSH-based approaches for similarity search over structured P2P network for high dimensional data. Nevertheless, it only provides the approximate results, not exactly solution. But the biggest difference between these papers and our framework is the setting of the network. While their framework apply on a more general P2P system, our setting is that every local machine is only able to communicate with a single server.

Chapter 3

Methodology

In this chapter, we give the details about our framework.

3.1 Problem Setup

There are a query set $Q = \{q_1, q_2, ..., q_T\} \subset \mathbb{R}^D$ at the server P and a dataset $X_i \subset \mathbb{R}^D$ on each local machine M_i, $\forall i = 1, 2, ..., m$. For each coming query q_t, we want to find its k_{th} nearest neighborhood among these distributed datasets while reducing the transmission cost between P and each M_i.

3.2 Overview of Our Framework

In this section, we describe an overview of our framework. Then, we will give the details about the framework in the following sections.

Figure 3.1 is the overall flow on our framework. There are two main phases in our framework. For each X_i, the first phase (the red part) only needs to be done for once. On the other hand, we need to run the second phase (the cycle) for each new query q_t.

The first phase is an preprocessing procedure for the second phase. Its goal is to improve the performance of pruning in the second phase. We will prove in the section 3.5.2 that this pruning power is highly correlated to the distribution of the norm of the feature vectors. As a result, each M_i would learn an othorogal matrix W_i for its X_i to fit our de-

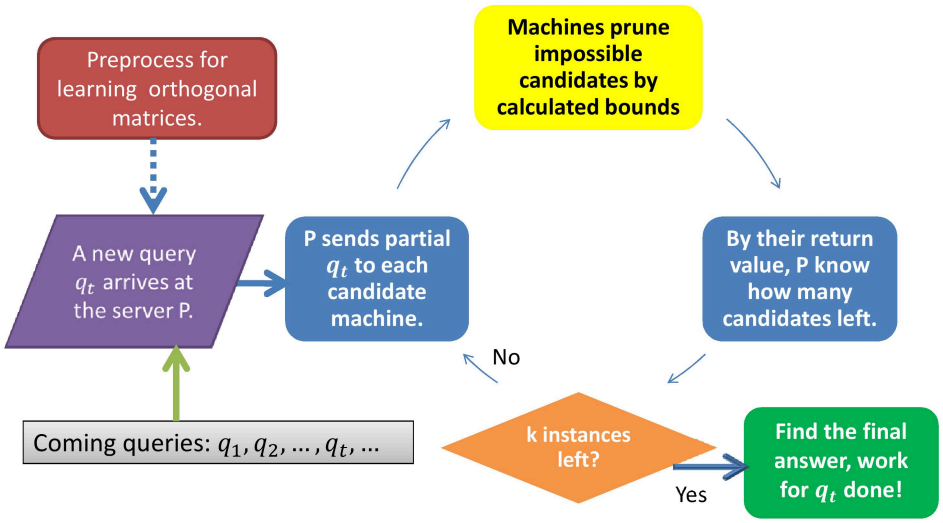

Figure 3.1: The overall flow of our framework.

sired distribution and then send each W_i back to P. We can notice that this phase is only dependent on X_i and independent of q_t. Therefore, we only need to do the first phase for once. we give the details about how to learn W_i, how to send it back to P in the section **??**.

The second phase is the main procedure of our framework. Note that P have already got W_i for each X_i in the beginning of the second phase. For each coming query q_t, we iteratively prune some candidates which are impossible to be the kNN of q_t to reduce the search space until there are only k candidates left.

To prune candidates iteratively, we divide the second phase into several rounds. For each round j, we use a function $S_i(q_t, j, \theta_t; W_i)$ to generate the values for trasmitting from P to M_i, where S_i is the importance-selecting function of M_i and θ_t is its parameters for deciding how many elements we need to send. (We put the details of S_i at the section 3.7.1.) By these values, each M_i could calculate the bounds between each candidate x_l and q_t. With these bounds, P would be able to determine which candidates are definitely not our answers and then disregards them in the following rounds. By these pruning, we could

achieve the goal of saving transmission cost from avoiding to consider the unnecessary candidates.

Note that we could use the square of the Euclidean distance instead of the origin Euclidean distance to find kNN as it is non-negative. So we will use the former one in our framework.

3.3 First Phase

In this section, we talk about the first phase of our framework.

3.3.1 Definition of Ortohogonal Transformation

The most important part of the first phase is the orthogonal transformation. So we give its definition as below.

Definition 1. A matrix $W \in \mathbb{R}^{D\times D}$ is orthogonal if whose columns and rows are orthogonal vectors, i.e.

$$W^T W = W W^T = I$$

where I is the identity matrix.

3.3.2 Property of Orthogonal Transformation

There are some useful properties in orthogonal transformation. Here we introduce the one that we will use in the derivation of our bounds.

Property 1. Let $x, y \in \mathbb{R}^{D\times 1}$, and $W \in \mathbb{R}^{D\times D}$ be an orthogonal matrix. Then,

$$Dist(x, y) = \sum_{d=1}^{D} (x[d] - y[d])^2 = \sum_{d=1}^{D} (W[d, :]x - W[d, :]y)^2 = Dist(Wx, Wy)$$

where $W[d, :]$ is the d_{th} row of W.

We will use this important property in the section 3.4.4.

3.4 Enhance the Bounds by the Orthogonal Transformation

In the section 3.2, we mentioned that the first phase is an auxiliary step for the second phase. After the introduction of the orthogonal transformation, we introduce this powerful tool into the first phase in our framework.

3.4.1 The Goal of the First Phase

Our goal in the first phase is to reduce the ranges of the bounds used in the second phase. Since we will use a threshold to prune the impossible candidates according to their bounds in the pruning procedure, the ranges of the bounds would be one of the most influential factor of the pruning power. It is easy to understand that when the ranges of their bounds is short, more candidates would be pruned than those with long ranges of the bounds. In other words, the shorter the range of the bound, the higher chance this candidate would be pruned if it is not our final answer of kNN.

Moreover, we would prove in the section 3.4.5 that to reduce the range of the bounds, the final goal of the first phase would become to learn an orthogonal transformation W which could make each raw feature vector $x \in \mathbb{R}^D$ (the orange vector in the figure 3.2) transformed to be Wx (the red vector in the figure 3.2) which would make the absolute values of elements in the forward part of the vector much larger and those in the later part much smaller.

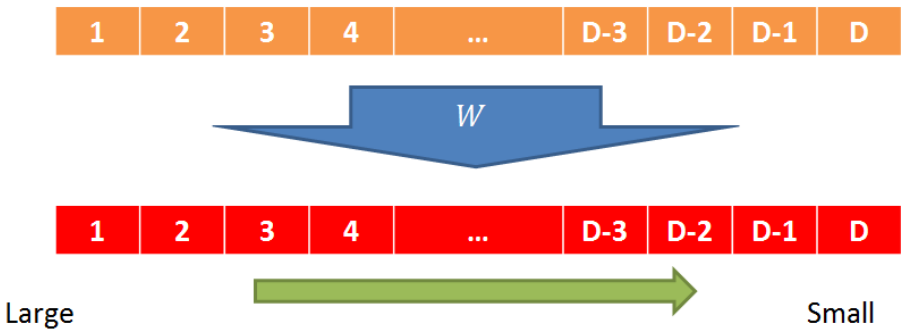

Figure 3.2: The goal of the first phase.

3.4.2 Definition of the Bounds

Before we explain why we need the orthogonal transformation to enhance our bounds, we need to define our bounds for the pruning. Recall that given a query q_t, our goal is to find its kNN in these distributed datasets X_i. Intuitively, we need to calculate the square of the Euclidean distance $Dist(q_t, x), \forall x \in \cup_i X_i$. However, to cacluate $Dist(q_t, x)$, we need to send the whole q_t to the local machines or send the whole x to P, which causes a huge transmission cost. Therefore, instead of the exact value of Euclidean distances, our propsed framework uses bounds to confine this distance for finding the kNN.

Definition 1. $\forall x, y \in \mathbb{R}^D$, a lower bound $LB(x, y)$ and a upper bound $UB(x, y)$ must satisfy the following inequation:

$$LB(x, y) \leq Dist(x, y) = \sum_{d=1}^{D} (x[d] - y[d])^2 \leq UB(x, y)$$

3.4.3 Relation Between the Norms and the Bounds

To acheive the goal of reducing the length of the ranges of the bounds, we could look into the derivation of the bounds. Suppose there are two vectors $x, y \in \mathbb{R}^D$, but we could only observe the first s dimensions of x and $\sum_{d=s+1}^{D} x[d]^2$, which is the square of two norm of the unobserved part $x[s+1:D]$. In the section 3.5.2, we derive the bounds as

$$\begin{aligned} LB(x, y) &= \sum_{d=1}^{s} (x[d] - y[d])^2. \\ UB(x, y) &= \sum_{d=1}^{s} (x[d] - y[d])^2 \\ &+ \sum_{d=s+1}^{D} x[d]^2 + \sum_{d=s+1}^{D} y[d]^2 \\ &+ 2 \times \sqrt{\sum_{d=s+1}^{D} x[d]^2 \times \sum_{d=s+1}^{D} y[d]^2}. \end{aligned}$$

Therefore, we could get the length of the range by the substraction.

$$\begin{aligned} Len &= UB(x,y) - LB(x,y) \\ &= \sum_{d=s+1}^{D} x[d]^2 + \sum_{d=s+1}^{D} y[d]^2 \\ &+2 \times \sqrt{\sum_{d=s+1}^{D} x[d]^2 \times \sum_{d=s+1}^{D} y[d]^2}. \end{aligned}$$

Since the term $\sum_{d=s+1}^{D} x[d]^2$ is given, all we could do to reduce the length is to make use of the $\sum_{d=s+1}^{D} y[d]^2$ term. If we could reduce $\sum_{d=s+1}^{D} y[d]^2$, the term $\sum_{d=s+1}^{D} x[d]^2 \times \sum_{d=s+1}^{D} y[d]^2$ would also decrease and then make Len smaller. Therefore, our goal now becomes to make the term $\sum_{d=s+1}^{D} y[d]^2$ as small as possible, which is the square norm of the vector $y[s+1:D]$.

Note that the lower (upper) bounds is non-decreasing (non-increasing) as s becomes larger and $LB(x,y) = UB(x,y)$ when $s = D$. This means that LB and UB would be exactly equal to $\sum_{d=1}^{D} (x[d] - y[d])^2$ eventually.

To reduce the length of the ranges Len for each s, we hope to make $\sum_{d=s+1}^{D} y[d]^2$ as small as possible. However, since the feature vector y is given from datasets, the value of $\sum_{d=s+1}^{D} y[d]^2$ is already determined when s is given. As a result, we introduce the orthogonal transformation to achieve this goal.

3.4.4 Equivalent Bounds After Transformation

From the section 3.3.2, we know the distance of two vectors won't be changed after an orthogonal transformation. Now we use this property to achieve our goal to reduce the length of the ranges Len given s.

Given an orthogonal transformation $W \in \mathbb{R}^{D \times D}$, we have $Dist(x,y) = Dist(Wx, Wy)$. This means that the bounds we derivated before could also be the bounds for $Dist(Wx, Wy)$. That is,

$$LB(x,y) \leq Dist(x,y) = Dist(\hat{x}, \hat{y}) \leq UB(x,y)$$

where $\hat{x} = Wx$.

This also means that we could use the same way to derivate the lower bounds and

upper bounds for $Dist(Wx, Wy)$ and these bounds are also the bounds for $Dist(x, y)$. That is,

$$LB(\hat{x}, \hat{y}) \leq Dist(x, y) = Dist(\hat{x}, \hat{y}) \leq UB(\hat{x}, \hat{y})$$

So, we could use the bounds $LB(\hat{x}, \hat{y}), UB(\hat{x}, \hat{y})$ to confine $Dist(x, y)$ and the length of range we want to reduce becomes

$$\begin{aligned} \hat{Len}(W) &= UB(\hat{x}, \hat{y}) - LB(\hat{x}, \hat{y}) \\ &= \textstyle\sum_{d=s+1}^{D} \hat{x}[d]^2 + \sum_{d=s+1}^{D} \hat{y}[d]^2 \\ &+2 \times \sqrt{\textstyle\sum_{d=s+1}^{D} \hat{x}[d]^2 \times \sum_{d=s+1}^{D} \hat{y}[d]^2}. \end{aligned}$$

which becomes a function of W.

As a result, instead of trying to reduce $\sum_{d=s+1}^{D} y[d]^2$ which is impossible as we mentioned in the section 3.4.3, our goal becomes to reduce $\sum_{d=s+1}^{D} \hat{y}[d]^2$ with the help of W.

3.4.5 Reduction of the Norm with Orthogonal Transformation

For $y \in \mathbb{R}^{D \times 1}$ and $W \in \mathbb{R}^{D \times D}$, given s, we want to reduce $\sum_{d=s+1}^{D} \hat{y}[d]^2$ as much as possible. However, since s is unknown while deciding W in the first phase of our framework, we have to handle all possible values which s could be. Moreover, since $\sum_{d=1}^{D} \hat{y}[d]^2$ is equal to $\sum_{d=1}^{D} y[d]^2$, which is independent with W, if the $\sum_{d=s+1}^{D} \hat{y}[d]^2$ decreases with some W, the term $\sum_{d=1}^{s} \hat{y}[d]^2$ must increase. Here, we use a more general strategy to deal with these problems.

We could look the term $\sum_{d=s+1}^{D} \hat{y}[d]^2$ from a different angle. Actually, this term is the square norm of the latter part of the vector $\hat{y}$. Therefore, although $\sum_{d=1}^{D} \hat{y}[d]^2$ is a constant for W, we could reduce the square norm of the latter part by increasing the forward part of it. In other words, we move the norm of the latter part of y to its forward part. To accomplish it, we design an objective function and then optimize this function to find our

ideal W.

$$f(W;y) = \sum_{d=1}^{D} w_d \times \hat{y}[d]^2 = \sum_{d=1}^{D} w_d \times (W[d,:]y[d])^2 \tag{3.1}$$

where $w_d = d, \forall d = 1, 2, \ldots, D$..

Because w_d would give the larger penalty as d increases, the elements in the latter part of $\hat{y}$ would be forced to become small while minimizing this objective function. This is exactly our goal to reduce $\sum_{d=s+1}^{D} \hat{y}[d]^2$. Therefore, our question becomes how to optimize this objective function with the constraints that W must be an orthogonal matrix.

3.4.6 Optimization over Orthogonal Constraints

Finally, we could introduce this concept of reducing the norms into our framework. In the first phase, we solve the following optimization problem to learn an orthogonal matrix W_i for each machine Mi.

$$\begin{aligned} \underset{W}{\text{minimize}} \quad & F_i(W) \\ \text{subject to} \quad & W^T W = W W^T = I \end{aligned} \tag{3.2}$$

where

$$F_i(W) = \sum_{x \in X_i} f(W;x) = \sum_{x \in X_i} \sum_{d=1}^{D} d \times (W[d,:]x[d])^2 \tag{3.3}$$

.

This is an optimization problem with constraints that its solution must be an orthogonal matrix. We could solve it efficiently with the help of the package from [11] as long as we have its gradient.

After we get the optimal W_i^* for each M_i, we send these matrices back to the server P. Since the learning of W_i^* is independent with the queries in the future, we only have to go through the procedure of learning W_i^* for once if X_i doesn't change too much. Then our first phase is done.

3.4.7 Reduce the Cost of Sending Matrices

However, the cost to sending an orthogonal matrix $W \in \mathbb{R}^{D\times D}$ is $D \times D$, which is too expensive. Therefore, in this section, we prove that we could reduce this cost to $\frac{D\times(D-1)}{2}$.

First, we introduce a lemma which would be used in our proof.

Lemma. For a vector $\mathbf{x} = (x_1, x_2, \ldots, x_D) \in \mathbb{R}^D$, we could use a spherical coordinate system with a radial coordinate r and $D-1$ angular coordinates $\phi_1, \phi_2, ..., \phi_{D-1}$ to represent it as

- $x_1 = r\cos(\phi_1)$
- $x_2 = r\sin(\phi_1)\cos(\phi_2)$
- $x_3 = r\sin(\phi_1)\sin(\phi_2)\cos(\phi_3)$
- $\vdots$
- $x_{D-1} = r\sin(\phi_1)\ldots\sin(\phi_{D-2})\cos(\phi_{D-1})$
- $x_D = r\sin(\phi_1)\ldots\sin(\phi_{D-2})\sin(\phi_{D-1})$

where $\phi_i \in [0, \pi],\ \forall i = 1, 2, ..., D-2$ and $\phi_{D-1} \in [0, 2\pi)$. ■

Since these row vectors $w_1, w_2, w_3, \ldots, w_D$ in W are orthonomal vectors, their r in this lemma is 1, which allows us to represent an orthonomal vector in $\mathbb{R}^D$ with $D-1$ parameters ($\phi_1, \phi_2, ..., \phi_{D-1}$ in the lemma). Moreover, since they are orthogonal to each other ($w_i w_j^T = 0,\ \forall i \neq j$), our goal becomes how to represent D vectors in $\mathbb{R}^D$ which are orthogonal to each other.

Now we are ready to give the proof by mathematical induction.

Theorem. Given D orthonomal vectors $w_1, w_2, w_3, \ldots, w_D \in \mathbb{R}^D$, we could use $\frac{D\times(D-1)}{2}$ parameters to represent them.

Proof. *Base case*:

If $d = 2$, we could use a single parameter θ for w_1, w_2 as

$$\begin{aligned} w_1 &= (\cos\theta, \sin\theta) \\ w_2 &= (\cos(\theta + \frac{\pi}{2}), \sin(\theta + \frac{\pi}{2})) \end{aligned} \tag{3.4}$$

Therefore, the total number of parameters here is 1 and equal to $\frac{2\times(2-1)}{2}$.

To be clear, we give one more base case here.

Base case-2:

When $d = 3$, we need to represent three vectors $w_1, w_2, w_3 \in \mathbb{R}^3$.

In the beginning, by the lemma above, we could use $3 - 1 = 2$ parameters to represent w_3.

Then, since w_1 and w_2 are orthogonal to w_3, they must lie in the plane whose normal vector is w_3. Since we already have w_3 (with two parameters), this plance is fixed. All we need to do is to decide w_1 and w_2 on this $\mathbb{R}^2$ plane. By projecting the x axis in the $\mathbb{R}^3$ to this plane, we could build a Cartesian coordinate system in two dimensions on this plane. And we know that we could use one parameter to decide w_1 and w_2 in an $\mathbb{R}^2$ plane from the base case above.

So, the total number of parameters for $\mathbb{R}^3$ case is $2 + 1 = 3$ and equal to $\frac{3\times(3-1)}{2}$

Inductive hypothesis:

Suppose the theorem holds for all values d up to some k, $k > 3$

Inductive step:

Let $d = k+1$, we need to represent $k+1$ vectors $w_1, w_2, w_3, \ldots, w_{k+1} \in \mathbb{R}^{k+1}$. We could use similar way like $d = 3$ to do it.

First, by the lemma above, we could use $(k + 1) - 1 = k$ parameters to represent w_{k+1}.

Then, our problem becomes to decide $w_1, w_2, \ldots, w_k$ on this hyperplane whose normal

vector is w_{k+1}. From the inductive hypothesis, we know it needs $\frac{k\times(k-1)}{2}$ parameters.

So, the total number of parameters for $\mathbb{R}^{k+1}$ case is

$$k + \frac{k \times (k-1)}{2} = \frac{k \times (k+1)}{2} \ldots \blacksquare$$

3.5 Second Phase

Now we start to discuss the second phase of our framework. In this section, we talk about how to prune the candidates if we already have bounds. Note that this mechanism is the most crucial part to achieve our goal to save the transmission cost.

3.5.1 Pruning the Candidates with the Bounds

For the query q_t, if we already know $LB(q_t, x)$ and $UB(q_t, x)$ $\forall x \in \cup_i X_i$, we could use the k_{th} smallest upper bounds and directly prune those instances whose lower bounds are higher than this value thr. I.e., we want to prune

$$\{x \,|LB(q_t, x) > thr, \forall x \in \cup_i X_i\}$$

where thr is the k_{th} largest $UB(q_t, x)$ $\forall x \in \cup_i X_i$. The following figure 3.3 is an example of how we prune impossible instances.

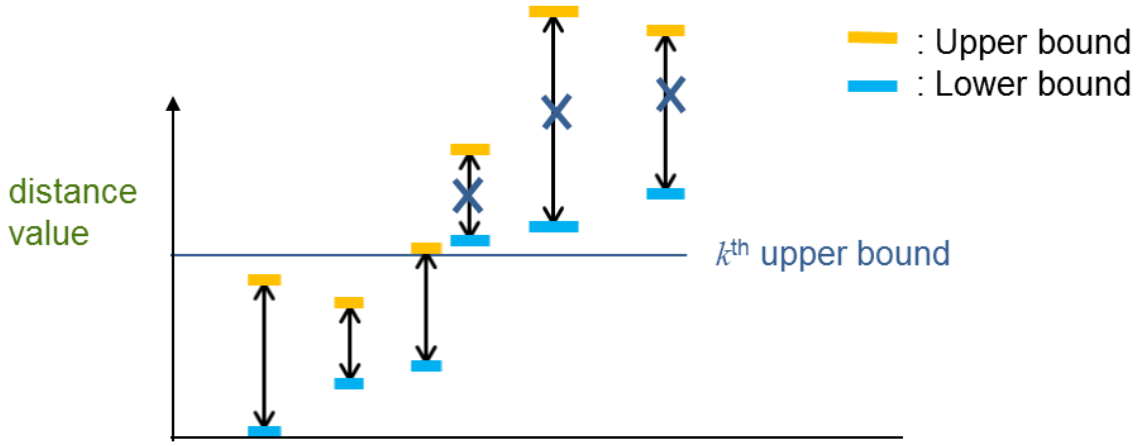

Figure 3.3: The procedure of pruning impossible candidates.

We will talk about the details of calculation these bounds and finding the threshold thr

in the section 3.5.4.

3.5.2 Derivation of our Bounds

Suppose there are two vectors $x, y \in \mathbb{R}^D$, we know the square of their Euclidean distance is

$$Dist(x, y) = \sum_{d=1}^{D} (x[d] - y[d])^2 \tag{3.5}$$

However, if we could only observe the first s dimensions of x, we could decompose their distance as

$$\begin{aligned} Dist(x, y) &= \sum_{d=1}^{D} (x[d] - y[d])^2 \\ &= \sum_{d=1}^{s} (x[d] - y[d])^2 + \sum_{d=s+1}^{D} (x[d] - y[d])^2 \end{aligned} \tag{3.6}$$

Since the first component of (3.6) is already known, all we need to do is to deal with the second term. Therefore, we further expand the second term as below:

$$\sum_{d=s+1}^{D} (x[d] - y[d])^2 = \sum_{d=s+1}^{D} x[d]^2 + \sum_{d=s+1}^{D} y[d]^2 - \sum_{d=s+1}^{D} 2 \times x[d] \times y[d].$$

By this analysis, we find the final term $\sum_{d=s+1}^{D} x[d] \times y[d]$ is the inner product between two partial vector $x[s+1 : D]$ and $y[s+1 : D]$, which could be approximated by Cauchy–Schwarz inequality

$$\sum_{d=s+1}^{D} x[d] \times y[d] \leq \sqrt{\sum_{d=s+1}^{D} x[d]^2 \times \sum_{d=s+1}^{D} y[d]^2}. \tag{3.7}$$

After combing with (3.6) and (3.7), we derive the bounds as

$$LB(x,y) = \sum_{d=1}^{s} (x[d] - y[d])^2. \tag{3.8}$$

$$\begin{aligned} UB(x,y) &= \sum_{d=1}^{s} (x[d] - y[d])^2 \\ &+ \sum_{d=s+1}^{D} x[d]^2 + \sum_{d=s+1}^{D} y[d]^2 \\ &+ 2 \times \sqrt{\sum_{d=s+1}^{D} x[d]^2 \times \sum_{d=s+1}^{D} y[d]^2}. \end{aligned} \tag{3.9}$$

We could notice that the calculation of the bounds only needs the first s dimensions of x and $\sum_{d=s+1}^{D} x[d]^2$. Therefore, we only need one more number $\sum_{d=s+1}^{D} x[d]^2$ to get the bounds for the unobserved part $x[s+1:D]$.

3.5.3 Calculation of our Bounds

After the derivation of the bounds, we describe the procedure of cacluating them in our framework.

For the query q_t, at the first round (i.e. $j = 1$), P sends the first s_1 dimensions of q_t and $\sum_{d=s_1+1}^{D} q_t[d]^2$ to each M_i . With these values, each M_i would be able to calculate the lower bounds $LB_1(q_t, x)$ and upper bounds $UB_1(q_t, x)$ for each $x \in X_i$. Then, after P getting the k_{th} smallest upper bounds as thr, we could run the pruning procedure.

In each following round (i.e. $j > 1$), P sends the next s_j dimensions of q_t to each M_i whose instances were not pruned completely. These M_i will update their bounds as follows:

$$LB_j(q_t, x) = LB_{j-1}(q_t, x) + \sum_{d=p_{j-1}+1}^{p_j} (q_t[d] - x[d])^2.$$

$$\begin{aligned} UB_j(q_t, x) &= LB_j(q_t, x) \\ &+ \sum_{d=p_j+1}^{D} q_t[d]^2 + \sum_{d=p_j+1}^{D} x[d]^2 \\ &+ 2 \times \sqrt{\sum_{d=p_j+1}^{D} q_t[d]^2 \times \sum_{d=p_j+1}^{D} x[d]^2}. \end{aligned}$$

where $p_j = \sum_{i=1}^{j} s_i$, LB_j and UB_j indicate the lower bounds and upper bounds at the round j repectively.

We call those p_j as pivots, which mean that each machine would observe the first p_i elements of q_t at the round j.

3.5.4 Finding the Threshold in Distributed Machines

The question now is to find the threshold thr for pruning. We know that thr is the k_th upper bounds in all bounds of these round. However, these bounds are calculated by each local machine and lied there. That means we have to find it from a distributed data. In [2], we directly send these bounds computed in each M_i back to the server P. However, it would make the transmission cost grow linearly with the number of total instances in these distributed machines and lead to expensive cost when our dataset is extremely huge. As a result, we propose a method that could make the growth of the cost independent with the number of total instances.

To be simplified, we could think this problem as follows: given many distribued numbers N_i, we want to find the k_{th} largest number among these N_i. The N_i here actually means the upper bounds at the machine M_i in our framework. Once we model this problem as this, we could solve it through modifying the work of [3].

In [3], there are also many phases to find the kNN. In the first phase, the server P would send the whole query to every machine. Then, in the following phases, it just focuses on finding the instances with the k_{th} largest distance with the query. To make it fit our problem, we can only use the phases of PRP except the first phase to find the k_{th} largest upper bound as our threshold thr. Its cost is linear to

$$m \times (\left\lfloor \frac{k}{m} \right\rfloor + 1) = m + k,$$

which is much lower than [2] when the number of total instances is very large.

3.6 Decision of the Pivots

The remaining problem is to decide how many dimensions (i.e. s_j) of $\hat{q}_t$ we have to send from the server P in each round j. If we send too few dimensions, the bounds would be too loose to prune any candidates and we will spend much unnecessary cost in finding the thresholds. On the other hand, if we send too many dimensions, although it could allow us to prune many candidates at once, it would send too many dimensions to some candidates which could be pruned by much fewer dimensions. That would also lead to the waste the transmission cost. Therefore, in this section, we propose a simple but effective method to decide how many dimensions of $\hat{q}_t$ we should send from P in each round.

3.6.1 Estimation of the Number of Residual Machines

From the discussion above, we could notice that the decision of pivots is highly dependent on the transmission cost. Therefore, if we could estimate the cost as a cost function of the pivots p_j, we could decide these pivots by optimizing this function.

However, to estimate the cost, we need to know the number of residual candidates machines before sending those dimensions of $\hat{q}_t$ in each round. But it is almost impossible to know how many machines would be left after we send part of $\hat{q}_t$ before we actually send the part of $\hat{q}_t$. Therefore, we estimate a vector called $EstResMach \in \mathbb{R}^D$ where $EstResMach[j]$ indicates our estimate of the number of residual machines *after* we send $\hat{q}_t[1:j]$ to each local machine.

To allow P to estimate this vector $EstResMach \in \mathbb{R}^D$ without causing any more transmission cost, we use the history information from $\hat{q}_1, \ldots, \hat{q}_t$. Suppose that we just finish finding the answer for $\hat{q}_t$, during the procedure, we would collect some such pairs of information $(index_j, ResMach_j)$ for some j where $index_j$ means each candidate machine would observe the first $index_j$ dimensions of $\hat{q}_t$ at the round j and $ResMach_j$ indicates the number of residual machines after pruning by $\hat{q}_t[1 : index_j]$. For those dimensions which are not in these pairs, we use linear interpolation to estimate their $ResMach$.

Figure 3.4 is an example for the above procedure. The blue vector means the pairs of information we collected for $\hat{q}_t$ as $(2, 400)$, $(4, 390), \ldots,$ $(D-3, 15)$, $(D-1, 13)$, $(D, 1)$.

And the red vector is the results after linear interpolation.

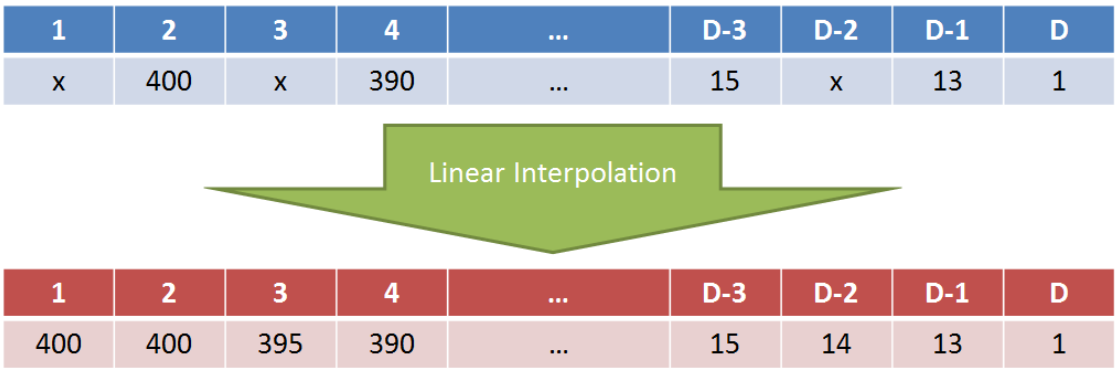

Figure 3.4: The example of linear interpolation.

By average this vector (the red one) collected from every query as $EstResMach$, P could estimate the number of residual machines *before* sending a query.

3.6.2 Estimation of the Transmission Cost

Once we have the vector $EstResMach$, we are ready to estimate the transmission cost *before* sending the query. For a query, we could estimate its transmission cost in the first round

$$Cost_1 = m \times s_1 + Cost_{PRP}(m)$$

And for those round $j > 1$ as follows,

$$Cost_j = EstResMach[p_{j-1}] \times s_j + Cost_{PRP}(EstResMach[p_{j-1}])$$

where $p_j = \sum_{i=1}^{j} s_i$.

The term $m \times s_1$ means the cost to send part of $\hat{q}_t$ in the first round which is the length of this part times the total number of machines. Similarily, the term $EstResMach[p_{j-1}] \times s_j$ is the cost to send the next part of $\hat{q}_t$ in the round j to the residual machines. On the other hand, the term $Cost_{PRP}(i)$ is the total cost to find the threshold thr and then send it to all i residual machines. Therefore, we could minimize the total cost to get the optimal pivots.

However, since there are D dimensions could be picked .there are too many variables to decide in this problem. According to our experiments, the most crucial variable is the

number of dimensions which will be sent in the first round, which is s_1 in this optimization problem. The other s_j don't have such huge influence like s_1. Therefore, we make all s_j be equal and then simplifiy this optimization problem as follows,

$$\begin{aligned} &\underset{StartD, EachLenD}{\text{minimize}} \sum_j Cost_j(StartD, EachLenD) \\ &\text{subject to } StartD, EachLenD \in \mathbb{N} \end{aligned} \tag{3.10}$$

where

$$Cost_1 = m \times StartD + Cost_{PRP}(m)$$

$$Cost_j = EstResMach[p_{j-1}] \times EachLenD + Cost_{PRP}(EstResMach[p_{j-1}]), \forall j > 1$$

$$p_j = StartD + (j-1) \times EachLenD.$$

3.6.3 Coordinate Descent to Decide the Pivots

Now we have reduced the number of variables to only two variables: $StartD$ and $EachLenD$. To solve this optimization problem efficiently, we apply the Coordinate Descent method for each query as follows.

algorthm?

After solving the optimal $StartD$ and $EachLenD$ for this query, we are able to decide its pivots as below:

$$p_j = StartD + (j-1) \times EachLenD \tag{3.11}$$

Since the vector $EstResMach$ would be updated for every new query, we will use Coordinate Descent to solve these pivots also for every new query.

3.7 Overall Framework

3.7.1 Importance-Selecting Function

Before giving the final version of our framework, we define a selection function S_i for convenience.

At each round j for q_t, we would decide what values to send from P to each M_i and then calculate the bounds at M_i. Actually, we could use a function to indicate these values. That is, our values sent from P to M_i at the round j for q_t are the return values of the importance-selecting function $S_i(q_t,\ k,\ \theta_t;\ W_i)$, which could be formulated as below.

$$S_i(q_t,\ j,\ \theta_t;\ W_i) =\ \hat{q}_t[p_{j-1}^t + 1 : p_j^t] \cup Meta_j$$

where

$$\begin{aligned}
&\hat{q}_t = W_i q_t, \\
&\theta_t = (StartD_t, LenD_t) \\
&p_j^t = min\{D, StartD_t + LenD_t \times (j-1)\},\ \forall j \geq 1,\ \forall t \\
&p_0^t = 0,\ \forall t \\
&Meta_1 = \sum_{d=StartD_t+1}^{D} \hat{q}[d]^2, Meta_j = \varnothing,\ \forall j > 1
\end{aligned} \tag{3.12}$$

Although we sent the dimensions of transformed query $\hat{q}_t$ instead of the original query q_t to each local machine, the cost of sending the values of the importance-selecting function S_t is the length of the original query q_t plus one which is the norm of the latter part of $\hat{q}_t$ in the first round. That is

$$\sum_j S_i(q_t,\ j,\ \theta_t;\ W_i) = D + 1$$

Note that this is the worst case, in our experiments, we could prune most candidates and thus don't need to send it until the last rounds.

3.7.2 Overall Framework

Finally, we give our algorithm as the following two tables.

Input: $X_1, X_2, X_3, \ldots, X_m$
Output: $W_1, W_2, W_3, \ldots, W_m$
for $i = 1; i \leq m; i = i + 1$ **do**
    M_i: Compute W_i with X_i by solving the optimization problem (3.2);
    M_i: Send W_i back to P;
end

Algorithm 1: First Phase

Input: q_t, k
Output: kNN of q_t
$P.CandidateMach(m) = True$;
$P.Counter = N$;
P: Solve θ_t by the section 3.6.3;
for $j = 1; P.Counter > k; j = j + 1$ **do**
    for $i = 1; i \leq m; i = i + 1$ **do**
        if $(CandidateMach(i) == True)$ **then**
            P: Send the values of $S_i(q_t,\ j,\ \theta_t;\ W_i)$ to M_i.;
            M_i: $\forall x \in X_i$, compute bounds by the section 3.5.3;
        end
    end
    P: Use PRP to find the threshold thr and send it to every candidate machine.;
    $P.Counter = 0$.;
    for $i = 1; i \leq m; i = i + 1$ **do**
        if $(CandidateMach(i) == True)$ **then**
            M_i: Prune instances by thr, return number of residual instances n_i back to P.;
            if $(n_i == 0)$ **then**
                $P.CandidateMach(i) = False$;
            end
            $P.Counter+ = n_i$;
        end
    end
end

Algorithm 2: Second Phase

Chapter 4

Experiment

4.1 Experiment Setup

In this section, we discuss the resluts of our experiments. There are four parts in our experiments. First, we compare our framework with other frameworks in the communication cost. Second, since there are several stages of improvement in our framework, we discuss each of their influence to our final model. Third, we consider the amortization of transmitting the orthogonal matrices. Finally, we compare the power of pruning among different bounds. For every experiment, we collect the results of 100 experiments by randomly picking our 100 instances as the queries.

4.2 Data Description

The table 4.2 is the description of those datasets we used in our experiments. Note that the $n \times m$ in the final column means that there are n instances placed in each machine and m machines used in this experiments. For instance, for the image dataset ANN with SIFT feature, there are totally 5000 machines and each has 200 instances in our experiments.

4.2.1 Time Series Data

The time series datasets we used is a synthetic dataset. We use the random walk data model in [12]. Each time series is generated by a random walk whose every step size is

Table 4.1: Summary for each dataset

Type	Dataset	Feature	Num of Dimensions	Num of Instances
Time Series	Random Walk	$N(0,1)$	128	200×5000
Image	ANN	SIFT	128	200×5000
	Flickr	CSD	256	500×2000
		SCD	256	500×2000
Audio	Million Songs	MVD	480	500×1900
		TRH	480	500×1900

a normal distributed random number with mean 0 and standard deviation 1. We also use this model to generate the synthetic dataset in the experiments of MsWave [2].

4.2.2 Image Data

We use two datasets in our experiments for images. First is the data provied in [13], which is a widely used dataset for evaluate the performance of approximate nearest neighbors search algorithms. The another one is the Flickr datasets with two kind of features used in [14]. The dataset is also a widely used dataset in the task of image retrieval. The CSD indicates *Color Structure Descriptor* while the SCD means *Scalable Color Descriptor*.

4.2.3 Audio Data

Here we use the audio data named Million Song Dataset from [15] which is a free-available collection of audio features for a million contemporary popular music tracks. For the features, MVD means *Modulation Frequency Variance Descriptor* and TRH is *Temporal Rhythm Histograms*. Please refer to [16–18] to see the details about how these features were extracted.

4.3 Comparison Among all Frameworks

$k = 10$, $|Q| = 500$, $M = 500, 1000, 1500, ...$ compare all diff frameworks for all data here. remember to add Mat cost.

4.3.1 Frameworks for Comparison

We compare our framework with those methods mentioned in the related work chapter. From [3], we use CP and PRP but with slightly modifications. In the origin CP, every machine would return the top k instances once receiving the query. But there is a trivial improvement that every machine only return the *distances* of these top k instances. Then, the server could know the distances of the kNN of this query and then ask those machines with answers to return those instances. Although it is a slight modification, it could reduce the cost of CP a lot when the number of machines is large. Also, we run LeeWave [1] in these experiments for comparision. We call our final framework as M̈ainïn the following figures.

Note that in the following experiments, the cost of our framework here does not include the cost of sending the orthogonal matrices. We will prove in the section 4.4 that the total cost including the matrices could be amortized by enough queries and thus to achieve the cost here. That is, we could see the cost of our framework here as the cost after amortized by enough queries. Due to the time limitations, we didn't conduct enough number of queries to achieve the amortized results for each dataset.

4.3.2 Results of Different Frameworks

The following figures are the results of our experiments. The x axis indicates the number of local machines while the y axis is the total transmission cost of the 100 queries. Since the differences among these are too large, we transform the y axis to logarithmic scale.

From these figures, we could see that our framework used the least transmission cost among all these frameworks for every dataset. And these differences between our framework and each other framework increased as the number of local machines increased. The reason is that when the number of local machines increases, there would be a higher chance to prune more local machines in the early round since the ratio of pruned machines doesn't change too much. But the other frameworks would be more sensitive to the number of local machines.

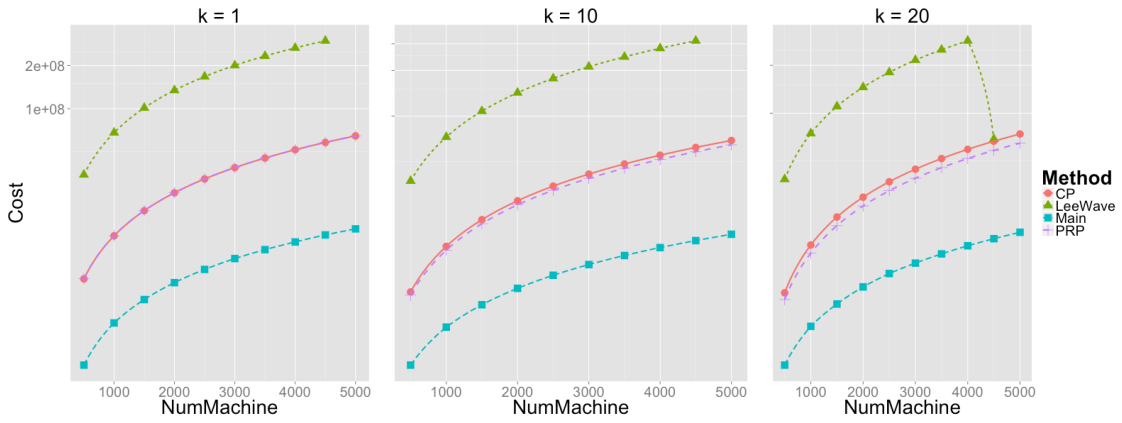

Figure 4.1: Time Series

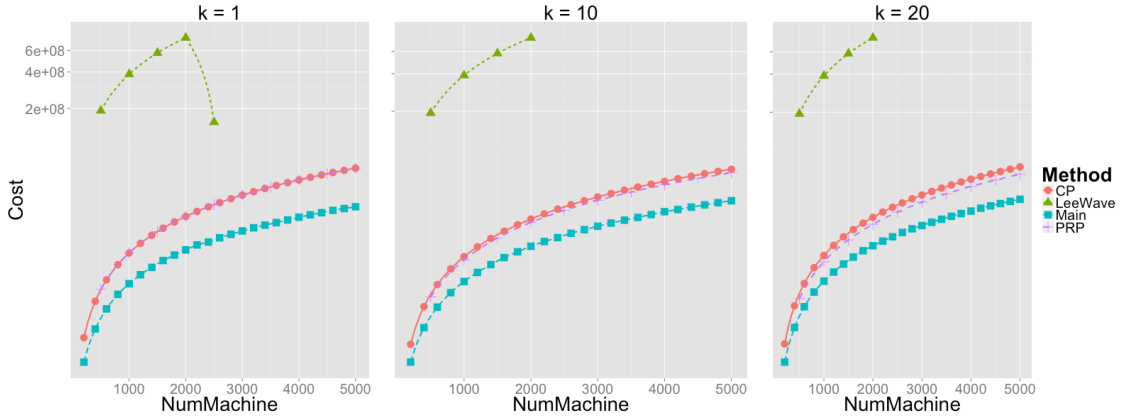

Figure 4.2: ANN

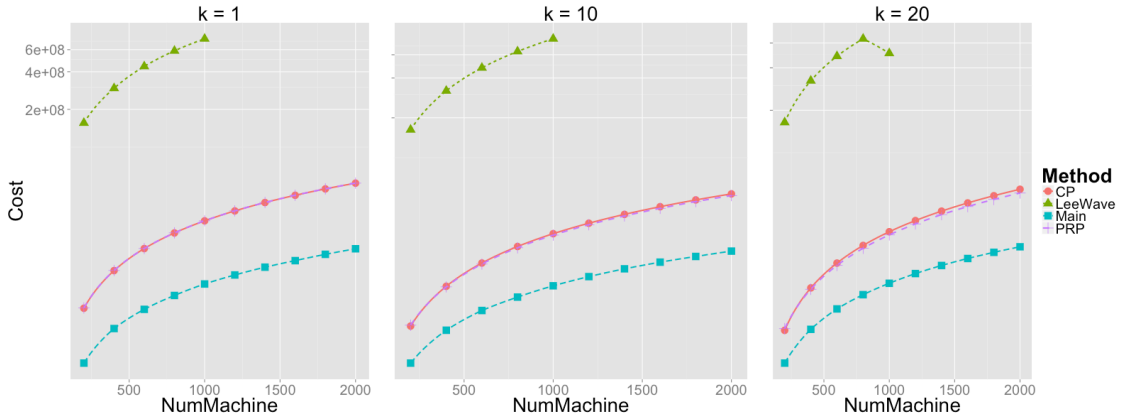

Figure 4.3: Flickr: CSD

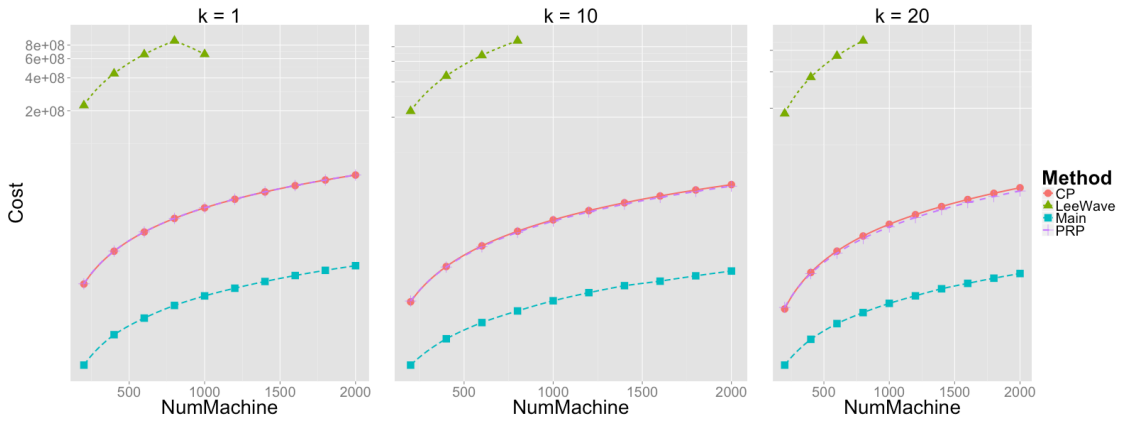

Figure 4.4: Flickr: SCD

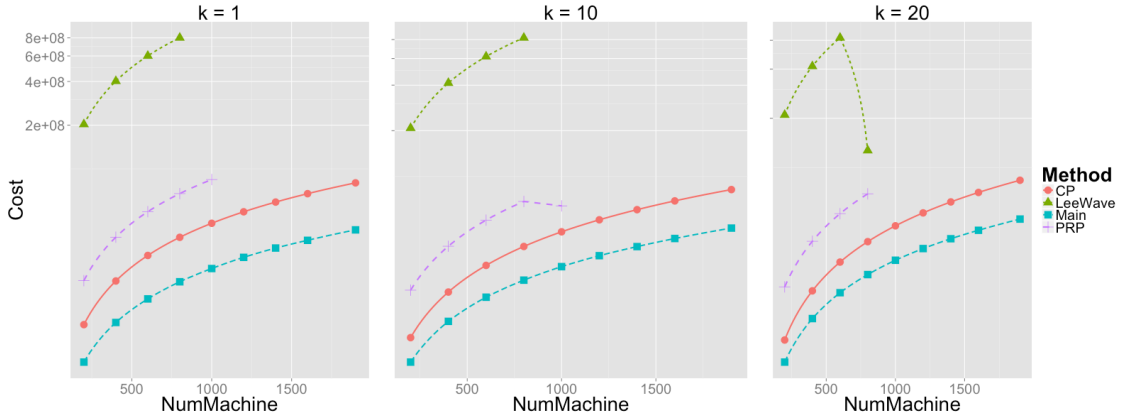

Figure 4.5: Million Song: MVD

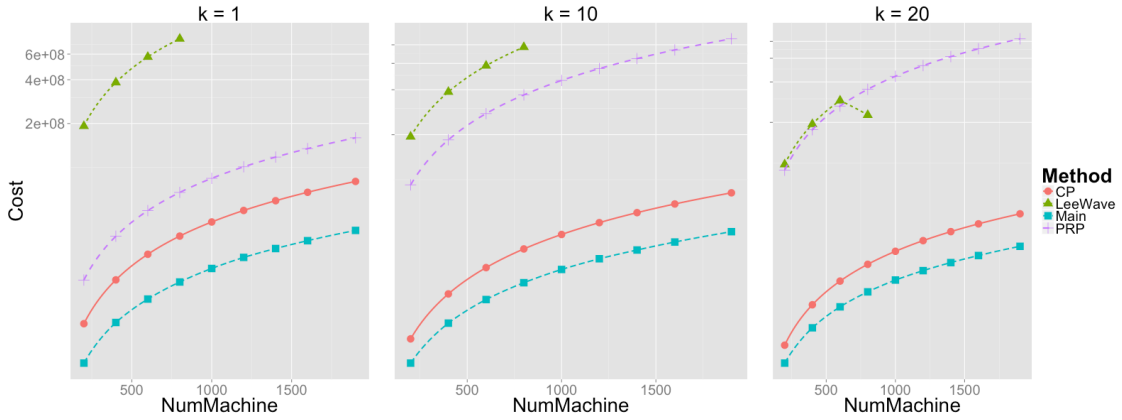

Figure 4.6: Million Song: TRH

PRP and CP.

We could also notice that LeeWave needs the largest transmission cost to finding the kNN for the 100 queries for all datasets. When the type of the dataset is time series (figure 4.1), the different between LeeWave and other frameworks are smaller than other datasets since its pruning power is still effective for this type of dataset. For other types of datasets like images or audio, LeeWave almost could not prune any candidiate machines until the last round which would sent the whole query to each machine and thus used a large amount of transmission cost.

Even though the LeeWave could prune some candidates machines when the type of the dataset is time series, there is a big gap between it and CP, PRP. The reason of the large transmission cost in LeeWave here is not the pruning power but its way to calculate the bounds. In each round, after sending the coefficients in this level of the error tree, LeeWave requires every instances to return some metadata back for calculating the bounds at the server. This causes one term in the transmission cost of LeeWave would grows linearly with the total number of instances in all local machines. Since we conducted our experiments on the datasets with about one million instances, this term would be large enough to cover the saving from the pruning. On the other hand, the transmission cost of CP and PRP is independent of the total number of instance but only dependent on the number of local machines and k. Therefore, when the total instances is large, LeeWave could use more transmission cost than CP and PRP even when its pruning power still exists.

Table 4.2: Summary for each dataset

Type	Dataset	Feature	Total Num of Instances	Num of Queries
Time Series	Random Walk	$N(0,1)$	1000000	123
Image	ANN	SIFT	1000000	456
	Flickr	CSD	1000000	789
		SCD	1000000	1
Audio	Million Songs	MVD	950000	2
		TRH	950000	5

4.4 Number of Queries for Amortizing the Cost of Matrices

4.5 Comparison Among our Framework with Different Configurations

There are many stages of algorithms which lead to the final version of our framework. Therefore, in this section, we would like to discuss the performance of our framwork with or without.

From MsWave [2] to our framework, we have enhanced it

4.6 Power of the Pruning Procedure

Comp among LeeWave, NoW, Main for ResSite.

Bibliography

[1] Mi-Yen Yeh, Kun-Lung Wu, Philip S. Yu, and Ming-Syan Chen. LeeWave: level-wise distribution of wavelet coefficients for processing $k$nn queries over distributed streams. *Proc. VLDB Endow.*, 2008.

[2] Jui-Pin Wang, Yu-Chen Lu, Mi-Yen Yeh, Shou-De Lin, and Phillip B. Gibbons. Communication-efficient distributed multiple reference pattern matching for m2m systems. *Proc. of IEEE ICDM*, 2013.

[3] Apostolos N. Papadopoulos and Yannis Manolopoulos. Distributed processing of similarity queries. *Distributed and Parallel Databases*, 2001.

[4] Sylvia Ratnasamy, Paul Francis, Mark Handley, and Richard Karp Scott Shenker. A scalable content-addressable network. *SIGCOMM*, 2001.

[5] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. *SIGCOMM*, 2001.

[6] Adina Crainiceanu, Prakash Linga, Ashwin Machanavajjhala, Johannes Gehrke, and Jayavel Shanmugasundaram. P-ring: an efficient and robust p2p range index structure. *SIGMOD*, 2007.

[7] Ashwin R. Bharambe, Mukesh Agrawal, and Srinivasan Seshan. Mercury: Supporting scalable multi-attribute range queries. *SIGCOMM*, 2004.

[8] Erik Buchmann and Klemens Bohm. Efficient evaluation of nearest-neighbor queries in content-addressable networks. *From Integrated Publication and Information Systems to Virtual Information and Knowledge Environments*, 2005.

[9] Farnoush Banaei-Kashani and Cyrus Shahabi. Swam: A family of access methods for similarity-search in peer-to-peer data networks. *CIKM*, 2004.

[10] Parisa Haghani, Sebastian Michel, Philippe Cudre-Mauroux, and Karl Aberer. Lsh at large – distributed knn search in high dimensions. *International Workshop on Web and Databases*, 2008.

[11] Zaiwen Wen and Wotao Yin. Optimization with orthogonality constraints. `http://optman.blogs.rice.edu/`, Aril 2012.

[12] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In *Proc. of ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, 2012.

[13] Hervé Jégou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2011.

[14] Nitish Srivastava and Ruslan R. Salakhutdinov. Multimodal learning with deep boltzmann machines. In *Neural Information Processing Systems*, 2012.

[15] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[16] Thomas Lidy and Andreas Rauber. Evaluation of feature extractors and psychoacoustic transformations for music genre classification. In *Proceedings of the Sixth International Conference on Music Information Retrieval (ISMIR 2005)*, pages 34–41, London, UK, September 11-15 2005.

[17] A. Rauber, E. Pampalk, and D. Merkl. The SOM-enhanced JukeBox: Organization and visualization of music collections based on perceptual models. *Journal of New Music Research*, 2003.

[18] A. Rauber and M. Frühwirth. Automatically analyzing and organizing music archives. In *Proceedings of the 5. European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2001)*, Springer Lecture Notes in Computer Science, Darmstadt, Germany, Sept. 4-8 2001. Springer. http://www.ifs.tuwien.ac.at/ifs/research/publications.html.