How to Read and Summarize Existing Research Studies?

Tsz Nam Chan

edisonchan@szu.edu.cn

College of Computer Science & Software Engineering

Why Do We Need to Summarize Existing Research Studies?

- You are not familiar with that topic.
  - You are interested in conducting research for it.
  - You want to understand the state-of-the-art solutions.
- You want to write a survey paper for that topic.
- You want to conduct a tutorial (write a tutorial paper) for that topic.
- You need to write the “Related Work” section.
- Your PhD supervisor asks you to do so.

Steps for Summarizing Research Papers

1. Read papers.

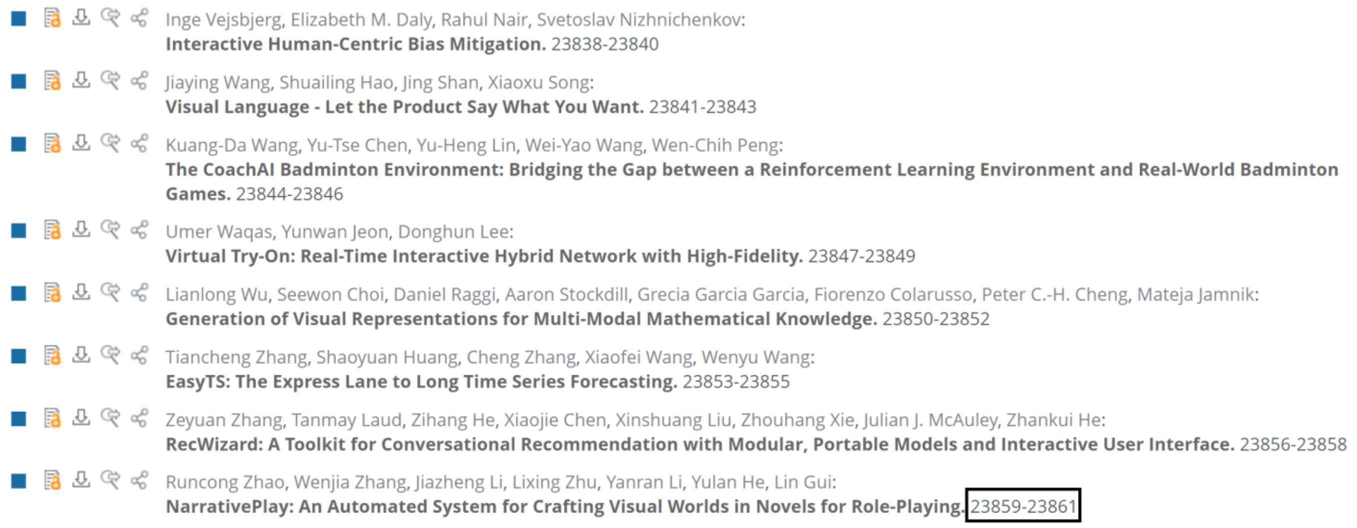
2. Categorize those papers into different groups.

3. Summarize different groups of research papers.
    1. Core ideas of different groups.
    2. The main differences (e.g., advantages and disadvantages) between different groups of papers.

How to Read Papers?

- Looks easy. (But it is very difficult.)

- Not necessary (and not feasible) to fully read every paper.



Inge Vejsbjerg, Elizabeth M. Daly, Rahul Nair, Svetoslav Nizhnichenkov:
Interactive Human-Centric Bias Mitigation. 23838-23840

Jiaying Wang, Shuailing Hao, Jing Shan, Xiaoxu Song:
Visual Language - Let the Product Say What You Want. 23841-23843

Kuang-Da Wang, Yu-Tse Chen, Yu-Heng Lin, Wei-Yao Wang, Wen-Chih Peng:
The CoachAI Badminton Environment: Bridging the Gap between a Reinforcement Learning Environment and Real-World Badminton Games. 23844-23846

Umer Waqas, Yunwan Jeon, Donghun Lee:
Virtual Try-On: Real-Time Interactive Hybrid Network with High-Fidelity. 23847-23849

Lianlong Wu, Seewon Choi, Daniel Raggi, Aaron Stockdill, Grecia Garcia Garcia, Fiorenzo Colarusso, Peter C.-H. Cheng, Mateja Jamnik:
Generation of Visual Representations for Multi-Modal Mathematical Knowledge. 23850-23852

Tiancheng Zhang, Shaoyuan Huang, Cheng Zhang, Xiaofei Wang, Wenyu Wang:
EasyTS: The Express Lane to Long Time Series Forecasting. 23853-23855

Zeyuan Zhang, Tanmay Laud, Zihang He, Xiaojie Chen, Xinshuang Liu, Zhouhang Xie, Julian J. McAuley, Zhankui He:
RecWizard: A Toolkit for Conversational Recommendation with Modular, Portable Models and Interactive User Interface. 23856-23858

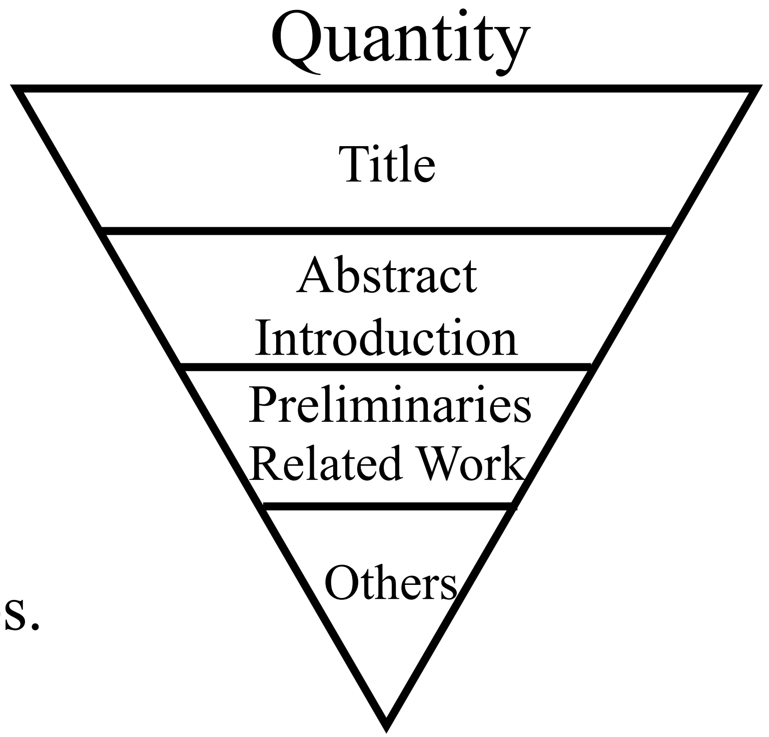
Runcong Zhao, Wenjia Zhang, Jiazheng Li, Lixing Zhu, Yanran Li, Yulan He, Lin Gui:
NarrativePlay: An Automated System for Crafting Visual Worlds in Novels for Role-Playing. 23859-23861

In AAAI 2024

- Fully reading a paper does not indicate that you understand that paper.

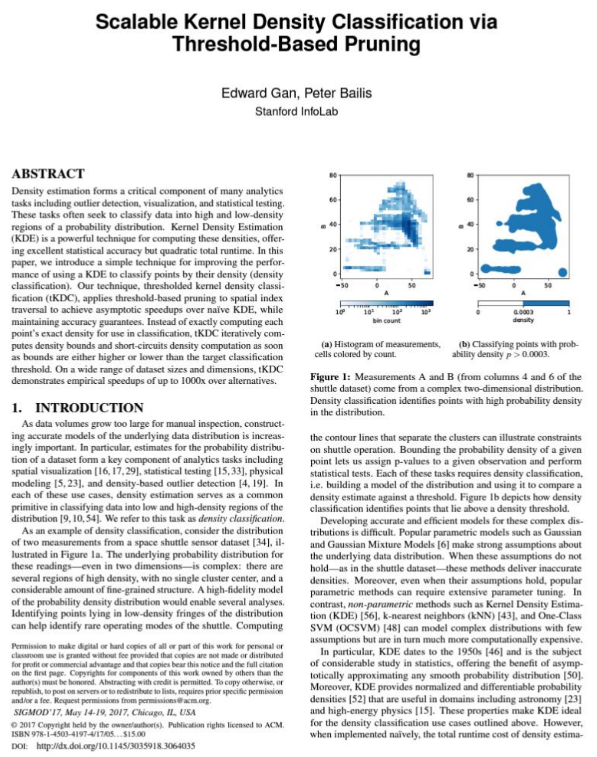
How to Read Papers?

- Read title.
  - You can filter a large portion of papers based on this.
- Read Abstract and Introduction.
  - You can categorize those papers into different groups.
  - You can identify important papers.
- Read Preliminaries and Related Work.
  - You can (further) categorize those papers into different groups.
  - You can identify some missing papers.
- Only need to fully read a few (important) papers.



How to Read Papers?

- How many papers did I fully read for writing the paper “KARL: Fast Kernel Aggregation Queries. ICDE 2019”? Answer: 3



Scalable Kernel Density Classification via Threshold-Based Pruning

Edward Gan, Peter Bailis
Stanford InfoLab

ABSTRACT

Density estimation forms a critical component of many analytics tasks including outlier detection, visualization, and statistical testing. These tasks often seek to classify data into high and low-density regions of a probability distribution. Kernel Density Estimation (KDE) is a powerful technique for computing these densities, offering excellent statistical accuracy but quadratic total runtime. In this paper, we introduce a simple technique for improving the performance of using a KDE to classify points by their density (density classification). Our technique, thresholded kernel density classification (tKDC), applies threshold-based pruning to spatial index traversal to achieve asymptotic speedups over naïve KDE, while maintaining accuracy guarantees. Instead of exactly computing each point’s exact density for use in classification, tKDC iteratively computes density bounds and short-circuits density computation as soon as bounds are either higher or lower than the target classification threshold. On a wide range of dataset sizes and dimensions, tKDC demonstrates empirical speedups of up to 1000x over alternatives.

(a) Histogram of measurements, cells colored by count. (b) Classifying points with probability density $p > 0.0003$.

Figure 1: Measurements A and B (from columns 4 and 6 of the shuttle dataset) come from a complex two-dimensional distribution. Density classification identifies points with high probability density in the distribution.

1. INTRODUCTION

As data volumes grow too large for manual inspection, constructing accurate models of the underlying data distribution is increasingly important. In particular, estimates for the probability distribution of a dataset form a key component of analytics tasks including spatial visualization [16, 17, 29], statistical testing [15, 33], physical modeling [5, 23], and density-based outlier detection [4, 19]. In each of these use cases, density estimation serves as a common primitive in classifying data into low and high-density regions of the distribution [9, 10, 54]. We refer to this task as *density classification*.

As an example of density classification, consider the distribution of two measurements from a space shuttle sensor dataset [34], illustrated in Figure 1a. The underlying probability distribution for these readings—even in two dimensions—is complex: there are several regions of high density, with no single cluster center, and a considerable amount of fine-grained structure. A high-fidelity model of the probability density distribution would enable several analyses. Identifying points lying in low-density fringes of the distribution can help identify rare operating modes of the shuttle. Computing the contour lines that separate the clusters can illustrate constraints on shuttle operation. Bounding the probability density of a given point lets us assign p-values to a given observation and perform statistical tests. Each of these tasks requires density classification, i.e. building a model of the distribution and using it to compare a density estimate against a threshold. Figure 1b depicts how density classification identifies points that lie above a density threshold.

Developing accurate and efficient models for these complex distributions is difficult. Popular parametric models such as Gaussian and Gaussian Mixture Models [6] make strong assumptions about the underlying data distribution. When these assumptions do not hold—as in the shuttle dataset—these methods deliver inaccurate densities. Moreover, even when their assumptions hold, popular parametric methods can require extensive parameter tuning. In contrast, *non-parametric* methods such as Kernel Density Estimation (KDE) [56], k-nearest neighbors (kNN) [43], and One-Class SVM (OCSVM) [48] can model complex distributions with few assumptions but are in turn much more computationally expensive.

In particular, KDE dates to the 1950s [46] and is the subject of considerable study in statistics, offering the benefit of asymptotically approximating any smooth probability distribution [50]. Moreover, KDE provides normalized and differentiable probability densities [52] that are useful in domains including astronomy [23] and high-energy physics [15]. These properties make KDE ideal for the density classification use cases outlined above. However, when implemented naïvely, the total runtime cost of density estima-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

SIGMOD’17, May 14-19, 2017, Chicago, IL, USA

© 2017 Copyright held by the owner/author(s). Publication rights licensed to ACM. ISBN 978-1-4503-4197-4/17/05…$15.00

DOI: http://dx.doi.org/10.1145/3035918.3064035

SIGMOD 2017

Nonparametric Density Estimation:
Toward Computational Tractability

Alexander G. Gray
Department of Computer Science
Carnegie Mellon University
agray@cs.cmu.edu

Andrew W. Moore
Robotics Inst. and Dept. Comp. Sci.
Carnegie Mellon University
awm@cs.cmu.edu

January 24, 2003

Abstract

Density estimation is a core operation of virtually all probabilistic learning methods (as opposed to discriminative methods). Approaches to density estimation can be divided into two principal classes, parametric methods, such as Bayesian networks, and nonparametric methods such as kernel density estimation and smoothing splines. While neither choice should be universally preferred for all situations, a well-known benefit of nonparametric methods is their ability to achieve estimation optimality for ANY input distribution as more data are observed, a property that no model with a parametric assumption can have, and one of great importance in exploratory data analysis and mining where the underlying distribution is decidedly unknown. To date, however, despite a wealth of advanced underlying statistical theory, the use of nonparametric methods has been limited by their computational intractibility for all but the smallest datasets. In this paper, we present an algorithm for kernel density estimation, the chief nonparametric approach, which is dramatically faster than previous algorithmic approaches in terms of both dataset size and dimensionality. Furthermore, the algorithm provides arbitrarily tight accuracy guarantees, provides anytime convergence, works for all common kernel choices, and requires no parameter tuning. The algorithm is an instance of a new principle of algorithm design: multi-recursion, or higher-order divide-and-conquer.

Keywords: kernel density estimation, nonparametric statistics, algorithms, divide-and-conquer, space-partitioning trees.

1 Introduction: Data Analysis Without Assumptions

In this section we’ll briefly review the fundamental problem of density estimation and the reasons that nonparametric density estimation approaches are particularly well-suited to exploratory data mining. We’ll then describe the severe computational obstacles posed by nonparametric density estimation, the main factor limiting their use in large-scale data analysis.

1.1 The fundamental problem of density estimation. In any probabilistic learning method (as opposed to discriminative methods such as support vector machines) the task of estimating a probability density from data is a fundamental one, upon which subsequent inference and decision-making procedures are based. For example, in classification one must find $\hat{P}(C|\underline{X}) = \frac{\hat{p}(\underline{X}|C)P(C)}{\hat{p}(\underline{X})}$, where C is one of K classes and $\hat{p}(\underline{X}|C)$ is the (class-conditional) density of the data $\underline{X}$. Direct density estimation provides a principled way to formulate many common types of analyses, for example outlier detection (as low-density points), or more generally scoring of points according to how ’common’ they are. In general, density estimation provides a classical basis across statistics for virtually any kind of data analysis in principle, including clustering, classification, regression, time series analysis, active learning, and so on [7, 1].

1.2 Methods of estimating a density. The task of estimating a probability density from data is a fundamental one, upon which subsequent inference, learning, and decision-making procedures are based. Density estimation has thus been heavily studied, under three primary umbrellas: parametric, semi-parametric, and nonparametric. *Parametric* methods are useful when the underlying distribution is known in advance or is simple enough to well-modeled by a standard distribution. *Semi-parametric* models (such as mixtures of simpler distributions) are more flexible and more forgiving of the user’s lack of the true model, but usually require significant computation in order to fit the resulting nonlinear models (such as the EM iterative re-estimation method). *Nonparametric* methods assume the least structure of the three, and take the strongest stance of letting the data speak for themselves [22]. They are useful in the setting of arbitrary-shape distributions coming from complex real-world data sources. They are generally the method of choice in exploratory data analysis for this reason, and can be used, as the other types of models, for the entire range of statistical settings, from supervised learning to unsupervised learning to reinforcement learning. However, they apparently often come at the heaviest computational cost of the three types

Copyright © by SIAM. Unauthorized reproduction of this article is prohibited 203

SDM 2003

Optimal Multi-Step *k*-Nearest Neighbor Search

Thomas Seidl
University of Munich, Germany
Institute for Computer Science
http://www.dbs.informatik.uni-muenchen.de
seidl@dbs.informatik.uni-muenchen.de

Hans-Peter Kriegel
University of Munich, Germany
Institute for Computer Science
http://www.dbs.informatik.uni-muenchen.de
kriegel@dbs.informatik.uni-muenchen.de

Abstract

For an increasing number of modern database applications, efficient support of similarity search becomes an important task. Along with the complexity of the objects such as images, molecules and mechanical parts, also the complexity of the similarity models increases more and more. Whereas algorithms that are directly based on indexes work well for simple medium-dimensional similarity distance functions, they do not meet the efficiency requirements of complex high-dimensional and adaptable distance functions. The use of a multi-step query processing strategy is recommended in these cases, and our investigations substantiate that the number of candidates which are produced in the filter step and exactly evaluated in the refinement step is a fundamental efficiency parameter. After revealing the strong performance shortcomings of the state-of-the-art algorithm for *k*-nearest neighbor search [Korn et al. 1996], we present a novel multi-step algorithm which is guaranteed to produce the minimum number of candidates. Experimental evaluations demonstrate the significant performance gain over the previous solution, and we observed average improvement factors of up to 120 for the number of candidates and up to 48 for the total runtime.

1 INTRODUCTION

More and more applications of database systems require the efficient support of similarity search. Examples include molecular biology [BMH 92], medical imaging [Kor+ 96], CAD/CAM systems [BK 97], and multimedia databases [Fal+ 94] [Haf+ 95] [SK 97] among many others [Jag 91] [AFS 93] [GM 93] [FRM 94] [ALSS 95]. In all of these approaches, similarity is defined in terms of a more or less complex similarity distance function. The smaller the similarity distance value, the more similar are two objects. Typical query types are the similarity range query which is specified by a query object and a similarity distance range $[0, \varepsilon]$, and the *k*-nearest neighbor query which is specified by a query object and a number *k* for the *k* most similar objects to be retrieved.

Whereas single-step algorithms for similarity search already meet the requirements of very large databases, these solutions suffer from the increasing complexity of the objects and of the similarity distance functions. For classic spatial queries such as point queries and region queries, multi-step algorithms have been developed to efficiently support complex objects [OM 88] [BHKS 93]. The paradigm of multi-step query processing has already been extended to complex similarity search, and available algorithms aim at similarity range queries [AFS 93] [FRM 94] and *k*-nearest neighbor queries [Kor+ 96]. However, we observed a bad performance of the latter solution in our experiments on large image and biomolecular databases. Starting from a theoretical analysis of the situation, we develop a novel, optimal multi-step algorithm for *k*-nearest neighbor search that implies a minimum number of exact object distance evaluations.

The paper is organized as follows: In the remainder of this introduction, we specify our problem of complex similarity search. Section 2 is dedicated to algorithms for similarity search and incremental similarity ranking that directly work on index structures in a way they are employed by our new method. In section 3, we present the available multi-step algorithm for *k*-nearest neighbor search of [Kor+ 96] including the significant efficiency shortcomings of the solution. Experiments substantiate that the number of candidates is a fundamental efficiency parameter. We present our novel algorithm in section 4 along with a proof that it exactly generates the minimum number of candidates. The experimental evaluation in section 5 demonstrates the substantial performance improvement before the paper is concluded in section 6.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
SIGMOD ’98 Seattle, WA, USA
© 1998 ACM 0-89791-995-5/98/006...$5.00

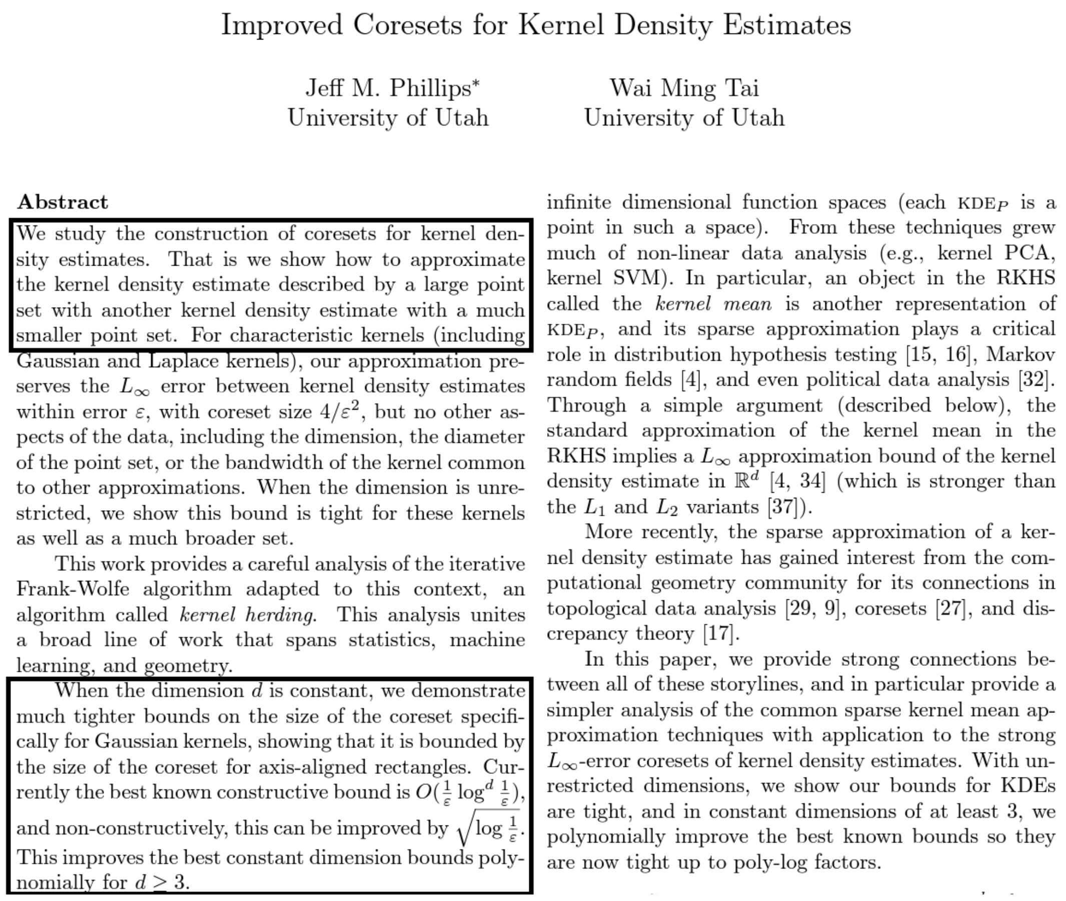
154

SIGMOD 1998

- How many papers did I (partly) read for writing this paper? Answer: Too many

How to Categorize Papers into Different Groups?

- Understanding the main technical novelty of this paper.
  - This can normally be found in the abstract and introduction sections.
- What are the main novelties of this paper?
  - Develop a new sampling method.
  - Achieve non-trivial sampling guarantees.
- Which category does this paper belong to?
  - Sampling



Improved Coresets for Kernel Density Estimates

Jeff M. Phillips* University of Utah

Wai Ming Tai University of Utah

Abstract

We study the construction of coresets for kernel density estimates. That is we show how to approximate the kernel density estimate described by a large point set with another kernel density estimate with a much smaller point set. For characteristic kernels (including Gaussian and Laplace kernels), our approximation preserves the L_∞ error between kernel density estimates within error ε, with coreset size $4/\varepsilon^2$, but no other aspects of the data, including the dimension, the diameter of the point set, or the bandwidth of the kernel common to other approximations. When the dimension is unrestricted, we show this bound is tight for these kernels as well as a much broader set.

This work provides a careful analysis of the iterative Frank-Wolfe algorithm adapted to this context, an algorithm called *kernel herding*. This analysis unites a broad line of work that spans statistics, machine learning, and geometry.

When the dimension d is constant, we demonstrate much tighter bounds on the size of the coreset specifically for Gaussian kernels, showing that it is bounded by the size of the coreset for axis-aligned rectangles. Currently the best known constructive bound is $O(\frac{1}{\varepsilon}\log^d\frac{1}{\varepsilon})$, and non-constructively, this can be improved by $\sqrt{\log\frac{1}{\varepsilon}}$. This improves the best constant dimension bounds polynomially for $d \geq 3$.

infinite dimensional function spaces (each KDE$_P$ is a point in such a space). From these techniques grew much of non-linear data analysis (e.g., kernel PCA, kernel SVM). In particular, an object in the RKHS called the *kernel mean* is another representation of KDE$_P$, and its sparse approximation plays a critical role in distribution hypothesis testing [15, 16], Markov random fields [4], and even political data analysis [32]. Through a simple argument (described below), the standard approximation of the kernel mean in the RKHS implies a L_∞ approximation bound of the kernel density estimate in $\mathbb{R}^d$ [4, 34] (which is stronger than the L_1 and L_2 variants [37]).

More recently, the sparse approximation of a kernel density estimate has gained interest from the computational geometry community for its connections in topological data analysis [29, 9], coresets [27], and discrepancy theory [17].

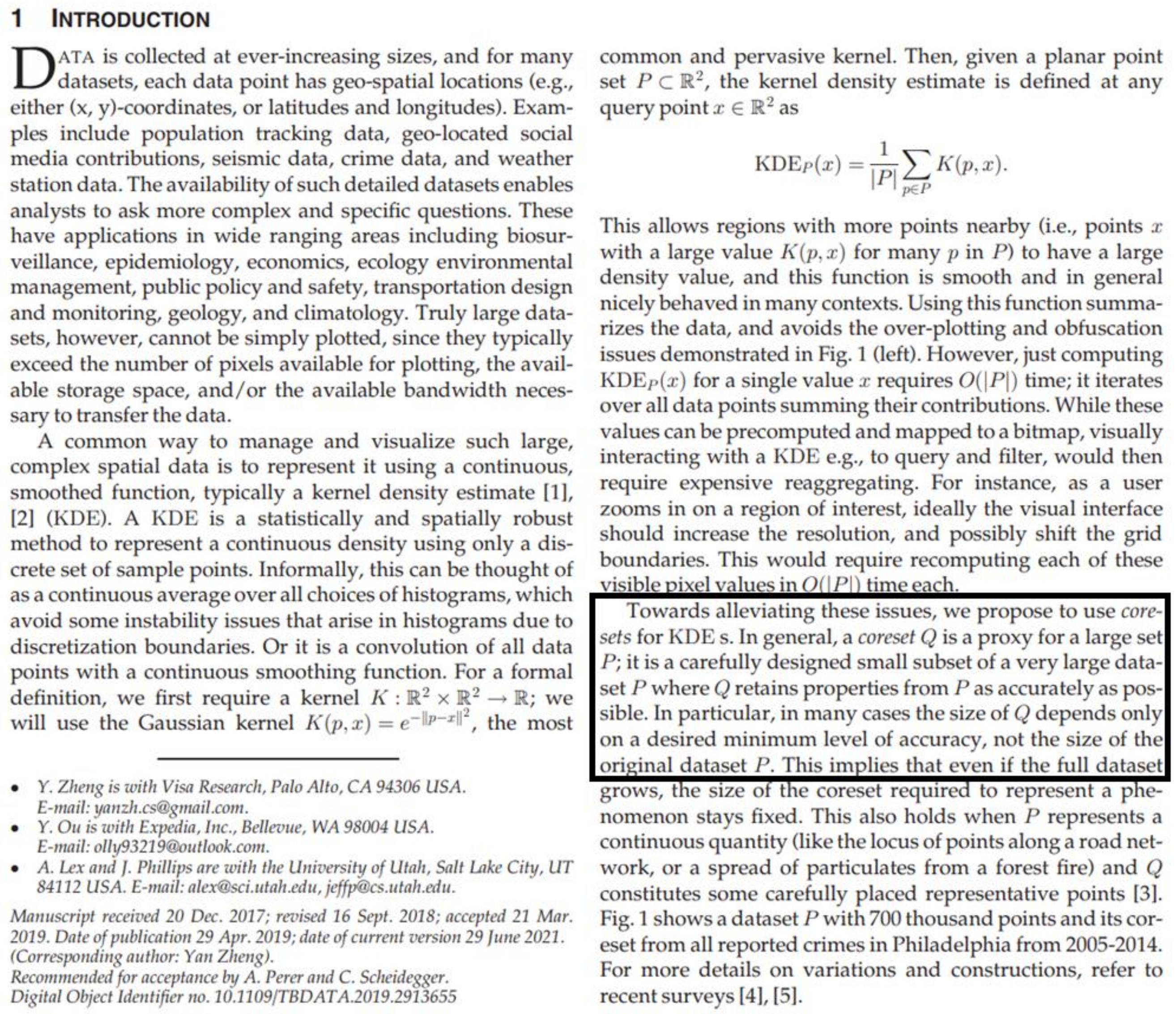
In this paper, we provide strong connections between all of these storylines, and in particular provide a simpler analysis of the common sparse kernel mean approximation techniques with application to the strong L_∞-error coresets of kernel density estimates. With unrestricted dimensions, we show our bounds for KDEs are tight, and in constant dimensions of at least 3, we polynomially improve the best known bounds so they are now tight up to poly-log factors.

SODA 2018

How to Categorize Papers into Different Groups?

- What are the main novelties of this paper?
  - Develop a new sampling method.
  - Achieve non-trivial sampling guarantees.

- Which category does this paper belong to?
  - Sampling



1 Introduction

Data is collected at ever-increasing sizes, and for many datasets, each data point has geo-spatial locations (e.g., either (x, y)-coordinates, or latitudes and longitudes). Examples include population tracking data, geo-located social media contributions, seismic data, crime data, and weather station data. The availability of such detailed datasets enables analysts to ask more complex and specific questions. These have applications in wide ranging areas including biosurveillance, epidemiology, economics, ecology environmental management, public policy and safety, transportation design and monitoring, geology, and climatology. Truly large datasets, however, cannot be simply plotted, since they typically exceed the number of pixels available for plotting, the available storage space, and/or the available bandwidth necessary to transfer the data.

A common way to manage and visualize such large, complex spatial data is to represent it using a continuous, smoothed function, typically a kernel density estimate [1], [2] (KDE). A KDE is a statistically and spatially robust method to represent a continuous density using only a discrete set of sample points. Informally, this can be thought of as a continuous average over all choices of histograms, which avoid some instability issues that arise in histograms due to discretization boundaries. Or it is a convolution of all data points with a continuous smoothing function. For a formal definition, we first require a kernel $K : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$; we will use the Gaussian kernel $K(p, x) = e^{-\|p-x\|^2}$, the most common and pervasive kernel. Then, given a planar point set $P \subset \mathbb{R}^2$, the kernel density estimate is defined at any query point $x \in \mathbb{R}^2$ as

$$\mathrm{KDE}_P(x) = \frac{1}{|P|}\sum_{p\in P} K(p, x).$$

This allows regions with more points nearby (i.e., points x with a large value $K(p, x)$ for many p in P) to have a large density value, and this function is smooth and in general nicely behaved in many contexts. Using this function summarizes the data, and avoids the over-plotting and obfuscation issues demonstrated in Fig. 1 (left). However, just computing $\mathrm{KDE}_P(x)$ for a single value x requires $O(|P|)$ time; it iterates over all data points summing their contributions. While these values can be precomputed and mapped to a bitmap, visually interacting with a KDE e.g., to query and filter, would then require expensive reaggregating. For instance, as a user zooms in on a region of interest, ideally the visual interface should increase the resolution, and possibly shift the grid boundaries. This would require recomputing each of these visible pixel values in $O(|P|)$ time each.

Towards alleviating these issues, we propose to use *coresets* for KDE s. In general, a *coreset* Q is a proxy for a large set P; it is a carefully designed small subset of a very large dataset P where Q retains properties from P as accurately as possible. In particular, in many cases the size of Q depends only on a desired minimum level of accuracy, not the size of the original dataset P. This implies that even if the full dataset grows, the size of the coreset required to represent a phenomenon stays fixed. This also holds when P represents a continuous quantity (like the locus of points along a road network, or a spread of particulates from a forest fire) and Q constitutes some carefully placed representative points [3]. Fig. 1 shows a dataset P with 700 thousand points and its coreset from all reported crimes in Philadelphia from 2005-2014. For more details on variations and constructions, refer to recent surveys [4], [5].

- *Y. Zheng is with Visa Research, Palo Alto, CA 94306 USA. E-mail: yanzh.cs@gmail.com.*
- *Y. Ou is with Expedia, Inc., Bellevue, WA 98004 USA. E-mail: olly93219@outlook.com.*
- *A. Lex and J. Phillips are with the University of Utah, Salt Lake City, UT 84112 USA. E-mail: alex@sci.utah.edu, jeffp@cs.utah.edu.*

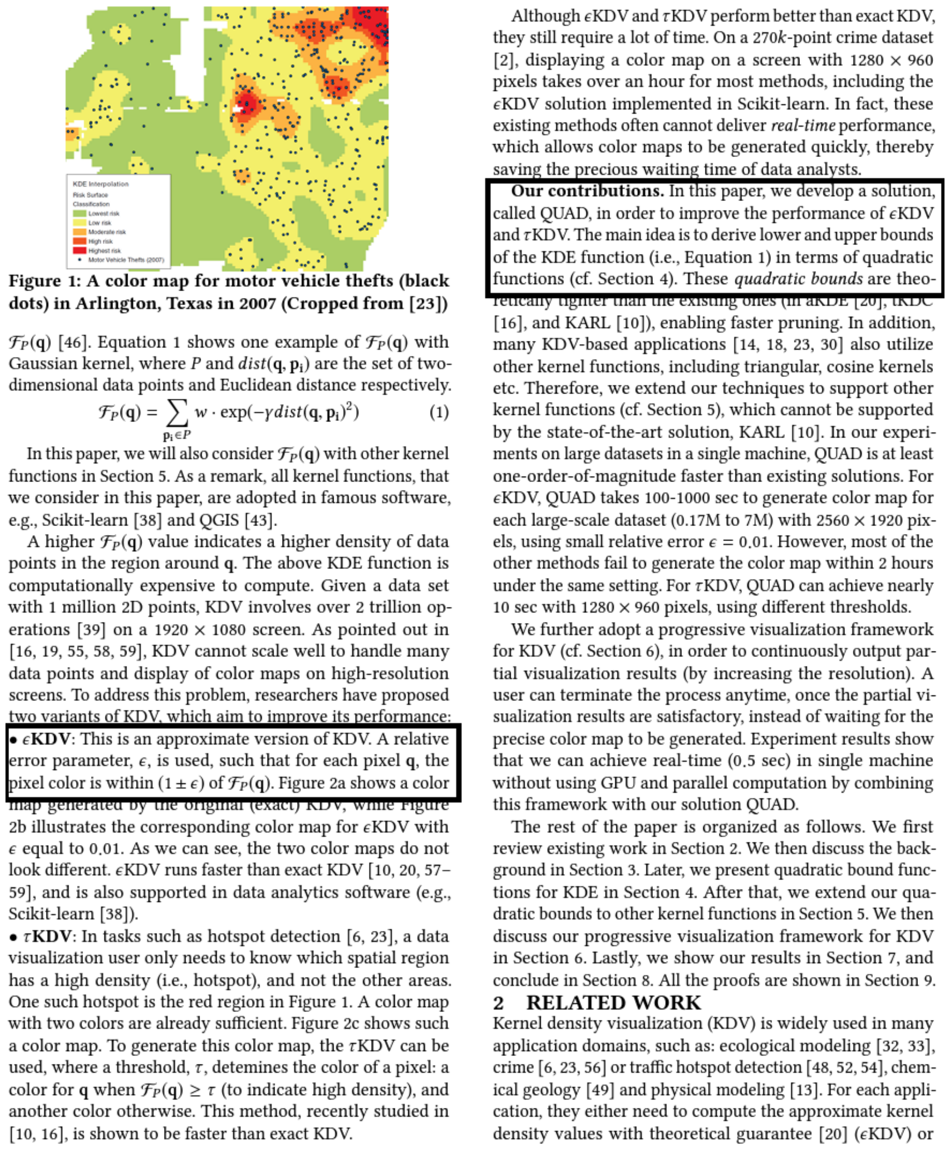
Manuscript received 20 Dec. 2017; revised 16 Sept. 2018; accepted 21 Mar. 2019. Date of publication 29 Apr. 2019; date of current version 29 June 2021. (Corresponding author: Yan Zheng).
Recommended for acceptance by A. Perer and C. Scheidegger.
Digital Object Identifier no. 10.1109/TBDATA.2019.2913655

Transactions on Big Data 2021

How to Categorize Papers into Different Groups?

- What are the main novelties of this paper?
  - Develop a new function approximation method.
  - Achieve non-trivial approximation guarantees.

- Which category does this paper belong to?
  - Function approximation



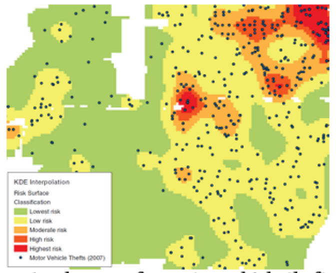


Figure 1: A color map for motor vehicle thefts (black dots) in Arlington, Texas in 2007 (Cropped from [23])

$\mathcal{F}_P(\mathbf{q})$ [46]. Equation 1 shows one example of $\mathcal{F}_P(\mathbf{q})$ with Gaussian kernel, where P and $dist(\mathbf{q}, \mathbf{p}_i)$ are the set of two-dimensional data points and Euclidean distance respectively.

$$\mathcal{F}_P(\mathbf{q}) = \sum_{\mathbf{p}_i \in P} w \cdot \exp(-\gamma dist(\mathbf{q}, \mathbf{p}_i)^2) \quad (1)$$

In this paper, we will also consider $\mathcal{F}_P(\mathbf{q})$ with other kernel functions in Section 5. As a remark, all kernel functions, that we consider in this paper, are adopted in famous software, e.g., Scikit-learn [38] and QGIS [43].

A higher $\mathcal{F}_P(\mathbf{q})$ value indicates a higher density of data points in the region around $\mathbf{q}$. The above KDE function is computationally expensive to compute. Given a data set with 1 million 2D points, KDV involves over 2 trillion operations [39] on a 1920×1080 screen. As pointed out in [16, 19, 55, 58, 59], KDV cannot scale well to handle many data points and display of color maps on high-resolution screens. To address this problem, researchers have proposed two variants of KDV, which aim to improve its performance:

- ϵ**KDV**: This is an approximate version of KDV. A relative error parameter, ϵ, is used, such that for each pixel $\mathbf{q}$, the pixel color is within $(1 \pm \epsilon)$ of $\mathcal{F}_P(\mathbf{q})$. Figure 2a shows a color map generated by the original (exact) KDV, while Figure 2b illustrates the corresponding color map for ϵKDV with ϵ equal to 0.01. As we can see, the two color maps do not look different. ϵKDV runs faster than exact KDV [10, 20, 57–59], and is also supported in data analytics software (e.g., Scikit-learn [38]).
- τ**KDV**: In tasks such as hotspot detection [6, 23], a data visualization user only needs to know which spatial region has a high density (i.e., hotspot), and not the other areas. One such hotspot is the red region in Figure 1. A color map with two colors are already sufficient. Figure 2c shows such a color map. To generate this color map, the τKDV can be used, where a threshold, τ, detemines the color of a pixel: a color for $\mathbf{q}$ when $\mathcal{F}_P(\mathbf{q}) \geq \tau$ (to indicate high density), and another color otherwise. This method, recently studied in [10, 16], is shown to be faster than exact KDV.

Although ϵKDV and τKDV perform better than exact KDV, they still require a lot of time. On a $270k$-point crime dataset [2], displaying a color map on a screen with 1280×960 pixels takes over an hour for most methods, including the ϵKDV solution implemented in Scikit-learn. In fact, these existing methods often cannot deliver *real-time* performance, which allows color maps to be generated quickly, thereby saving the precious waiting time of data analysts.

Our contributions. In this paper, we develop a solution, called QUAD, in order to improve the performance of ϵKDV and τKDV. The main idea is to derive lower and upper bounds of the KDE function (i.e., Equation 1) in terms of quadratic functions (cf. Section 4). These *quadratic bounds* are theoretically tighter than the existing ones (in aKDE [20], tKDC [16], and KARL [10]), enabling faster pruning. In addition, many KDV-based applications [14, 18, 23, 30] also utilize other kernel functions, including triangular, cosine kernels etc. Therefore, we extend our techniques to support other kernel functions (cf. Section 5), which cannot be supported by the state-of-the-art solution, KARL [10]. In our experiments on large datasets in a single machine, QUAD is at least one-order-of-magnitude faster than existing solutions. For ϵKDV, QUAD takes 100-1000 sec to generate color map for each large-scale dataset (0.17M to 7M) with 2560×1920 pixels, using small relative error $\epsilon = 0.01$. However, most of the other methods fail to generate the color map within 2 hours under the same setting. For τKDV, QUAD can achieve nearly 10 sec with 1280×960 pixels, using different thresholds.

We further adopt a progressive visualization framework for KDV (cf. Section 6), in order to continuously output partial visualization results (by increasing the resolution). A user can terminate the process anytime, once the partial visualization results are satisfactory, instead of waiting for the precise color map to be generated. Experiment results show that we can achieve real-time (0.5 sec) in single machine without using GPU and parallel computation by combining this framework with our solution QUAD.

The rest of the paper is organized as follows. We first review existing work in Section 2. We then discuss the background in Section 3. Later, we present quadratic bound functions for KDE in Section 4. After that, we extend our quadratic bounds to other kernel functions in Section 5. We then discuss our progressive visualization framework for KDV in Section 6. Lastly, we show our results in Section 7, and conclude in Section 8. All the proofs are shown in Section 9.

2 RELATED WORK

Kernel density visualization (KDV) is widely used in many application domains, such as: ecological modeling [32, 33], crime [6, 23, 56] or traffic hotspot detection [48, 52, 54], chemical geology [49] and physical modeling [13]. For each application, they either need to compute the approximate kernel density values with theoretical guarantee [20] (ϵKDV) or

How to Categorize Papers into Different Groups?

- What is good categorization of papers?
  - Each category should not have too many/too few papers.
  - Every paper in a category should share a lot of similar properties.
  - Different categories should have significant differences.

- Bad categorization ☹
  1. Paper 1 uses a linear function to approximate the kernel density function.
  2. Paper 2 uses a quadratic function to approximate the kernel density function.
  3. Paper 3 uses a polynomial function to approximate the kernel density function.

What's wrong with this?

How to Categorize Papers into Different Groups?

- Good categorization ☺
  1. Paper A_1, Paper A_2, Paper A_3,… adopt the function approximation method for handling this problem.
  2. Paper B_1, Paper B_2, Paper B_3,… adopt the sampling method for handling this problem.
  3. Paper C_1, Paper C_2, Paper C_3,… adopt the computational geometry method for handling this problem.

Why is this categorization good?

How to Summarize Different Groups of Research Papers?

- Write down the core ideas of each group of research papers in your own words.

- Clearly point out the advantages and disadvantages.
  - Not just a copy-and-paste work.
  - Write these differences (or advantages/disadvantages) in your own words.

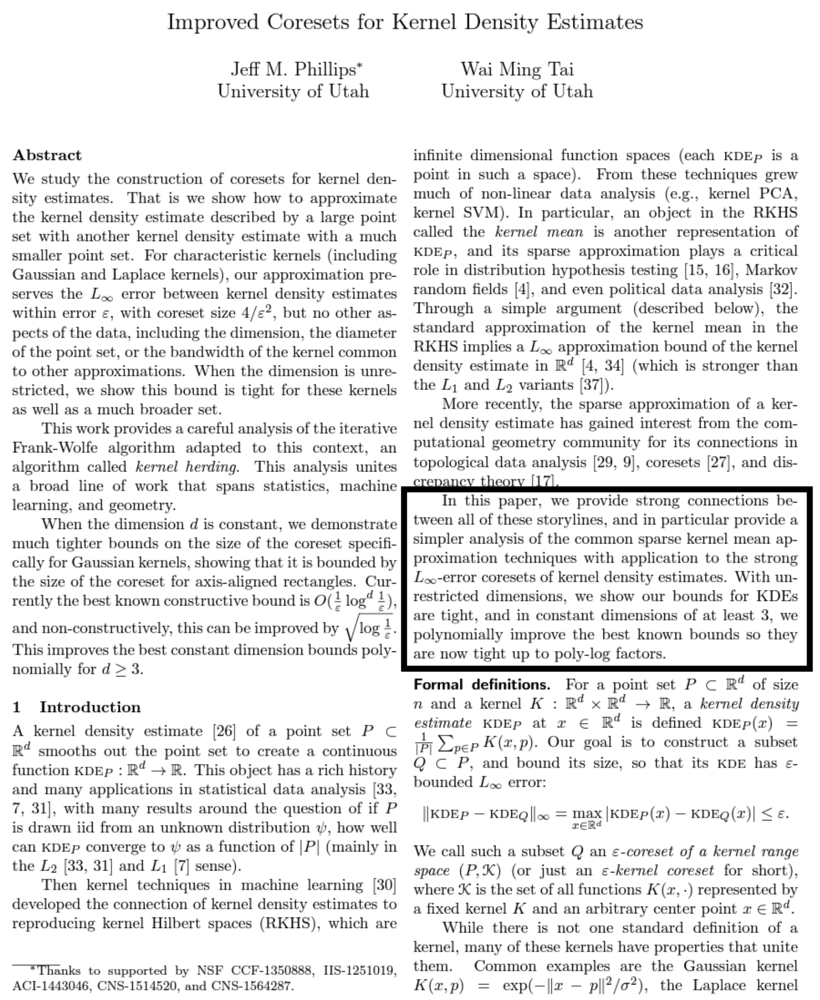
- Need to clearly understand different groups of papers.

How to Summarize Different Groups of Research Papers?

- Summarization of the sampling methods from some weak researchers.

"The main contribution is that they provide strong connections between all of these storylines, and in particular provide a simpler analysis of the common sparse kernel mean approximation techniques with application to the strong L∞-error coresets of kernel density estimates. With unrestricted dimensions, they show their bounds for KDEs are tight, and in constant dimensions of at least 3, they polynomially improve the best known bounds so these bounds are now tight up to poly-log factors."

No understanding



Improved Coresets for Kernel Density Estimates

Jeff M. Phillips*
University of Utah

Wai Ming Tai
University of Utah

Abstract

We study the construction of coresets for kernel density estimates. That is we show how to approximate the kernel density estimate described by a large point set with another kernel density estimate with a much smaller point set. For characteristic kernels (including Gaussian and Laplace kernels), our approximation preserves the L_∞ error between kernel density estimates within error ε, with coreset size $4/\varepsilon^2$, but no other aspects of the data, including the dimension, the diameter of the point set, or the bandwidth of the kernel common to other approximations. When the dimension is unrestricted, we show this bound is tight for these kernels as well as a much broader set.

This work provides a careful analysis of the iterative Frank-Wolfe algorithm adapted to this context, an algorithm called *kernel herding*. This analysis unites a broad line of work that spans statistics, machine learning, and geometry.

When the dimension d is constant, we demonstrate much tighter bounds on the size of the coreset specifically for Gaussian kernels, showing that it is bounded by the size of the coreset for axis-aligned rectangles. Currently the best known constructive bound is $O(\frac{1}{\varepsilon}\log^d\frac{1}{\varepsilon})$, and non-constructively, this can be improved by $\sqrt{\log\frac{1}{\varepsilon}}$. This improves the best constant dimension bounds polynomially for $d \geq 3$.

1 Introduction

A kernel density estimate [26] of a point set $P \subset \mathbb{R}^d$ smooths out the point set to create a continuous function $\text{KDE}_P : \mathbb{R}^d \to \mathbb{R}$. This object has a rich history and many applications in statistical data analysis [33, 7, 31], with many results around the question of if P is drawn iid from an unknown distribution ψ, how well can KDE_P converge to ψ as a function of $|P|$ (mainly in the L_2 [33, 31] and L_1 [7] sense).

Then kernel techniques in machine learning [30] developed the connection of kernel density estimates to reproducing kernel Hilbert spaces (RKHS), which are infinite dimensional function spaces (each KDE_P is a point in such a space). From these techniques grew much of non-linear data analysis (e.g., kernel PCA, kernel SVM). In particular, an object in the RKHS called the *kernel mean* is another representation of KDE_P, and its sparse approximation plays a critical role in distribution hypothesis testing [15, 16], Markov random fields [4], and even political data analysis [32]. Through a simple argument (described below), the standard approximation of the kernel mean in the RKHS implies a L_∞ approximation bound of the kernel density estimate in $\mathbb{R}^d$ [4, 34] (which is stronger than the L_1 and L_2 variants [37]).

More recently, the sparse approximation of a kernel density estimate has gained interest from the computational geometry community for its connections in topological data analysis [29, 9], coresets [27], and discrepancy theory [17].

In this paper, we provide strong connections between all of these storylines, and in particular provide a simpler analysis of the common sparse kernel mean approximation techniques with application to the strong L_∞-error coresets of kernel density estimates. With unrestricted dimensions, we show our bounds for KDEs are tight, and in constant dimensions of at least 3, we polynomially improve the best known bounds so they are now tight up to poly-log factors.

Formal definitions. For a point set $P \subset \mathbb{R}^d$ of size n and a kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, a *kernel density estimate* KDE_P at $x \in \mathbb{R}^d$ is defined $\text{KDE}_P(x) = \frac{1}{|P|}\sum_{p\in P} K(x,p)$. Our goal is to construct a subset $Q \subset P$, and bound its size, so that its KDE has ε-bounded L_∞ error:

$$\|\text{KDE}_P - \text{KDE}_Q\|_\infty = \max_{x\in\mathbb{R}^d} |\text{KDE}_P(x) - \text{KDE}_Q(x)| \leq \varepsilon.$$

We call such a subset Q an *ε-coreset of a kernel range space* $(P, \mathcal{K})$ (or just an *ε-kernel coreset* for short), where $\mathcal{K}$ is the set of all functions $K(x,\cdot)$ represented by a fixed kernel K and an arbitrary center point $x \in \mathbb{R}^d$.

While there is not one standard definition of a kernel, many of these kernels have properties that unite them. Common examples are the Gaussian kernel $K(x,p) = \exp(-\|x-p\|^2/\sigma^2)$, the Laplace kernel

*Thanks to supported by NSF CCF-1350888, IIS-1251019, ACI-1443046, CNS-1514520, and CNS-1564287.

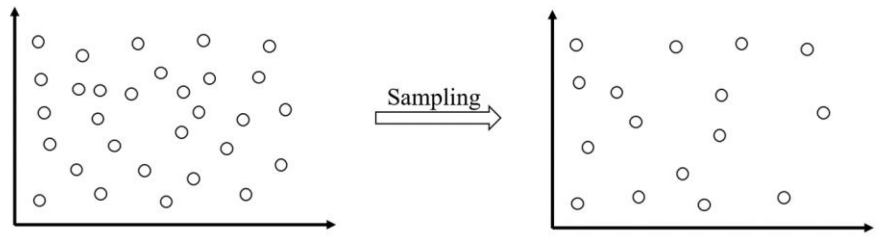
How to Summarize Different Groups of Research Papers?

- Summarization of the sampling methods from professional researchers.

Data Sampling

- Consider the kernel density function (with the Gaussian kernel).

$$\mathcal{F}_P(\mathbf{q}) = \sum_{\mathbf{p} \in P} w \cdot \exp\left(-\frac{1}{b^2} dist(\mathbf{q}, \mathbf{p})^2\right)$$



- Compute the modified kernel density function based on the sampled dataset S.

$$\mathcal{F}_S^{(M)}(\mathbf{q}) = \sum_{\mathbf{p}_i \in S} w_i \cdot \exp\left(-\frac{1}{b^2} dist(\mathbf{q}, \mathbf{p}_i)^2\right)$$

Advantages and Disadvantages of Data Sampling

- Advantages ☺
  - Can achieve probabilistic approximation guarantees for generating KDV.
  - Can reduce the worst-case time complexity for generating KDV.
  - Can handle all kernel functions.
- Disadvantages ☹
  - Cannot achieve exact solution.
  - Can still be slow for generating KDV.
  - Can degrade the practical visualization quality.

With deep understanding

How to Summarize the Main Differences Between Different Groups of Papers?

- Summarization needs your own view (the most difficult part for students).

- Be brave
  - Academic freedom protects you. ☺
    - No one will laugh at you.
    - No one will blame you even if you are wrong.
    - If they laugh at you or blame you, you can simply say this is academic freedom (I can say whatever I say as long as they are ethical or they do not break the law.).
  - Don’t be afraid if it is wrong.
    - “Everything you say” is only based on the best of your knowledge at that time.
    - You only need to say “I am sorry.” or “I learn it.” when it turns out to be wrong.

Examples of Summarization of Research Papers

- “Large-scale Geospatial Analytics: Problems, Challenges, and Opportunities” in SIGMOD 2023 Tutorial (from pages 27 - 36 in the slides from this link)

- “Kernel Density Visualization for Big Geospatial Data: Algorithms and Applications” in MDM 2023 Tutorial (from pages 18 - 32 in the slides from this link)

More Suggestions: Reading Papers

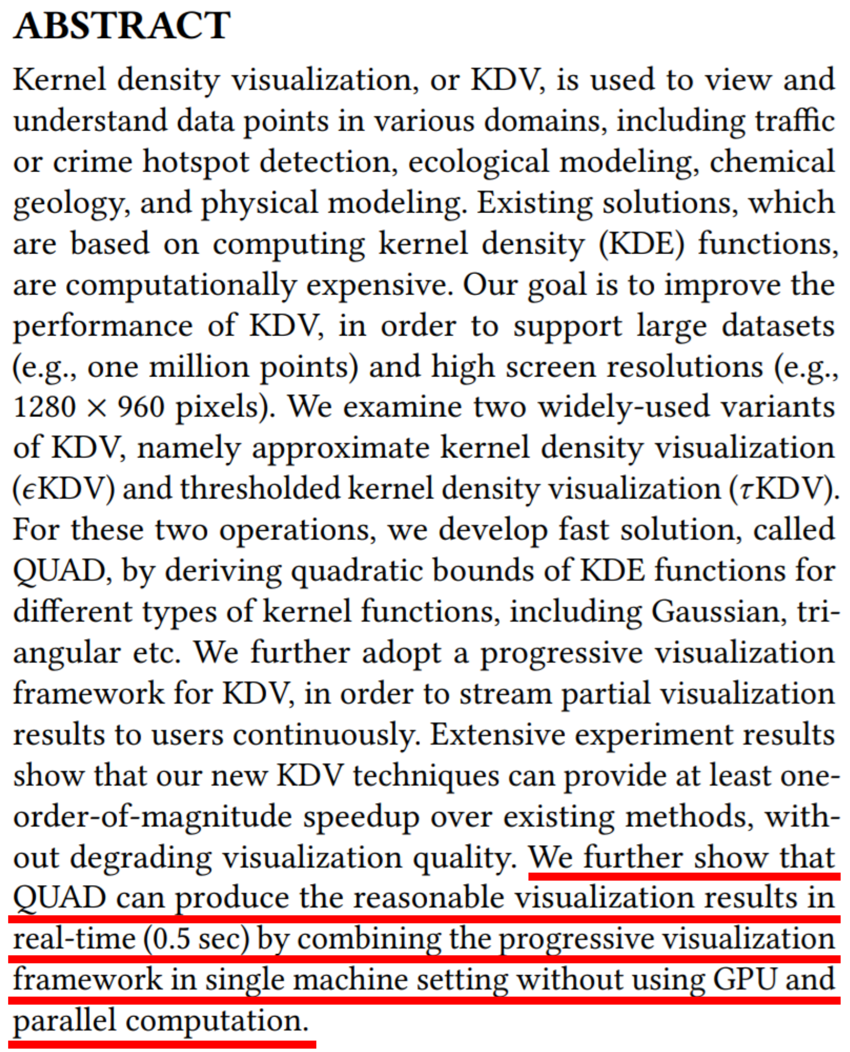
- Not necessary to read every word of the paper.
- Don’t believe everything in the paper.
- Don’t destroy your confidence.

More Suggestions: Reading Papers

- Suppose that you want to improve the efficiency of Kernel Density Visualization and you see this paper. What do you think?

- Some students: This is the end of the world.

- Some good researchers: Worth for investigating more.



ABSTRACT

Kernel density visualization, or KDV, is used to view and understand data points in various domains, including traffic or crime hotspot detection, ecological modeling, chemical geology, and physical modeling. Existing solutions, which are based on computing kernel density (KDE) functions, are computationally expensive. Our goal is to improve the performance of KDV, in order to support large datasets (e.g., one million points) and high screen resolutions (e.g., 1280×960 pixels). We examine two widely-used variants of KDV, namely approximate kernel density visualization (ϵKDV) and thresholded kernel density visualization (τKDV). For these two operations, we develop fast solution, called QUAD, by deriving quadratic bounds of KDE functions for different types of kernel functions, including Gaussian, triangular etc. We further adopt a progressive visualization framework for KDV, in order to stream partial visualization results to users continuously. Extensive experiment results show that our new KDV techniques can provide at least one-order-of-magnitude speedup over existing methods, without degrading visualization quality. We further show that QUAD can produce the reasonable visualization results in real-time (0.5 sec) by combining the progressive visualization framework in single machine setting without using GPU and parallel computation.

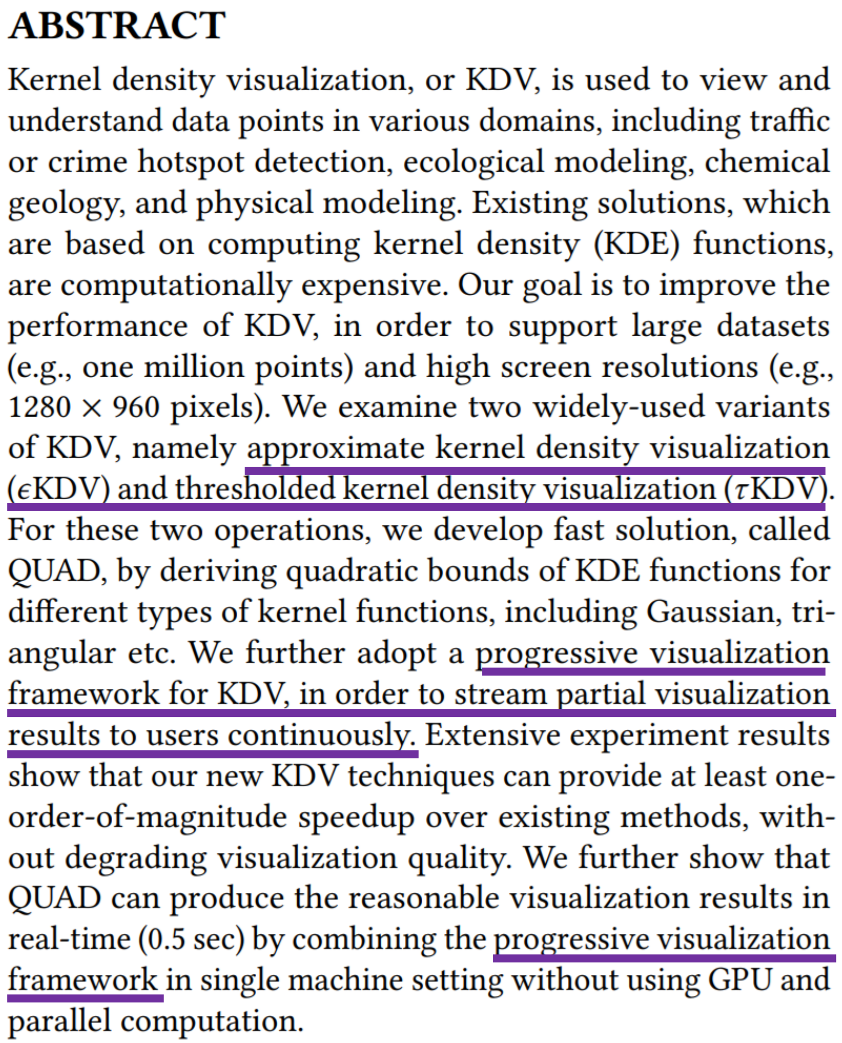
More Suggestions: Reading Papers

- They focus on developing approximate algorithms. How about exact algorithms?

- Maybe they use some “good” datasets.

- It is possible that progressive visualization framework provides bad results.

DOUBT

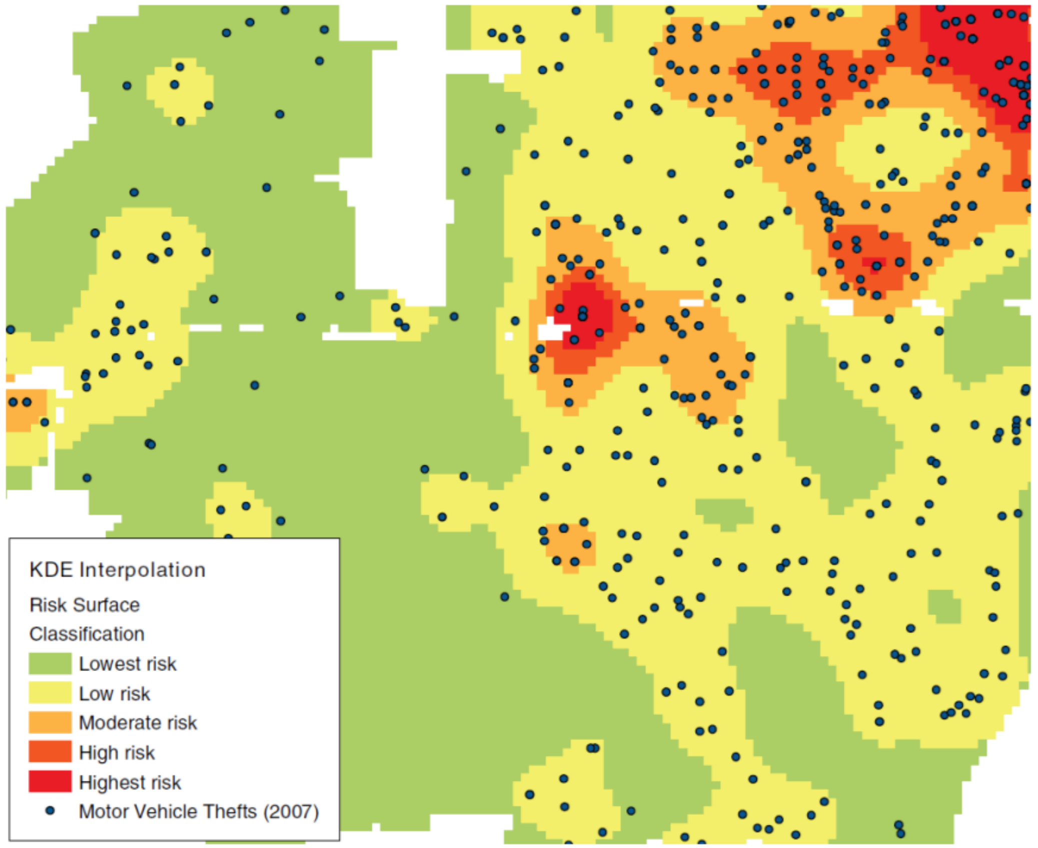


ABSTRACT

Kernel density visualization, or KDV, is used to view and understand data points in various domains, including traffic or crime hotspot detection, ecological modeling, chemical geology, and physical modeling. Existing solutions, which are based on computing kernel density (KDE) functions, are computationally expensive. Our goal is to improve the performance of KDV, in order to support large datasets (e.g., one million points) and high screen resolutions (e.g., 1280 × 960 pixels). We examine two widely-used variants of KDV, namely approximate kernel density visualization (ϵKDV) and thresholded kernel density visualization (τKDV). For these two operations, we develop fast solution, called QUAD, by deriving quadratic bounds of KDE functions for different types of kernel functions, including Gaussian, triangular etc. We further adopt a progressive visualization framework for KDV, in order to stream partial visualization results to users continuously. Extensive experiment results show that our new KDV techniques can provide at least one-order-of-magnitude speedup over existing methods, without degrading visualization quality. We further show that QUAD can produce the reasonable visualization results in real-time (0.5 sec) by combining the progressive visualization framework in single machine setting without using GPU and parallel computation.

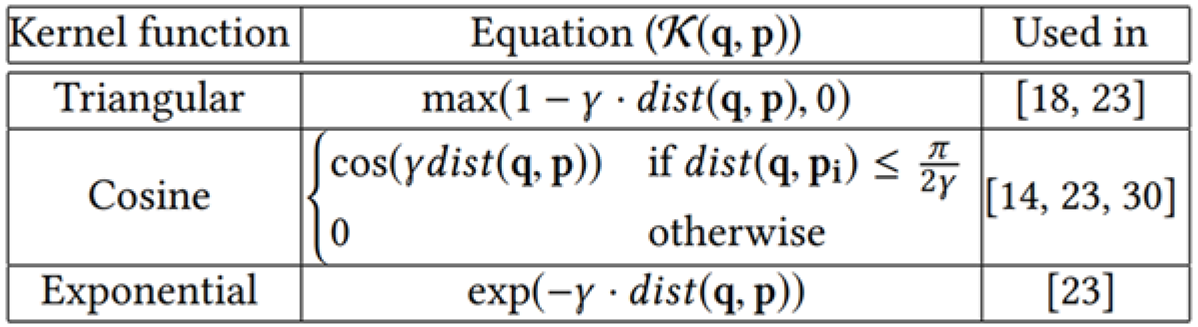
More Suggestions: Reading Papers

- Only focus on generating KDV in the planar space.
- How about the road network space?
- How about the temporal part of those data points?



More Suggestions: Reading Papers

- Only focus on Gaussian kernel function and those kernel functions in the following table.



Kernel function	Equation ($\mathcal{K}(\mathbf{q}, \mathbf{p})$)	Used in
Triangular	$\max(1 - \gamma \cdot dist(\mathbf{q}, \mathbf{p}), 0)$	[18, 23]
Cosine	$\begin{cases} \cos(\gamma dist(\mathbf{q}, \mathbf{p})) & \text{if } dist(\mathbf{q}, \mathbf{p_i}) \leq \frac{\pi}{2\gamma} \\ 0 & \text{otherwise} \end{cases}$	[14, 23, 30]
Exponential	$\exp(-\gamma \cdot dist(\mathbf{q}, \mathbf{p}))$	[23]

- How about other kernels? (Epanechnikov kernel and quartic kernel are also very famous.)

NOT NECESSARY TO FOLLOW THE SAME SETTING

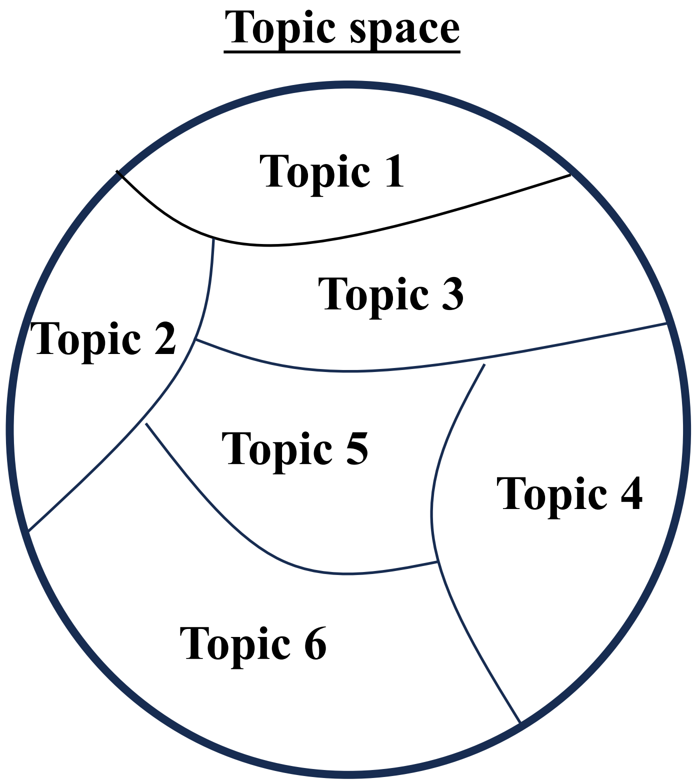
How to Find Other Settings?

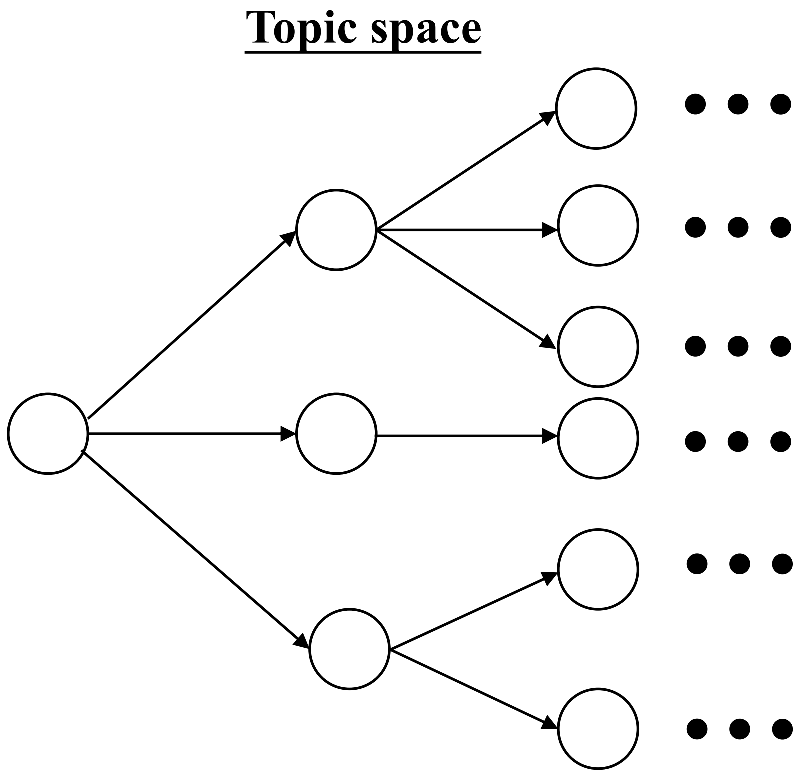
- Keep answering questions when you are reading papers.
  - How about the road network space? [a]
  - How about the temporal part of those data points? [b]
  - How about other kernels? [a, b, c]

[a] **Tsz Nam Chan**, Zhe Li, Leong Hou U, Jianliang Xu, Reynold Cheng: "Fast Augmentation Algorithms for Network Kernel Density Visualization" **PVLDB 2021** (vol 14), pages 1503-1516.
[b] **Tsz Nam Chan**, Pak Lon Ip, Leong Hou U, Byron Choi, Jianliang Xu: "SWS: A Complexity-Optimized Solution for Spatial-Temporal Kernel Density Visualization" **PVLDB 2022** (vol 15), pages 814-827.
[c] **Tsz Nam Chan**, Leong Hou U, Byron Choi, Jianliang Xu: "SLAM: Efficient Sweep Line Algorithms for Kernel Density Visualization" **SIGMOD 2022**, pages 2120-2134.

- Possible for you to have new problem settings (i.e., new papers) by answering those questions.

More Suggestions: Writing Papers

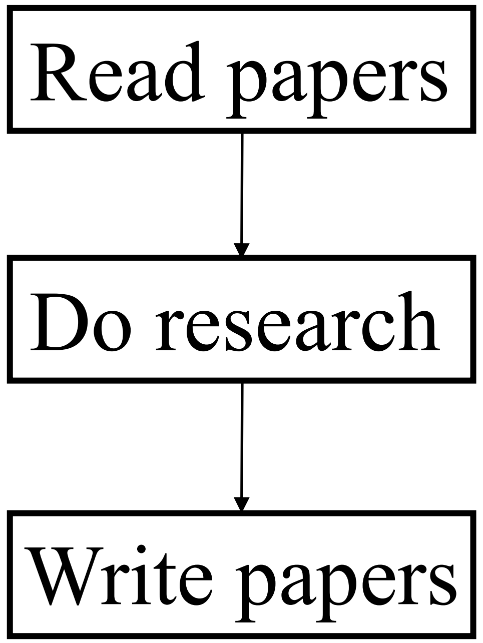




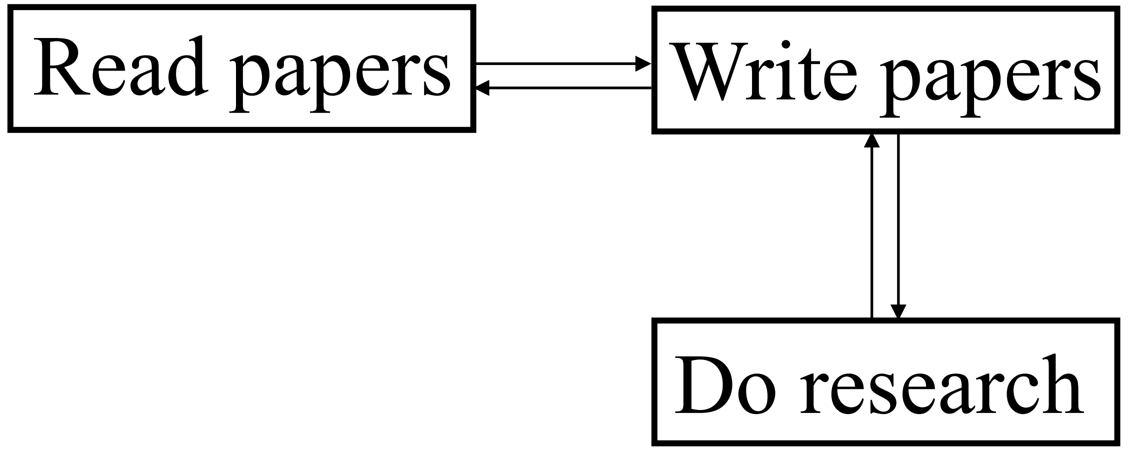
Weak researcher: There will be no topic to work with when I get a tenure-track position. I need to retain some of them for publishing when I get the tenure-track position.

Strong researcher: The more I write, the more topics I will have.

More Suggestions: Reading and Writing Papers



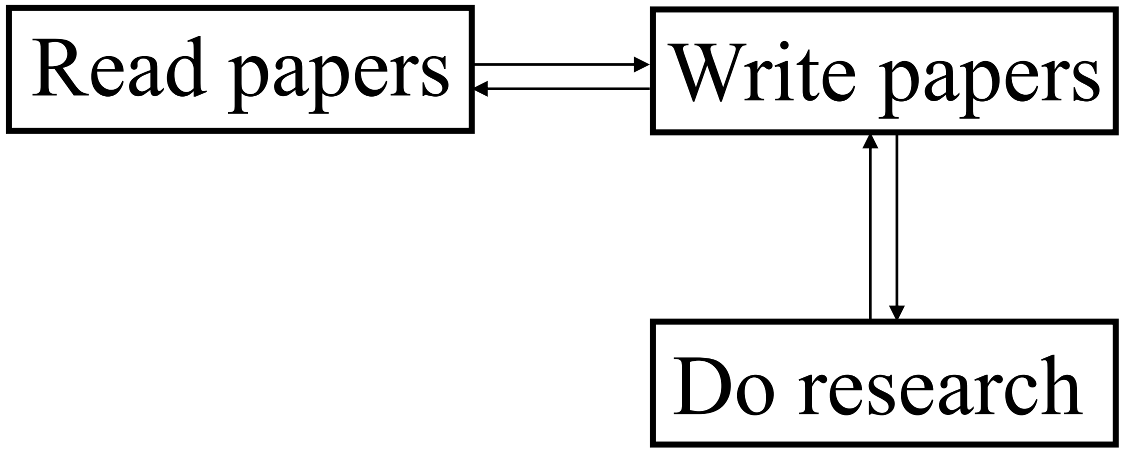
Junior students



“Senior” students

More Suggestions: Reading and Writing Papers

1. Want to work on KDV.
2. Start reading some papers in KDV.
3. I have learnt somethings. Write it down in the draft. Moreover, there are still some questions (e.g., Can indexing structures be used in KDV?).
4. Check the literature again and figure out the answers and ask some additional questions…



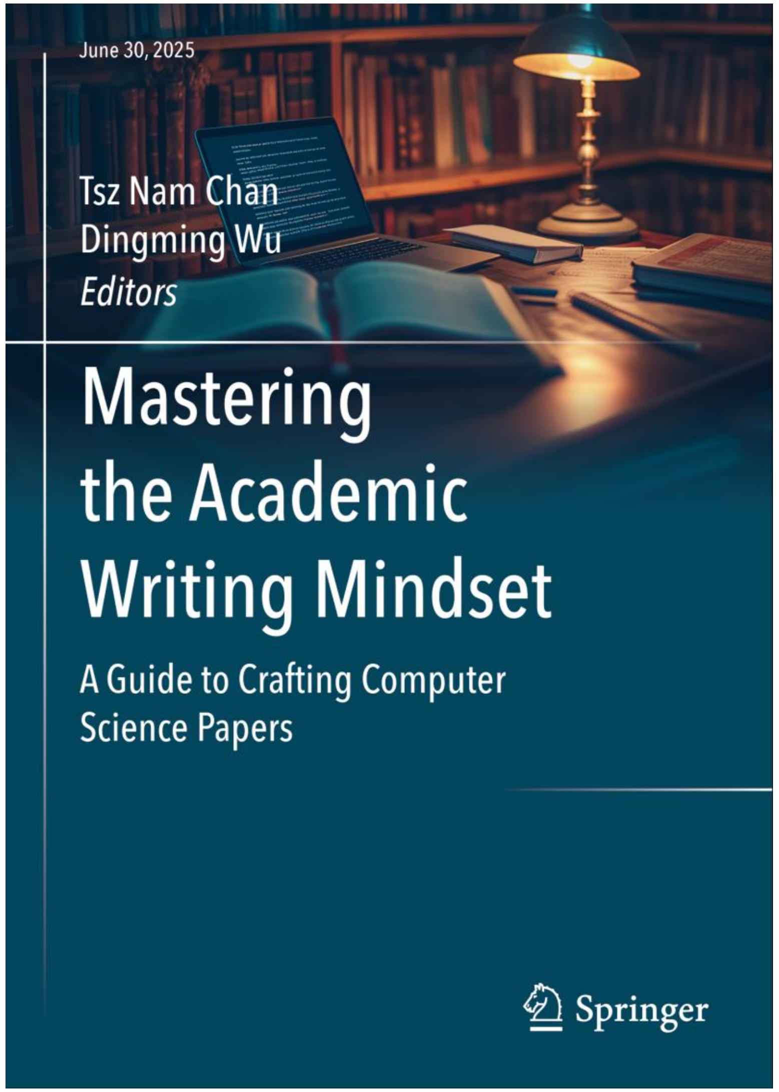
More Suggestions: Reading and Writing Papers

5. Are those indexing structure good? (Do some research)
6. I have learnt somethings. Write it down in the draft. Moreover, there are still some other questions.
7. Check the literature again and figure out the answers and ask some additional questions…
8. Come up with new idea.
9. Write it down in the draft and do research.

$$\begin{tikzcd}
\boxed{\text{Read papers}} \arrow[r, shift left] & \boxed{\text{Write papers}} \arrow[l, shift left] \arrow[d, shift left] \\
& \boxed{\text{Do research}} \arrow[u, shift left]
\end{tikzcd}$$

$\boxed{\text{Write every day!}}$

Our New Book



June 30, 2025

Tsz Nam Chan
Dingming Wu
Editors

Mastering the Academic Writing Mindset

A Guide to Crafting Computer Science Papers

Springer

Tsz Nam Chan, Dingming Wu

Mastering the Academic Writing Mindset

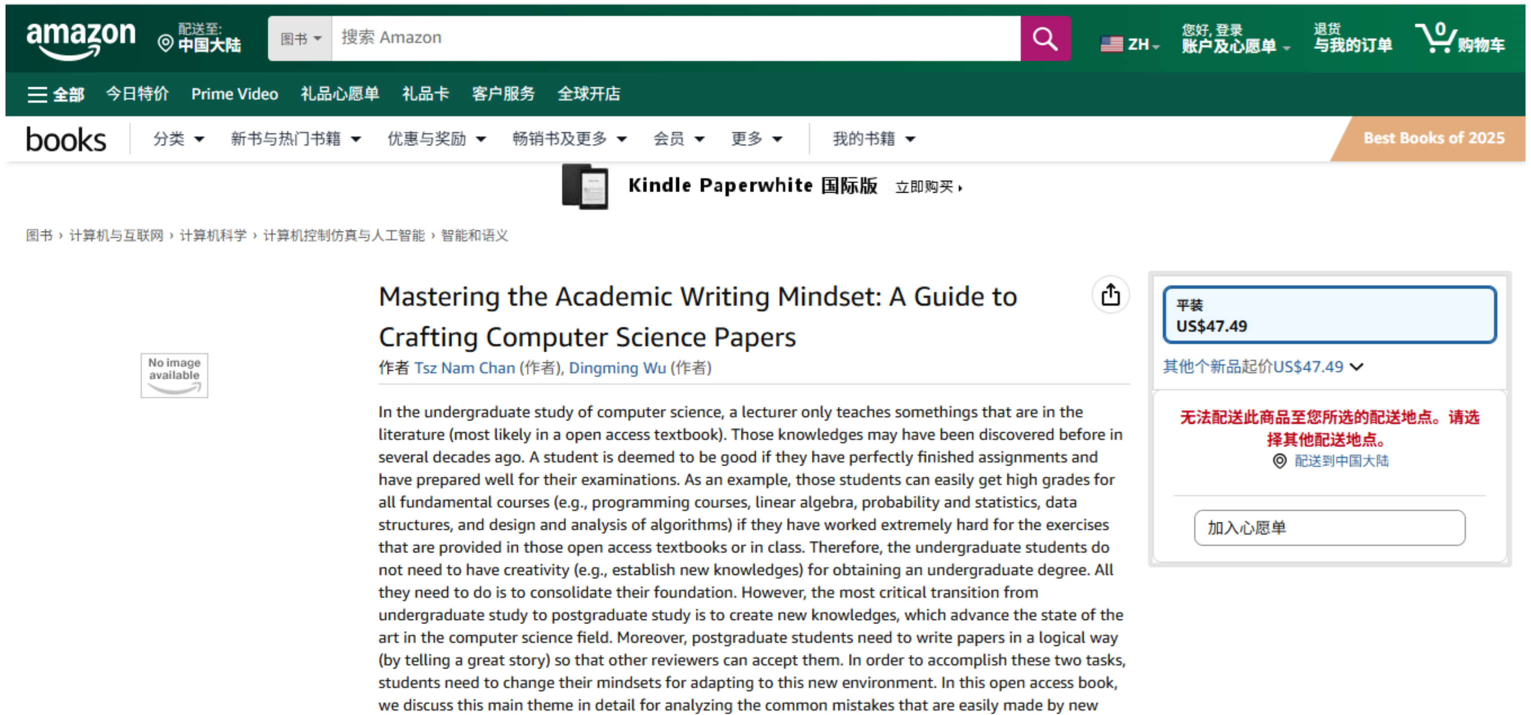
A Guide to Crafting Computer Science Papers

September 12, 2025

Springer Nature

Our New Book

- Hard copy can be bought in some bookstores (e.g., Amazon).
  - My signature is free. ☺
- Soft copy is available online (open access).
- Available on 31st March 2026.



Mastering the Academic Writing Mindset: A Guide to Crafting Computer Science Papers

作者 Tsz Nam Chan (作者), Dingming Wu (作者)

In the undergraduate study of computer science, a lecturer only teaches somethings that are in the literature (most likely in a open access textbook). Those knowledges may have been discovered before in several decades ago. A student is deemed to be good if they have perfectly finished assignments and have prepared well for their examinations. As an example, those students can easily get high grades for all fundamental courses (e.g., programming courses, linear algebra, probability and statistics, data structures, and design and analysis of algorithms) if they have worked extremely hard for the exercises that are provided in those open access textbooks or in class. Therefore, the undergraduate students do not need to have creativity (e.g., establish new knowledges) for obtaining an undergraduate degree. All they need to do is to consolidate their foundation. However, the most critical transition from undergraduate study to postgraduate study is to create new knowledges, which advance the state of the art in the computer science field. Moreover, postgraduate students need to write papers in a logical way (by telling a great story) so that other reviewers can accept them. In order to accomplish these two tasks, students need to change their mindsets for adapting to this new environment. In this open access book, we discuss this main theme in detail for analyzing the common mistakes that are easily made by new

平装
US$47.49

其他个新品起价US$47.49

无法配送此商品至您所选的配送地点。请选择其他配送地点。

配送到中国大陆