KARL: Fast Kernel Aggregation Queries

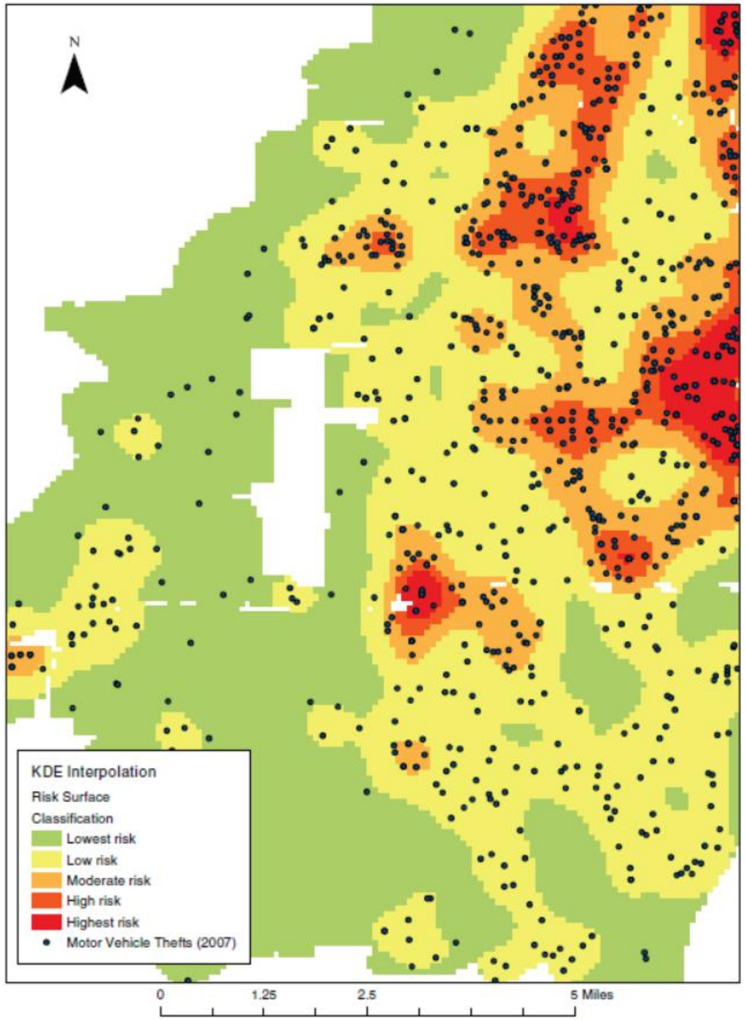
(Edison) Tsz Nam Chan[1,2], Man Lung Yiu[2], Leong Hou U[3]

[1]Univ. of Hong Kong [2]Hong Kong Polytechnic Univ. [3]Univ. of Macau

What is Kernel Aggregation?

Compute the function: $$\underbrace{\mathcal{F}_P}_{\text{dataset}}(\overbrace{\mathbf{q}}^{\text{query}}) = \sum_{\mathbf{p_i}\in P} \overbrace{w_i}^{\text{weighting}} \exp(-\underbrace{\gamma}_{\text{constant}} \cdot \overbrace{dist(\mathbf{q},\mathbf{p_i})}^{\text{Euclidean distance}}{}^2)$$

- *Application*: crime rate prediction
  - Each **p** (black dot) represents the location of a crime
    - e.g., robbery,
      commercial burglary,
      motor vehicle theft
  - Predict the crime rate of a given location (**q**) by computing $\mathcal{F}_P(\mathbf{q})$



T. Hart and P. Zandbergen “Kernel density estimation and hotspot mapping Examining the influence of interpolation method, grid cell size, and bandwidth on crime forecasting”, International Journal of Police Strategies and Management 2014

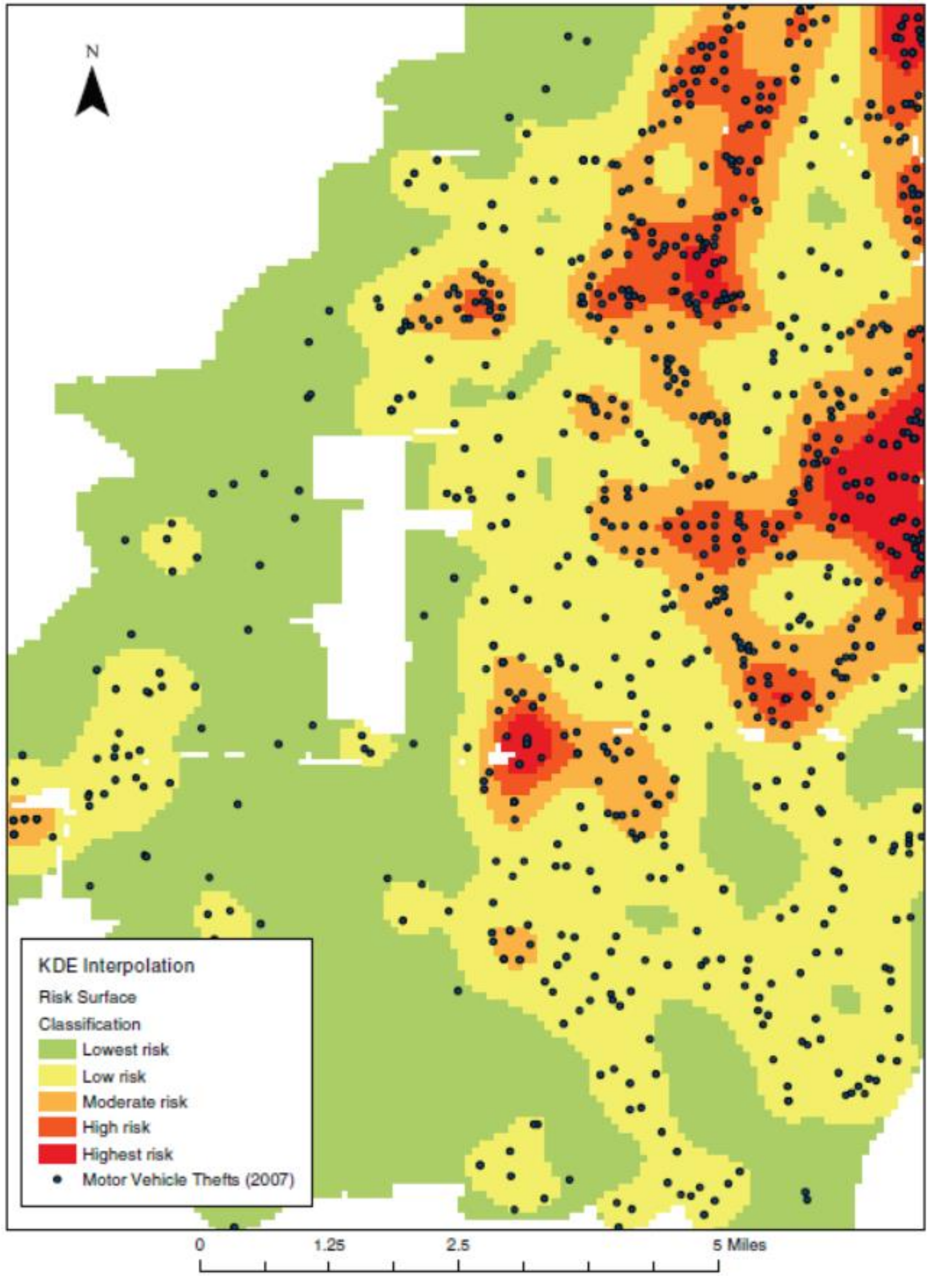
Kernel Aggregation: More Applications

Kernel aggregation function: $$\underbrace{\mathcal{F}_P}_{\text{dataset}}(\overbrace{\mathbf{q}}^{\text{query}}) = \sum_{\mathbf{p_i} \in P} \overbrace{w_i}^{\text{weighting}} \exp(-\underbrace{\gamma}_{\text{constant}} \cdot \overbrace{dist(\mathbf{q}, \mathbf{p_i})}^{\text{Euclidean distance}}{}^2)$$

Type of weighting	**Used in model**	**Application(s)**
Type I: identical, positive w_i (most specific)	Kernel Density Classification	Crime rate prediction Ecological modeling Particle Searching
Type II: positive w_i (subsuming Type I)	1-class SVM	Network fault detection Outlier detection
Type III: no restriction on w_i (subsuming Type II)	2-class SVM	Network fault detection Tumor samples classification Image classification

Kernel Aggregation Queries

- *Application*: crime rate prediction
- Just classify $\mathcal{F}_P(\mathbf{q})$ into
  - Lowest risk: [0, 10)
  - Low risk:: [10, 20)
  - Moderate risk: [20, 50)
  - High risk: [50, 100)
  - Highest risk: [100, ∞)
- Threshold kernel aggregation query (τ-KAQ)
  - Input: query vector $\mathbf{q}$, dataset P, threshold τ
  - Output: Boolean value
    - true (if $\mathcal{F}_P(\mathbf{q}) \geq \tau$) or
    - false (if $\mathcal{F}_P(\mathbf{q}) < \tau$)



T. Hart and P. Zandbergen "Kernel density estimation and hotspot mapping Examining the influence of interpolation method, grid cell size, and bandwidth on crime forecasting", International Journal of Police Strategies and Management 2014

Expensive to compute the exact $\mathcal{F}_P(\mathbf{q})$!

Le et al. *"Despite their successes, what makes kernel methods difficult to use in many large scale problems is the fact that* ***computing the decision function is typically expensive, especially at prediction time.****"* [1]

$$\mathcal{F}_P(\mathbf{q}) = \underbrace{\sum_{\mathbf{p_i} \in P} w_i \exp(-\gamma \cdot dist(\mathbf{q}, \mathbf{p_i})^2)}_{\text{O(|P|} \times \text{d) time}}$$

What are typical values of |P| and d?

[1] Quoc V. Le, Tamas Sarlos, and Alexander J. Smola. "Fastfood- computing hilbert space expansions in loglinear time." In ICML2013

Our contribution: KARL

- Kernel Aggregation Rapid Library (KARL)
  - **2.5-738x** speed up over state-of-the-art in different datasets
  - The **Fastest** library in **online prediction phase**

Library	**Support indexing**	**Response time**
LibSVM	no	high
Scikit-learn	yes	high
KARL (this paper)	yes	low

KARL: https://github.com/edisonchan2013928/KARL-Fast-Kernel-Aggregation-Queries
LibSVM: https://www.csie.ntu.edu.tw/~cjlin/libsvm/
Scikit-learn: https://scikit-learn.org/

How to speed up?

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

$$LB(\mathbf{q}) \leq \mathcal{F}_P(\mathbf{q}) \leq UB(\mathbf{q})$$

Requirements:

1. Fast to compute, e.g., O(d) time
2. As tight as possible

<u>Cases to avoid computing exact $\mathcal{F}_P(\mathbf{q})$:</u>

Case 1 ($\mathcal{F}_P(\mathbf{q}) \geq \tau$):

$UB(\mathbf{q})$

$LB(\mathbf{q})$

τ

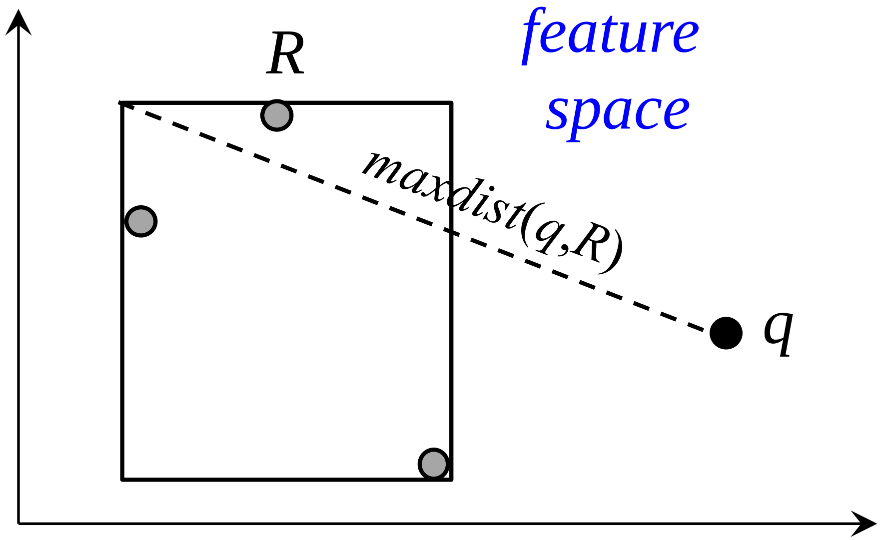
Case 2 ($\mathcal{F}_P(\mathbf{q}) < \tau$):

τ

$UB(\mathbf{q})$

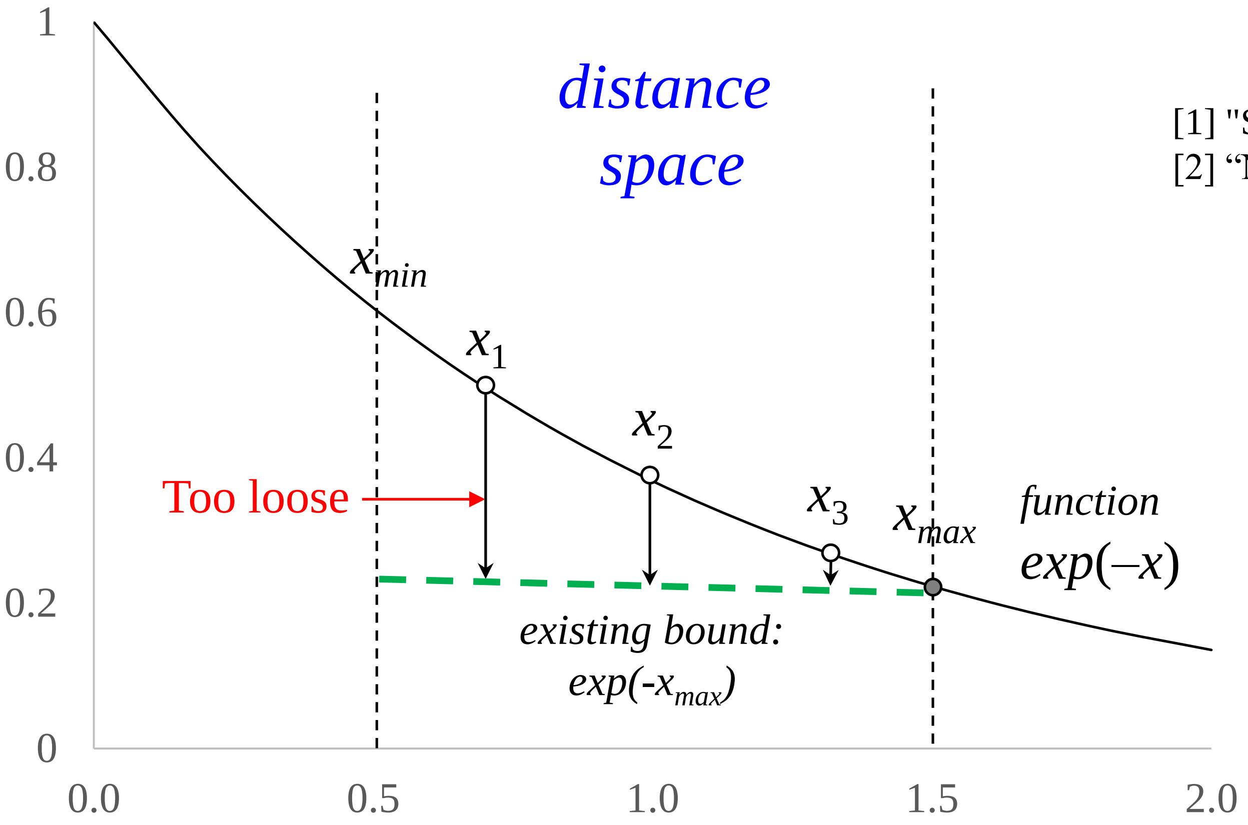
$LB(\mathbf{q})$

Existing Bounding Function



$$\mathcal{F}_P(\mathbf{q}) = \sum_{\mathbf{p_i} \in P} w \exp(-\underbrace{\gamma \cdot dist(\mathbf{q}, \mathbf{p_i})^2}_{\mathrm{x_i}})$$

$$LB_R(\mathbf{q}) = w \cdot R.\mathrm{count} \cdot \exp(-\underbrace{\gamma \cdot maxdist(\mathbf{q}, R)^2}_{\mathrm{x_{max}}})$$



[1] "Scalable kernel density classification via threshold based pruning" SIGMOD2017
[2] "Nonparametric Density Estimation: Toward Computational Tractability" SDM03

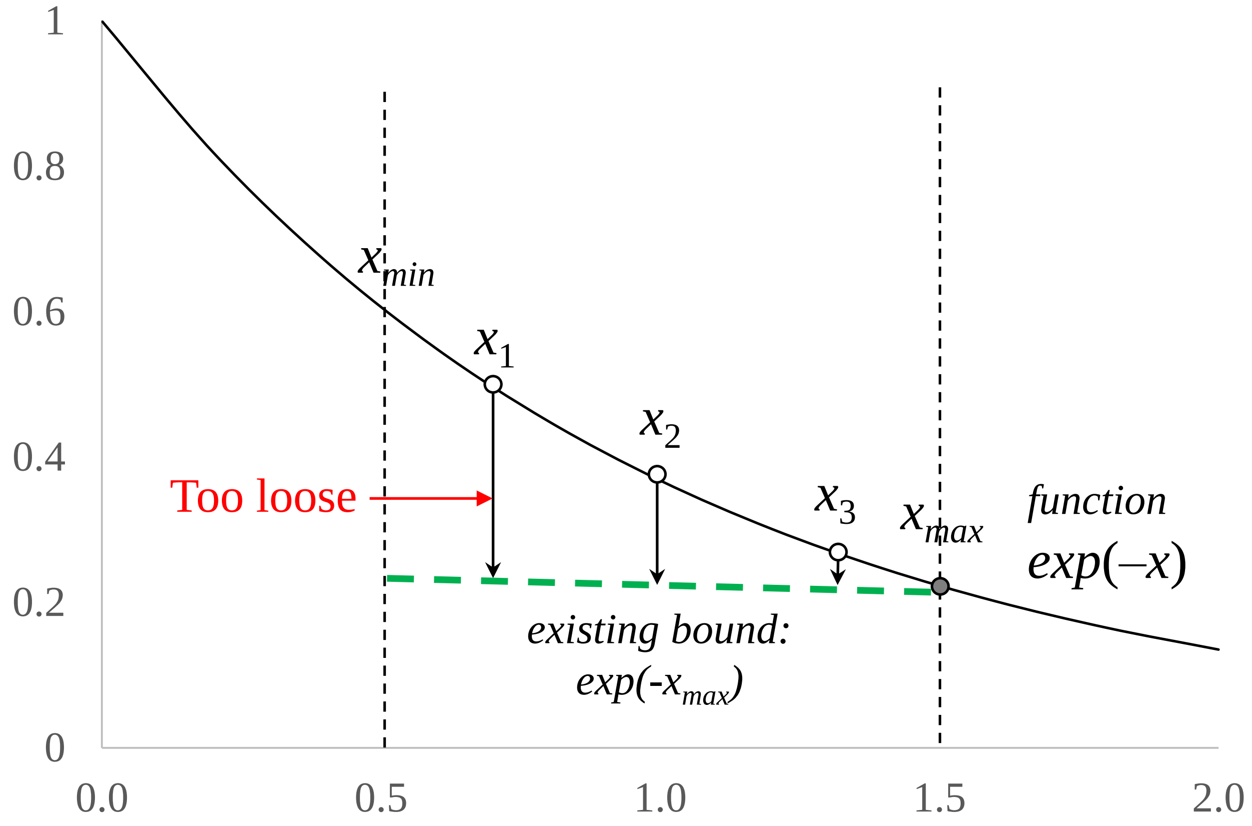
Summary:

☑ Fast to compute, e.g., O(d) time

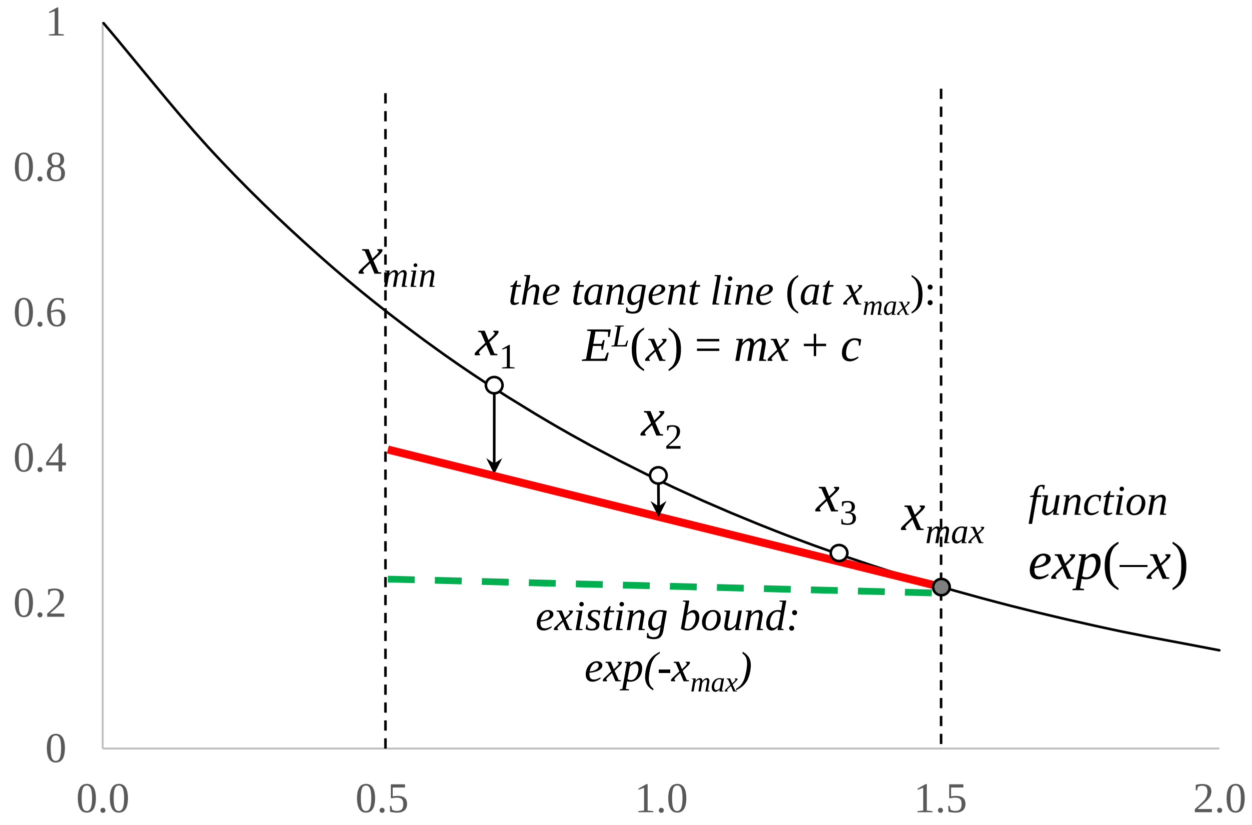
☒ Too loose

Research question 1:

How to develop a **tighter** lower bound function (than the existing one)?



Our Idea: Tangent Bound



$$\mathcal{F}_P(\mathbf{q}) = \sum_{\mathbf{p_i} \in P} w \exp(-\underbrace{\gamma \cdot dist(\mathbf{q}, \mathbf{p_i})^2}_{\mathrm{x_i}})$$

<u>Tangent Bound:</u>

$$\mathcal{FL}_P(\mathbf{q}, Lin_{m,c}) = \sum_{\mathbf{p_i} \in P} w\,(m(\underbrace{\gamma dist(\mathbf{q}, \mathbf{p_i})^2}_{\mathrm{x_i}}) + c)$$

☺ always tighter than the existing bound

<u>Research question 2:</u>

How to compute this tighter bound quickly?

O(d)-time Tighter Linear Bound

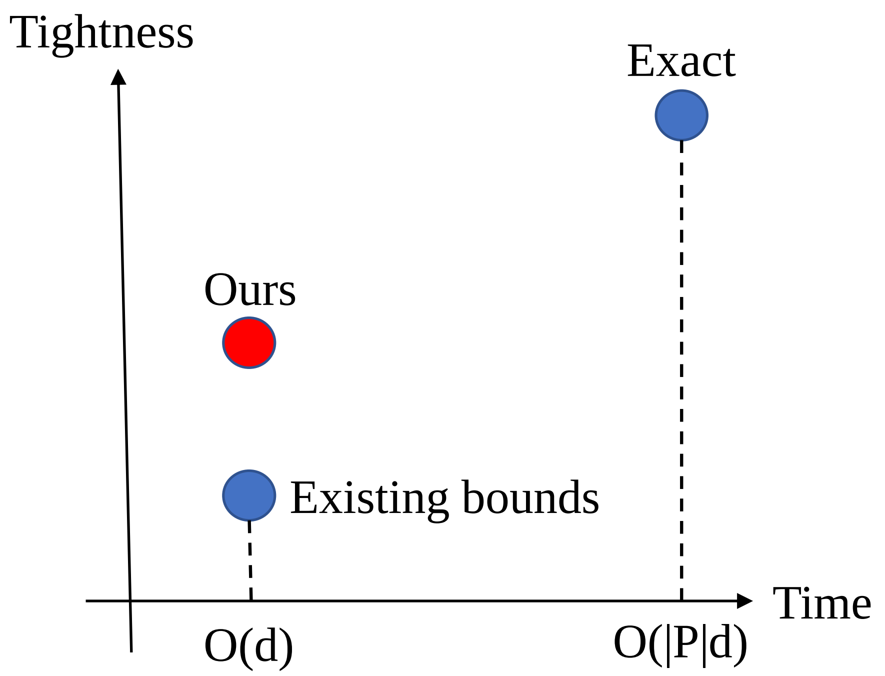
$$\mathcal{F}_P(\mathbf{q}) = \sum_{\mathbf{p_i} \in P} w \exp(-\underbrace{\gamma \cdot dist(\mathbf{q}, \mathbf{p_i})^2}_{\mathrm{x_i}})$$

Linear bound of $\mathcal{F}_P(\mathbf{q})$:

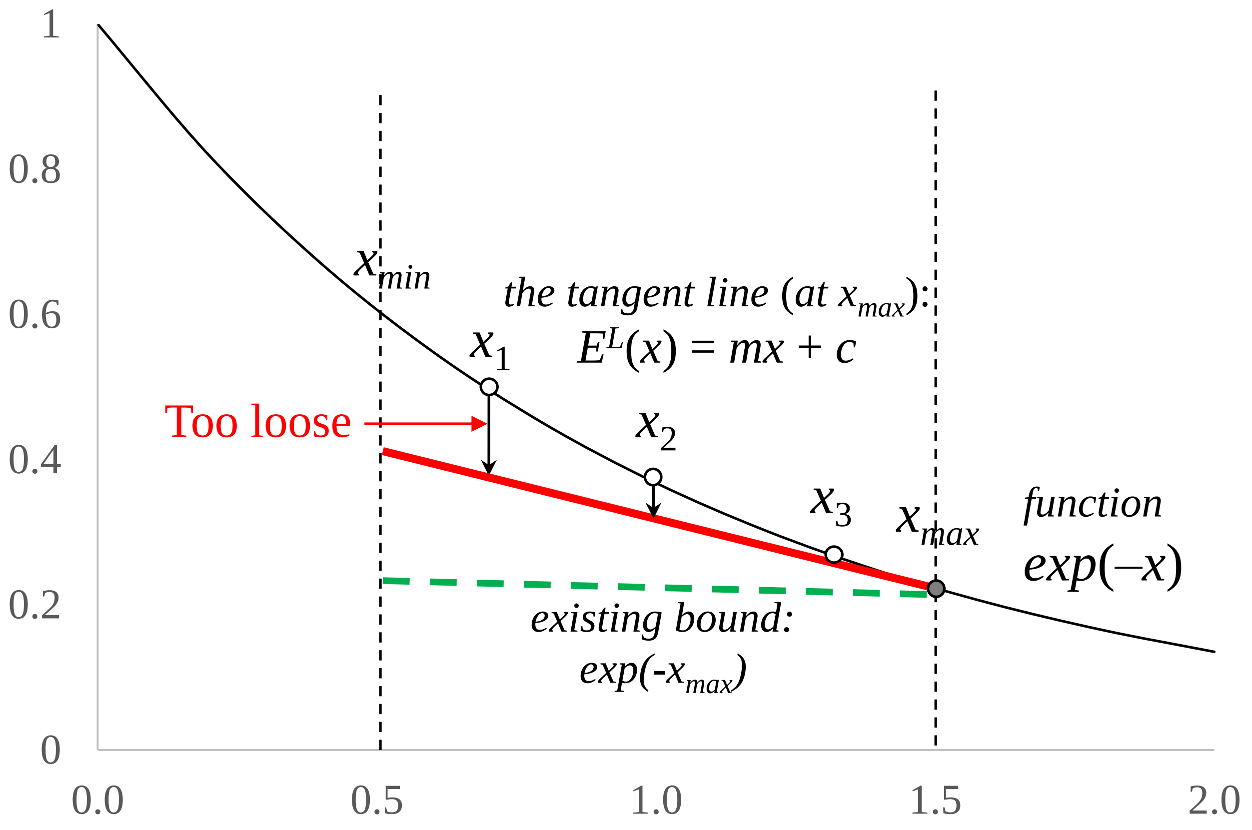
$$\mathcal{FL}_P\left(\mathbf{q}, Lin_{m,c}\right) = \sum_{\mathbf{p_i} \in P} w\,(m(\underbrace{\gamma dist(\mathbf{q}, \mathbf{p_i})^2}_{\mathrm{x_i}}) + c)$$

$$\mathcal{FL}_P\left(\mathbf{q}, Lin_{m,c}\right) = wm\gamma(|P|\underbrace{\|\mathbf{q}\|^2}_{\mathrm{O(d)}} - 2\underbrace{\mathbf{q} \cdot \mathbf{a_P}}_{\mathrm{O(d)}} + b_P) + wc|P|$$

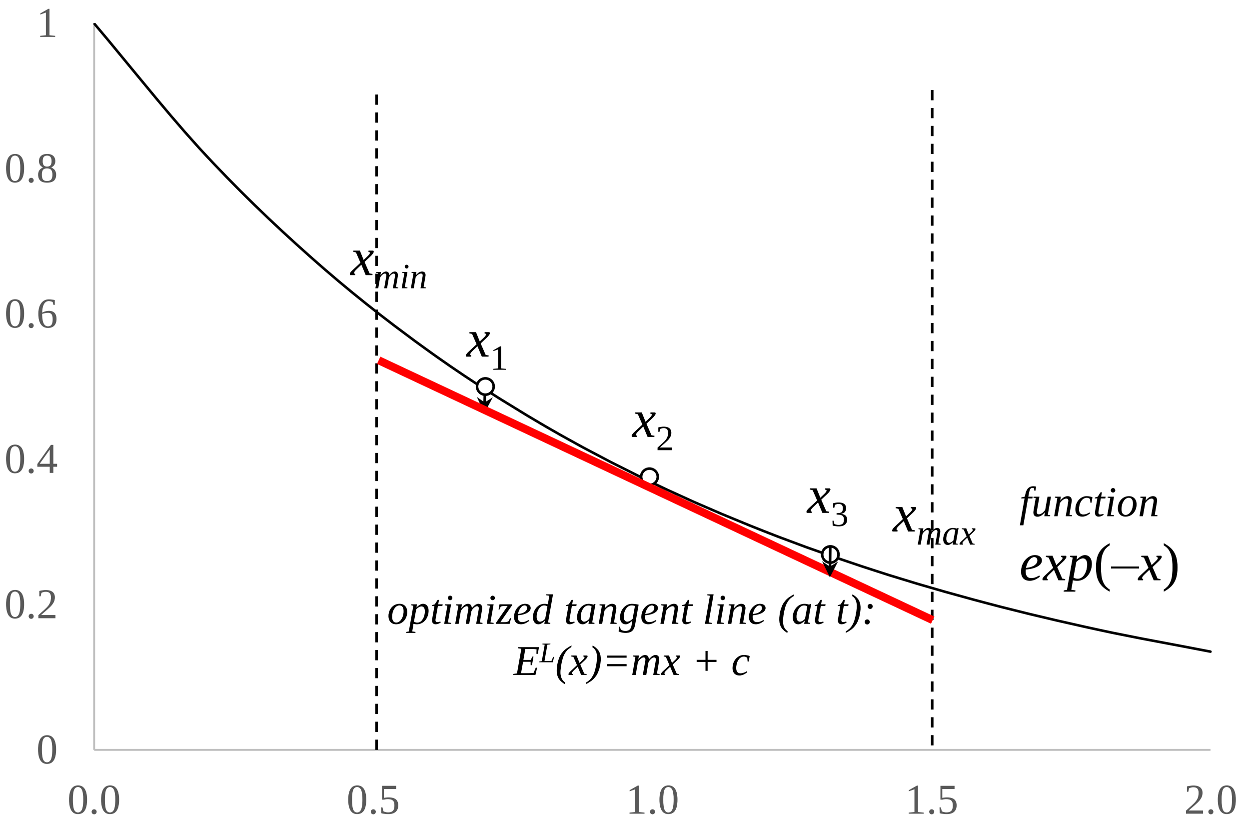
where $\mathbf{a_P} = \sum_{\mathbf{p_i} \in P} \mathbf{p_i}$ and $b_P = \sum_{\mathbf{p_i} \in P} \|\mathbf{p_i}\|^2$



Research question 3:
How to further tighten our bound (tangent bound)?



Observation:
The optimal tangent line depends on the distance values (e.g., x_1, x_2, x_3)



Challenge:
How to compute the optimal tangent **efficiently** (e.g., O(d) time) **without** knowing the exact distance values (e.g., x_1, x_2, x_3)?

Solved in our paper

Experimental Results

Methods: SCAN (baseline), LIBSVM, SOTA and KARL
Datasets: UCI Machine Learning Repository/ LibSVM website

Throughput (Queries/sec)

Type	Datasets	SCAN	LIBSVM	SOTA	KARL
I-τ	miniboone	36.1	34	102	**510**
	home	15.2	14.1	93.2	**258**
	susy	2.02	1.86	3.58	**83.4**
II-τ	nsl-kdd	283	481	748	**20668**
	kdd99	260	520	1269	**11324**
	covtype	158	462	448	**6022**
III-τ	ijcnn1	903	1170	1119	**826928**
	a9a	162	610	546	**6885**
	covtype-b	13	38.4	33.9	**274**

SOTA: E. Gan and P. Bailis. "Scalable kernel density classification via threshold based pruning" SIGMOD2017

Conclusion and Future Work

- What we have done:
    1. Develop Kernel Aggregation Rapid Library (KARL)
        1. Support a wide range of models (in prediction stage)
            1. Kernel Density Classification
            2. One Class SVM
            3. Two Class SVM
        2. Achieve higher throughput than the state-of-the-art by 2.5-738x times
- Future Work:
    1. Integrate the implementation into the library (Ongoing)
    2. Support different types of machine learning models (Ongoing)
    3. Support different types of kernel functions (Ongoing)
    4. Support interesting applications, e.g., Kernel Density Visualization (Ongoing)

For more details…

- The Github page is in **Page 2** of our paper

Our proposal is **K**ernel **A**ggregation **R**apid **L**ibrary (KARL)[1], a comprehensive solution for addressing all the issues mentioned above. It utilizes a novel bounding technique and index tuning in order to achieve excellent efficiency. Experimental studies on many real datasets reveal that our proposed method achieves speedups of 2.5-738 over the state-of-the-art.

Two widely-used libraries, namely LibSVM [8] and Scikit-learn [30], provide convenient programming support for practitioners to handle kernel aggregation queries. Implementation-wise, LibSVM is based on the sequential scan method, and Scikit-learn is based on the algorithm in [16] for query type I. We compare them with our proposal (KARL) in Table II. As a remark, since Scikit-learn supports query types II and III via the wrapper of LibSVM [30], we remove those two query types from the row of Scikit-learn in Table II. The features of KARL are: (i) it supports all three types of weightings

[1]https://github.com/edisonchan2013928/KARL-Fast-Kernel-Aggregation-Queries