Nearest-Neighbor Search in NLP Applications using the Non-Metric Space Library (NMSLIB)

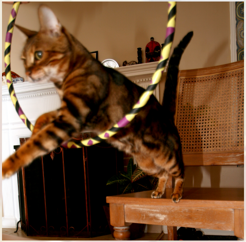
Leo (Leonid) Boytsov

https://github.com/searchivarius/NonMetricSpaceLib

Nearest-Neighbor Search in NLP Applications using the Non-Metric Space Library (NMSLIB)

Leo (Leonid) Boytsov

https://github.com/searchivarius/NonMetricSpaceLib



Acknowledgements and Some History

- Supported by CMU's OAQA project[1] and the European iAD[2] center.

- Code was written mostly by Bileg(saikhan) Naidan (NTNU) and Leo(nid) Boytsov (CMU)

- Catalyzed by an 11-701 course project

- Includes contributions from several people: Lawrence Cayton, Wei Dong, Avrelin Nikita, Alexander Ponomarenko, Yury Malkov, Daniel Lemire

[1] https://github.com/oaqa

[2] http://www.iad-center.com/

Talk Outline

Talk Outline

- Definition of the nearest-neighbor (NN) search
    - Importance of non-metric search

Talk Outline

- Definition of the nearest-neighbor (NN) search
  - Importance of non-metric search
- Survey of applications in ML/NLP
  - Is NN search useful?

Talk Outline

- Definition of the nearest-neighbor (NN) search
  - Importance of non-metric search
- Survey of applications in ML/NLP
  - Is NN search useful?
- Targeted mini-survey of the state of the art
  - How can ML improve NN search?
  - Some state-of-the-art comparisons

Talk Outline

- Definition of the nearest-neighbor (NN) search
  - Importance of non-metric search
- Survey of applications in ML/NLP
  - Is NN search useful?
- Targeted mini-survey of the state of the art
  - How can ML improve NN search?
  - Some state-of-the-art comparisons
- Overview of the Non-Metric Space Library (NMSLIB)
  - Technical details
  - More state-of-the-art comparisons
  - Future work

Part I

Problem Statement

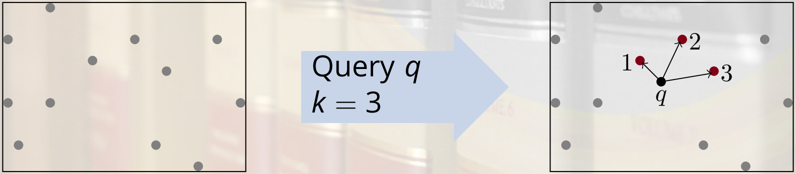
Importance of Non-Metric Access Methods

Nearest-neighbor search (k-NN search)

- **Input:** A set of n data points (objects) and a distance function $d(x, y)$
- **Query:** New object q and k
- **Task:** Quickly find k most similar objects in the data set to q

Nearest-neighbor search (k-NN search)

- **Input:** A set of n data points (objects) and a distance function $d(x, y)$
- **Query:** New object q and k
- **Task:** Quickly find k most similar objects in the data set to q



Distance functions

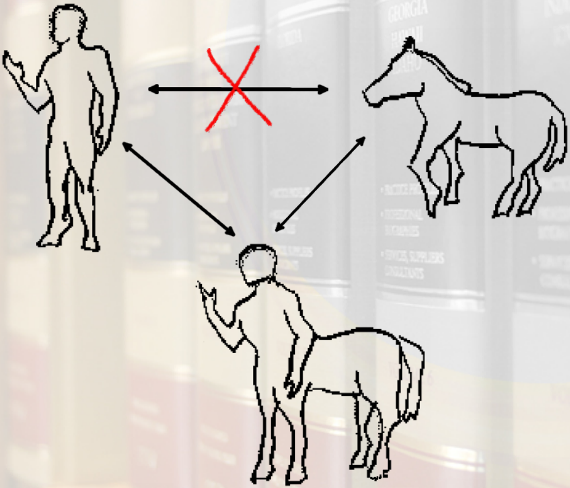
Name	$d(x,y)$	Symmetry	Triangle ineq.
Euclidean (L_2)	$\sqrt{\sum(x_i - y_i)^2}$	✓	✓
Cosine distance	$1 - \frac{x \cdot y}{\lvert x \rvert \lvert y \rvert}$	✓	✗
KL-diverg.	$\sum x_i \log \frac{x_i}{y_i}$	✗	✗
JS-diverg.	symmetrized & smoothed KL-diverg.	✓	✗

Distance functions can be **metric** or **non-metric**

Why Non-Metric Distances?

Source: Jacobs et al. (2000)

Why Non-Metric Distances?



Source: Jacobs et al. (2000)

More Non-Metric Examples

- Kullback-Leibler divergence:

$$\text{KL-div}(p, q) = \sum p_i \log \frac{p_i}{q_i}$$

- Many statistically learned similarity functions
- More examples: Jacobs et al. (2000)

Part II

Applications

Is Nearest-Neighbor Searching Useful?

Some Applications in ML and NLP

- Answering analogy questions

  — Mikolov et al. (2013b)

- Classification

  — Wan and Peng (2005); Kusner et al. (2015)

- Entity detection

  — Liu et al. (2011); Wang et al. (2009)

- Collaborative filtering

  — `slideshare.net/erikbern/music-recommendations-mlconf-2014`

- First story detection

  — Petrović et al. (2010)

- Data Imputation

  — Troyanskaya et al. (2001)

Analogy Questions

man is to **king** as **woman** is to **?**

- Substantial prior work: Turney (2012)
- **Human-level** performance achieved **10 years** ago (or earlier): Turney (2004)

Analogy Questions

man is to **king** as **woman** is to **queen**

- Substantial prior work: Turney (2012)
- **Human-level** performance achieved **10 years** ago (or earlier): Turney (2004)

Analogy Questions : word2vec

$$\textbf{man} - \textbf{king} \approx \textbf{woman} - \textbf{queen}$$

$$\textbf{king} - \textbf{man} \approx \textbf{queen} - \textbf{woman}$$

$$\textbf{queen} = \mathrm{argmax}_w \ \mathrm{similarity}(w, \textbf{king} - \textbf{man} + \textbf{woman})$$

— Cosine similarity is the best similarity function — Mikolov et al. (2013b)

Analogy Questions : word2vec

$$\textbf{man} - \textbf{king} \approx \textbf{woman} - \textbf{queen}$$

$$\textbf{king} - \textbf{man} \approx \textbf{queen} - \textbf{woman}$$

Is everything Ok here?

$$\textbf{queen} = \text{argmax}_w \ \text{similarity}(w, \textbf{king} - \textbf{man} + \textbf{woman})$$

— Cosine similarity is the best similarity function — Mikolov et al. (2013b)

Analogy Questions: You Need a Hack!

- **man** to **king** as **woman** to **?**
- **acorn** to **woods** as **apple** to **?**
- **pleasure** to **smile** as **pain** to **?**
- **France** to **Paris** as **Japan** to **?**

queen	**orchard**	**grimace**	**Tokyo**

Analogy Questions: You Need a Hack!

- **man** to **king** as **woman** to **?**
- **acorn** to **woods** as **apple** to **?**
- **pleasure** to **smile** as **pain** to **?**
- **France** to **Paris** as **Japan** to **?**

queen		orchard		grimace		Tokyo	
0.80	king	0.66	woods	0.63	smile	0.81	Tokyo

Analogy Questions: You Need a Hack!

- **man** to **king** as **woman** to **?**
- **acorn** to **woods** as **apple** to **?**
- **pleasure** to **smile** as **pain** to **?**
- **France** to **Paris** as **Japan** to **?**

queen		orchard		grimace		Tokyo	
0.80	king	0.66	woods	0.63	smile	0.81	Tokyo
0.71	queen	0.51	apple	0.61	pain	0.68	Japan

Analogy Questions: You Need a Hack!

- **man** to **king** as **woman** to **?**
- **acorn** to **woods** as **apple** to **?**
- **pleasure** to **smile** as **pain** to **?**
- **France** to **Paris** as **Japan** to **?**

queen		**orchard**		**grimace**		**Tokyo**	
0.80	king	0.66	woods	0.63	smile	0.81	Tokyo
0.71	queen	0.51	apple	0.61	pain	0.68	Japan
0.62	monarch	0.45	orchards	0.54	grin	0.66	Toyko

Analogy Questions: You Need a Hack!

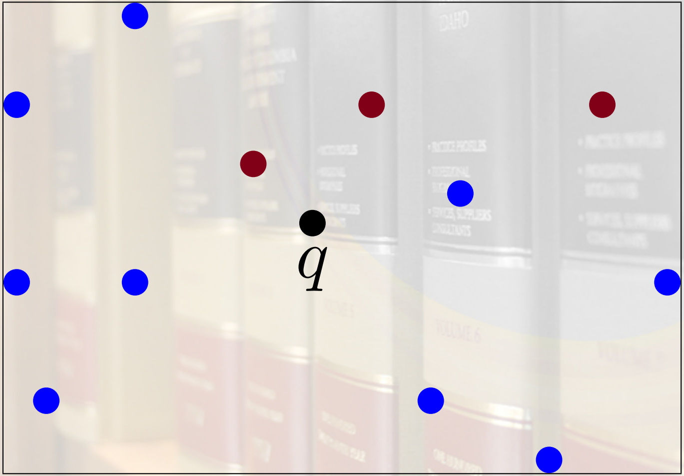
- **man** to **king** as **woman** to **?**
- **acorn** to **woods** as **apple** to **?**
- **pleasure** to **smile** as **pain** to **?**
- **France** to **Paris** as **Japan** to **?**

queen		**orchard**		**grimace**		**Tokyo**	
0.80	king	0.66	woods	0.63	smile	0.81	Tokyo
0.71	queen	0.51	apple	0.61	pain	0.68	Japan
0.62	monarch	0.45	orchards	0.54	grin	0.66	Toyko
0.59	princess	0.44	orchard	0.49	grimace	0.64	Osaka

— The hack was only briefly mentioned in another paper by Mikolov et al. (2013a)

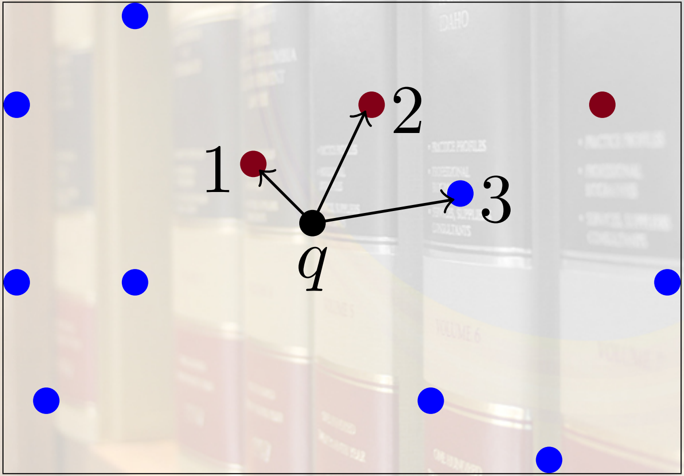
k-NN Classification

An example of 3-NN binary classification:



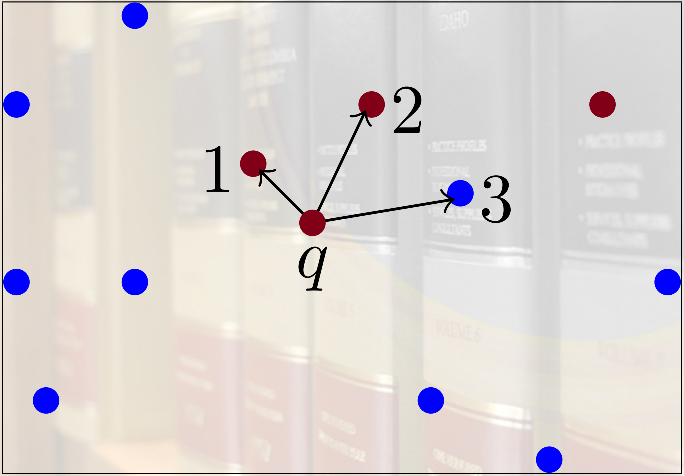
k-NN Classification

An example of 3-NN binary classification:



k-NN Classification

An example of 3-NN binary classification:



k-NN Classification

" Probably the main insight was that KNN is capable of making very good meta-features. Never underestimate nearest neighbours algorithm."

— Alexander Guschin, 2d place, Kaggle Otto Product Classification

k-NN Classification in NLP

One possible application to document classification:

- Compute pairwise similarities between document words using the "semantic" distance based either on WordNet: Wan and Peng (2005), or on word embeddings: Kusner et al. (2015)
- Aggregate pairwise similarities using either the Word Mover's Distance: Kusner et al. (2015), or the signature quadratic form distance: Beecks et al. (2010)

k-NN Classification in NLP

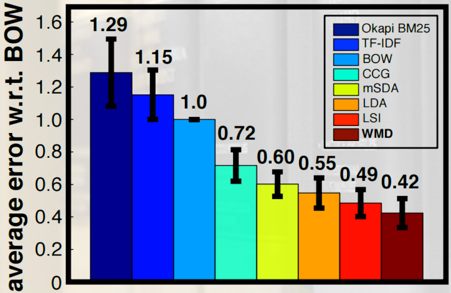


Figure 4. The kNN test errors of various document metrics averaged over all eight datasets, relative to kNN with BOW.

Source: Kusner et al. (2015)

Part III

State of the Art

Can ML Improve It?

How to find similar objects?

Two main options available:

- Brute-force (always **exact**)
- Indexing (can be exact or approximate)

How to find similar objects?

Are exact indexing methods sufficiently efficient in practice?

CURSE OF DIMENSIONALITY!

Source: http://liledekahlan.eklablog.com/tubes-paysages-mystiques-c176638

CURSE OF DIMENSIONALITY!

Source: http://liledekahlan.eklablog.com/tubes-paysages-mystiques-c176638

Curse of Dimensionality: Summary

- Exact indexing can be fast in low dimensions, but is **mostly slow** in a high-dimensional space
- **Approximate** search can be fast
- Approximate search may be **the only efficient** option in a non-metric space.

State-of-the-art approximate search methods (not exhaustive)

- Locality Sensitive Hashing (LSH)
- Proximity graphs (kNN-graphs)
- Permutation methods
- Hierarchical space partitioning (trees)
- Inverted files (usually used as an auxiliary data structure)

Locality Sensitive Hashing (LSH)

Locality Sensitive Hashing (LSH)

- Design a hash function $h(x)$ “sensitive” to locality

  — Proposed by Kushilevitz et al. (1998); Indyk and Motwani (1998)

Locality Sensitive Hashing (LSH)

- Design a hash function $h(x)$ "sensitive" to locality

  — Proposed by Kushilevitz et al. (1998); Indyk and Motwani (1998)

- The smaller is $d(x, y)$ the more likely $h(x) = h(y)$

Locality Sensitive Hashing (LSH)

- Design a hash function $h(x)$ “sensitive” to locality

  — Proposed by Kushilevitz et al. (1998); Indyk and Motwani (1998)
- The smaller is $d(x, y)$ the more likely $h(x) = h(y)$
- For reliable retrieval, use many hash functions

Locality Sensitive Hashing (LSH)

- Design a hash function $h(x)$ "sensitive" to locality

  — Proposed by Kushilevitz et al. (1998); Indyk and Motwani (1998)

- The smaller is $d(x, y)$ the more likely $h(x) = h(y)$

- For reliable retrieval, use many hash functions

- Works well for some L_p spaces & cosine similarity

Locality Sensitive Hashing (LSH)

- Design a hash function $h(x)$ "sensitive" to locality

  — Proposed by Kushilevitz et al. (1998); Indyk and Motwani (1998)

- The smaller is $d(x, y)$ the more likely $h(x) = h(y)$
- For reliable retrieval, use many hash functions
- Works well for some L_p spaces & cosine similarity
- Also, perhaps, for kernelized similarity functions

  — Kulis and Grauman (2009)

Locality Sensitive Hashing (LSH)

- Design a hash function $h(x)$ "sensitive" to locality

  — Proposed by Kushilevitz et al. (1998); Indyk and Motwani (1998)
- The smaller is $d(x, y)$ the more likely $h(x) = h(y)$
- For reliable retrieval, use many hash functions
- Works well for some L_p spaces & cosine similarity
- Also, perhaps, for kernelized similarity functions

  — Kulis and Grauman (2009)
- Much less is known about performance in a more general case — However, Athitsos et al. (2008)

Proximity/Neighborhood Graphs

- Ideas are quite old, but relatively unknown

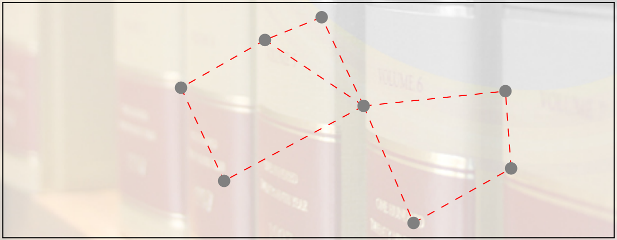
  — Arya and Mount (1993); Toussaint (1980)
- Link reasonably close points (not necessarily NNs)
- Use this graph during retrieval
- Several variants, we use the variant of Malkov et al. (2012)

Proximity/Neighborhood Graphs

- Ideas are quite old, but relatively unknown

  — Arya and Mount (1993); Toussaint (1980)
- Link reasonably close points (not necessarily NNs)
- Use this graph during retrieval
- Several variants, we use the variant of Malkov et al. (2012)

Each point is connected with **two** nearest neighbors:



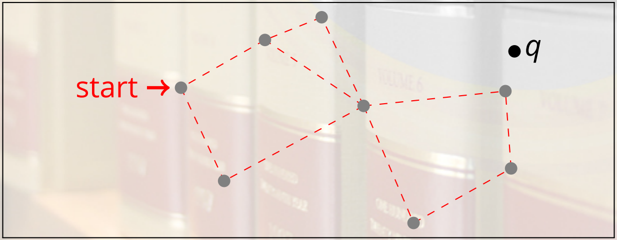
Proximity/Neighborhood Graphs

- Ideas are quite old, but relatively unknown

  — Arya and Mount (1993); Toussaint (1980)

- Link reasonably close points (not necessarily NNs)
- Use this graph during retrieval
- Several variants, we use the variant of Malkov et al. (2012)

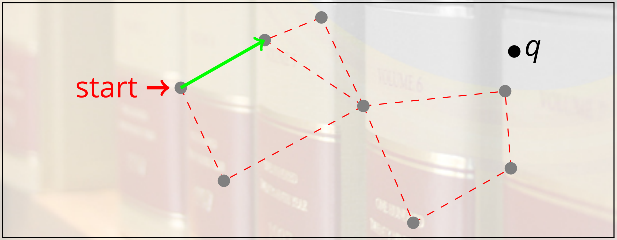
Each point is connected with **two** nearest neighbors:



Proximity/Neighborhood Graphs

- Ideas are quite old, but relatively unknown
  — Arya and Mount (1993); Toussaint (1980)
- Link reasonably close points (not necessarily NNs)
- Use this graph during retrieval
- Several variants, we use the variant of Malkov et al. (2012)

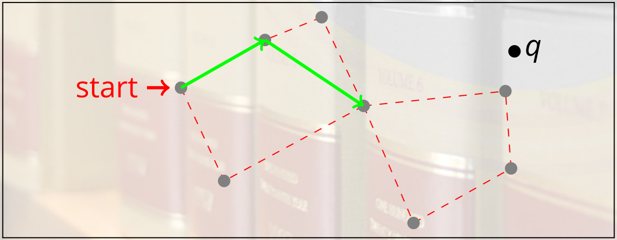
Each point is connected with **two** nearest neighbors:



Proximity/Neighborhood Graphs

- Ideas are quite old, but relatively unknown
  — Arya and Mount (1993); Toussaint (1980)
- Link reasonably close points (not necessarily NNs)
- Use this graph during retrieval
- Several variants, we use the variant of Malkov et al. (2012)

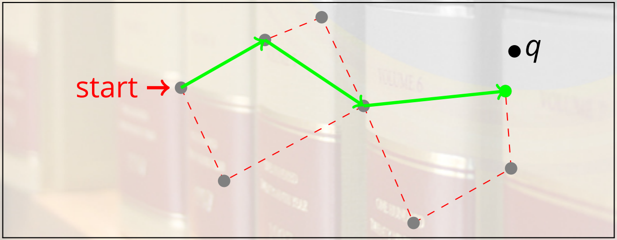
Each point is connected with **two** nearest neighbors:



Proximity/Neighborhood Graphs

- Ideas are quite old, but relatively unknown
  — Arya and Mount (1993); Toussaint (1980)
- Link reasonably close points (not necessarily NNs)
- Use this graph during retrieval
- Several variants, we use the variant of Malkov et al. (2012)

Each point is connected with **two** nearest neighbors:



Permutation Methods

Filter-and-refine using **pivot-based projection** to the Euclidean space (L_2):

- Select pivots randomly
- Rank pivots by their distances to data points
- Filter by comparing **pivot rankings**
- Refine by comparing remaining points to the query

Hierarchical space partitioning (VP-tree aka Ball-tree)

- A binary space-partitioning tree

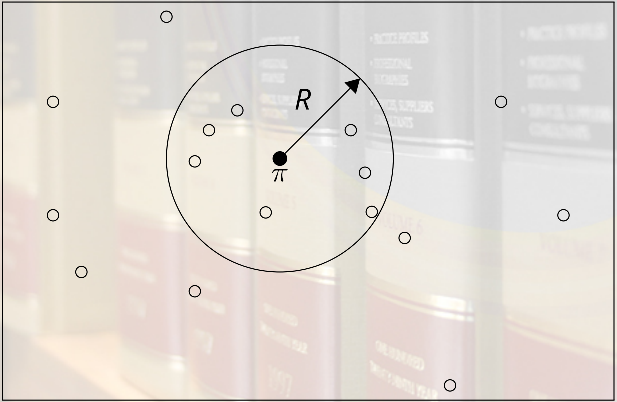
  — Proposed independently by Uhlmann (1991) and Yianilos (1993)

- A **metric-space** generalization of KD-tree
- Uses the **triangle inequality** to prune unpromising partitions

VP-tree: One Tree Node

Creating one index tree node:

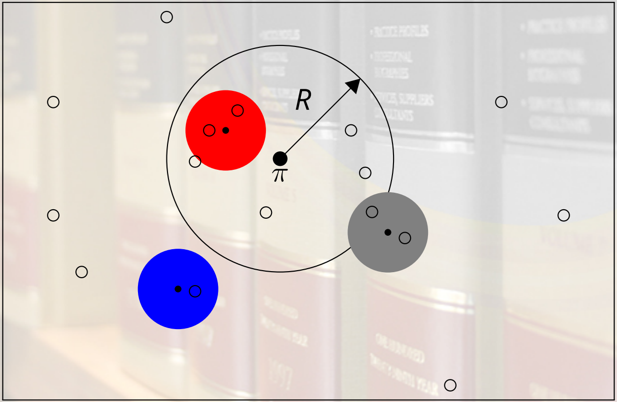
- A (random) pivot π is selected
- The space is divided by a sphere into two halves
- The radius of the sphere is a median distance to π.



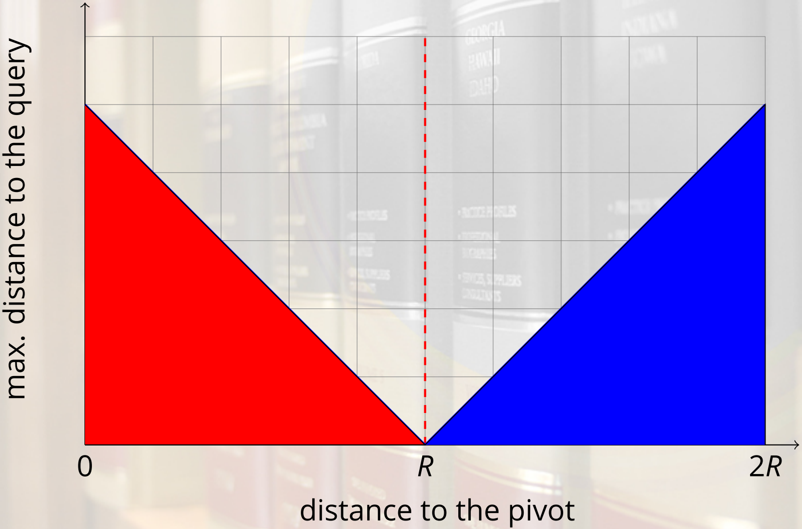
VP-tree: Three Types of Query Balls

The triangle inequality makes pruning possible:

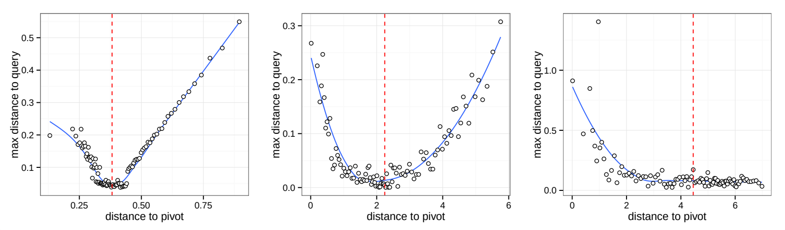
- Red query ball: prune the **outer** partition
- Blue query ball: prune the **inner** partition
- Gray query ball: **cannot** prune, visit both



VP-tree: Pruning Rule



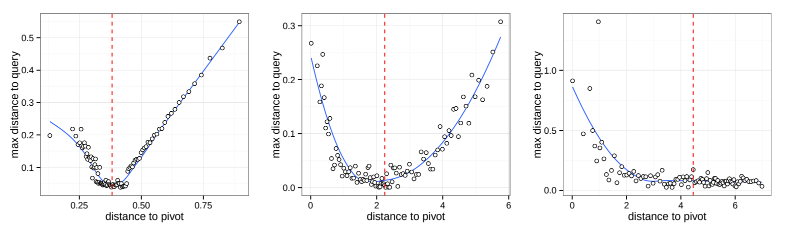
VP-tree: Pruning Rule Learned By Sampling



Colors, L_2 | RCV-8, KL-div | RCV-16, KL-div

The pruning function obtained by **sampling**. The red dashed line denotes a median distance R from data set points to the pivot π.

VP-tree: Pruning Rule Learned By Sampling



Colors, L_2 | RCV-8, KL-div | RCV-16, KL-div

The pruning function obtained by **sampling**. The red dashed line denotes a median distance R from data set points to the pivot π.

What if we learn a **parametric** piecewise-linear function instead?

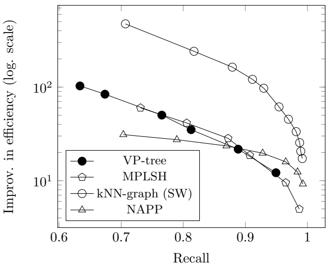
VP-tree: Simple Piecewise Linear Pruner

- Piecewise linear function has two parameters
- **Directly** optimize efficiency at a given recall

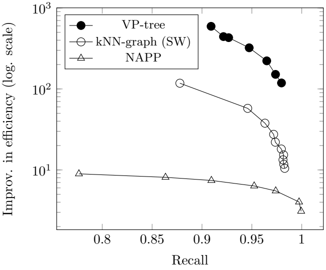
VP-tree: Simple Piecewise Linear Pruner

- Piecewise linear function has two parameters
- **Directly** optimize efficiency at a given recall

Efficiency vs. recall (10-NN search) : **higher and to the right** is better (VLDB’15 results):



SIFT, L_2



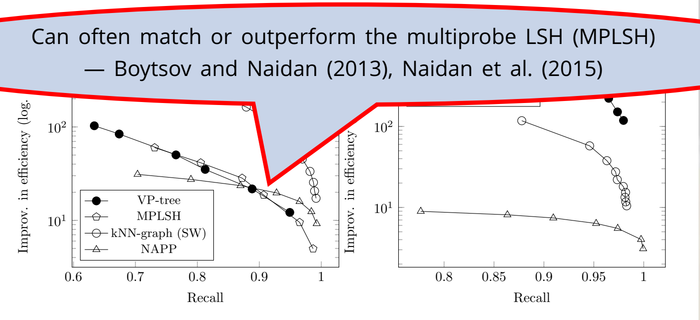
Wiki-8, KL-div

VP-tree: Simple Piecewise Linear Pruner

- Piecewise linear function has two parameters
- **Directly** optimize efficiency at a given recall

Efficiency vs. recall (10-NN search) : **higher and to the right** is better (VLDB’15 results):

Can often match or outperform the multiprobe LSH (MPLSH)
— Boytsov and Naidan (2013), Naidan et al. (2015)



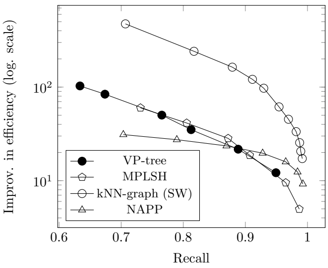
SIFT, L_2

Wiki-8, KL-div

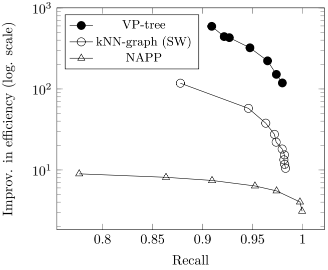
VP-tree: Simple Piecewise Linear Pruner

- Piecewise linear function has two parameters
- **Directly** optimize efficiency at a given recall

Efficiency vs. recall (10-NN search) : **higher and to the right** is better (VLDB'15 results):



SIFT, L_2



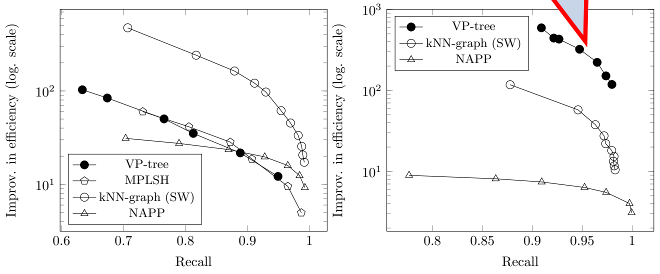
Wiki-8, KL-div

VP-tree: Simple Piecewise Linear Pruner

- Piecewise linear function has two parameters
- **Directly** optimize efficiency at a given recall

Efficiency vs. recall (10-NN search) : bi
better (VLDB'15 results):

VP-tree is better than the proximity graph



SIFT, L_2 Wiki-8, KL-div

More Examples of ML for Nearest-Neighbor Search

- Data-optimality: tune parameters to your data set

  — Dong et al. (2008); Cayton and Dasgupta (2007)

- Learn a distance function

  — Xing et al. (2002); Prekopcsák and Lemire (2012)

- Learn a monotonic transformation of the distance function

  — Skopal and Bartoš (2012)

Part IV

Non-Metric Space Library (NMSLIB)

- More state-of-the-art comparisons
- Using NMSLIB in other applications
- Next Steps

NMSLIB Overview: What is Non-Metric Space Library?

NMSLIB Overview: What is Non-Metric Space Library?

- NMSLIB is a collection of search methods for generic spaces

NMSLIB Overview: What is Non-Metric Space Library?

- NMSLIB is a collection of search methods for generic spaces
- NMSLIB has both metric and non-metric search algorithms

NMSLIB Overview: What is Non-Metric Space Library?

- NMSLIB is a collection of search methods for generic spaces
- NMSLIB has both metric and non-metric search algorithms
- NMSLIB has both exact and approximate search algorithms

NMSLIB Overview: What is Non-Metric Space Library?

- NMSLIB is a collection of search methods for generic spaces
- NMSLIB has both metric and non-metric search algorithms
- NMSLIB has both exact and approximate search algorithms
- The focus, however, is on approximate methods

NMSLIB Overview: What is Non-Metric Space Library?

- NMSLIB is a collection of search methods for generic spaces
- NMSLIB has both metric and non-metric search algorithms
- NMSLIB has both exact and approximate search algorithms
- The focus, however, is on approximate methods
- NMSLIB is an evaluation toolkit that simplifies experimentation and processing of results

NMSLIB Overview: What is Non-Metric Space Library?

- NMSLIB is a collection of search methods for generic spaces
- NMSLIB has both metric and non-metric search algorithms
- NMSLIB has both exact and approximate search algorithms
- The focus, however, is on approximate methods
- NMSLIB is an evaluation toolkit that simplifies experimentation and processing of results
- NMSLIB is extensible (new spaces and methods can be added)

NMSLIB Overview: What is Non-Metric Space Library?

- NMSLIB is a collection of search methods for generic spaces
- NMSLIB has both metric and non-metric search algorithms
- NMSLIB has both exact and approximate search algorithms
- The focus, however, is on approximate methods
- NMSLIB is an evaluation toolkit that simplifies experimentation and processing of results
- NMSLIB is extensible (new spaces and methods can be added)
- It was designed to be efficient: we provide efficient implementations of major distance functions

NMSLIB Overview: What is Non-Metric Space Library?

- NMSLIB is a collection of search methods for generic spaces
- NMSLIB has both metric and non-metric search algorithms
- NMSLIB has both exact and approximate search algorithms
- The focus, however, is on approximate methods
- NMSLIB is an evaluation toolkit that simplifies experimentation and processing of results
- NMSLIB is extensible (new spaces and methods can be added)
- It was designed to be efficient: we provide efficient implementations of major distance functions
- NMSLIB was designed as an experimental framework, but we work towards making it useful for a broader user base

NMSLIB Overview: Design Considerations

NMSLIB Overview: Design Considerations

- **Efficiency**
  - Implemented in C++
  - Vectorized (SIMD) distances (major)
  - Memory optimized layouts for trees

NMSLIB Overview: Design Considerations

- **Efficiency**
  - Implemented in C++
  - Vectorized (SIMD) distances (major)
  - Memory optimized layouts for trees
- **Reasonable portability & interoperability**
  - Use C++11, the code works on Linux & Win64
  - We have an experimental version works as a service (client can be Java, C++, Python, . . .)
  - We have experimental Python bindings

NMSLIB Overview: Design Considerations

- **Efficiency**
  - Implemented in C++
  - Vectorized (SIMD) distances (major)
  - Memory optimized layouts for trees
- **Reasonable portability & interoperability**
  - Use C++11, the code works on Linux & Win64
  - We have an experimental version works as a service (client can be Java, C++, Python, . . .)
  - We have experimental Python bindings
- **Some documentation**
  - Quick start guide
  - Detailed 60-page manual

NMSLIB Overview: Core Methods

- NMSLIB includes four core methods:
  - VP-tree
  - SW-graph (proximity graph)
  - NAPP (Neighborhood APProximation index)
  - Brute-force filtering of permutations

NMSLIB Overview: Core Methods

- NMSLIB includes four core methods:
  - VP-tree
  - SW-graph (proximity graph)
  - NAPP (Neighborhood APProximation index)
  - Brute-force filtering of permutations

- In **our** evaluations:
  - There was no single best core method
  - Some of the core methods outperformed other approaches
  - All core methods were reasonably effective for the **non-symmetric** and **non-metric** distance

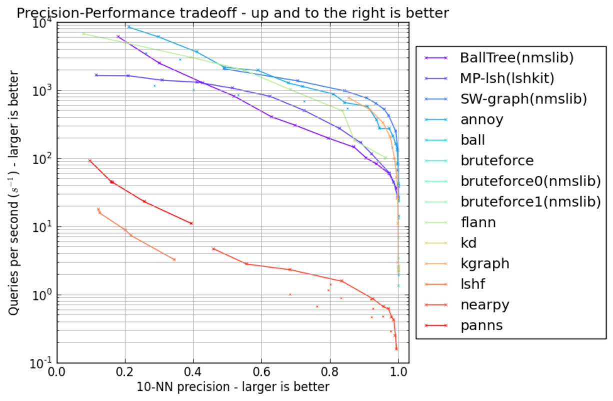
— Boytsov and Naidan (2013); Ponomarenko et al. (2014); Naidan et al. (2015)

More State-of-the-Art Comparisons: Public Benchmarks

Evaluation by Erik Bernhardsson

`https://github.com/erikbern/ann-benchmarks`

One million of SIFT vectors, L_2:

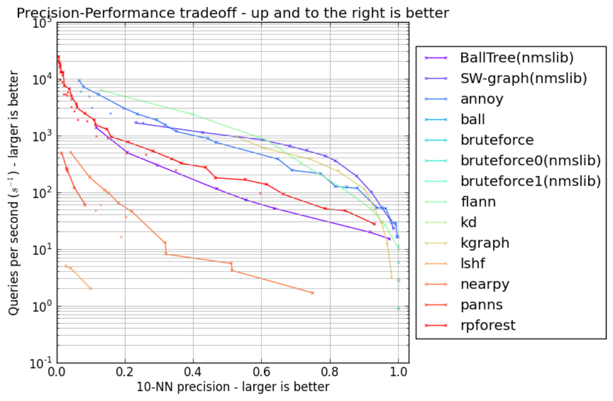


More State-of-the-Art Comparisons: Public Benchmarks

Evaluation by Erik Bernhardsson

https://github.com/erikbern/ann-benchmarks

1.19 million of Glove 100d embeddings, cosine:



Next Steps

Next Steps

- **Practical**
  - Index serialization for core methods is still work in progress
  - We have a version that works as a service, but it is not propagated to the main branch yet
  - No automatic parameter tuning for proximity graphs and permutation methods

Next Steps

- **Practical**
  - Index serialization for core methods is still work in progress
  - We have a version that works as a service, but it is not propagated to the main branch yet
  - No automatic parameter tuning for proximity graphs and permutation methods
- **Experimental/Scientific**
  - Implement and test a variety of proximity graphs
  - Compare proximity graphs against recent LSH indices (which we did not adopt yet)
  - Experiment with more challenging spaces

Talk Recap

- Nearest Neighbor search can be useful in ML and NLP
- Non-metric spaces are important
- Our NMSLIB library has decent support for such spaces
- NMSLIB includes SW-graph, which is quite efficient
- That said, NMSLIB is still work in progress
- LSH may not always be the best search method

Thank you for attention!

Our code is on GitHub:

`https://github.com/searchivarius/NonMetricSpaceLib`

Bibliography I

S. Arya and D. M. Mount. Approximate nearest neighbor queries in fixed dimensions. In *SODA*, volume 93, pages 271–280, 1993.

V. Athitsos, M. Potamias, P. Papapetrou, and G. Kollios. Nearest neighbor retrieval using distance-based hashing. In *ICDE'08*, pages 327–336. IEEE, 2008.

C. Beecks, M. S. Uysal, and T. Seidl. Signature quadratic form distance. In *CIVR'10*, pages 438–445, 2010.

L. Boytsov and B. Naidan. Learning to prune in metric and non-metric spaces. In *NIPS'13*, pages 1574–1582, 2013.

L. Cayton and S. Dasgupta. A learning framework for nearest neighbor search. *Advances in Neural Information Processing Systems*, 20, 2007.

W. Dong, Z. Wang, W. Josephson, M. Charikar, and K. Li. Modeling LSH for performance tuning. In *CIKM'08*, CIKM '08, pages 669–678, 2008.

P. Indyk and R. Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

D. Jacobs, D. Weinshall, and Y. Gdalyahu. Classification with nonmetric distances: Image retrieval and class representation. *Pattern Analysis and Machine Intelligence*, 22(6):583–600, 2000.

B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *International Conference on Computer Vision*, pages 2130–2137. IEEE, 2009.

Bibliography II

E. Kushilevitz, R. Ostrovsky, and Y. Rabani. Efficient search for approximate nearest neighbor in high dimensional spaces. In *Proceedings of the 30th annual ACM symposium on Theory of computing*, STOC '98, 1998.

M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger. From word embeddings to document distances. In *WUSTL*, pages 957–966, 2015.

X. Liu, S. Zhang, F. Wei, and M. Zhou. Recognizing named entities in tweets. In *NAACL HLT'11*, pages 359–367, 2011.

Y. Malkov, A. Ponomarenko, A. Logvinov, and V. Krylov. Scalable distributed algorithm for approximate nearest neighbor search problem in high dimensional general metric spaces. In *SISAP'12*, volume 7404, pages 132–147. 2012. ISBN 978-3-642-32152-8.

T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013a.

T. Mikolov, W.-t. Yih, and G. Zweig. Linguistic regularities in continuous space word representations. In *HLT-NAACL*, pages 746–751, 2013b.

B. Naidan, L. Boytsov, and E. Nyberg. Permutation search methods are efficient, yet faster search is possible. *PVLDB*, 8(12):1618–1629, 2015.

S. Petrović, M. Osborne, and V. Lavrenko. Streaming first story detection with application to twitter. In *NAACL'10*, pages 181–189. Association for Computational Linguistics, 2010.

Bibliography III

A. Ponomarenko, N. Avrelin, B. Naidan, and L. Boytsov. Comparative analysis of data structures for approximate nearest neighbor search. In *DATA ANALYTICS 2014, The Third International Conference on Data Analytics*, pages 125–130, 2014.

Z. Prekopcsák and D. Lemire. Time series classification by class-specific mahalanobis distance measures. *Advances in Data Analysis and Classification*, 6: 185–200, 2012. ISSN 1862-5347.

T. Skopal and T. Bartoš. Algorithmic exploration of axiom spaces for efficient similarity search at large scale. In *SISAP'12*, volume 7404, pages 40–53. 2012.

G. T. Toussaint. The relative neighbourhood graph of a finite planar set. *Pattern recognition*, 12(4):261–268, 1980.

O. Troyanskaya, M. Cantor, G. Sherlock, P. Brown, T. Hastie, R. Tibshirani, D. Botstein, and R. B. Altman. Missing value estimation methods for DNA microarrays. *Bioinformatics*, 17(6):520–525, 2001.

P. Turney. Human-level performance on word analogy questions by latent relational analysis. 2004.

P. D. Turney. Domain and function: A dual-space model of semantic relations and compositions. *Journal of Artificial Intelligence Research*, pages 533–585, 2012.

J. Uhlmann. Satisfying general proximity similarity queries with metric trees. *Information Processing Letters*, 40:175–179, 1991.

X. Wan and Y. Peng. The earth mover's distance as a semantic measure for document similarity. In *CIKM'05*, pages 301–302, 2005.

Bibliography IV

W. Wang, C. Xiao, X. Lin, and C. Zhang. Efficient approximate entity extraction with edit distance constraints. In *SIGMOD Conference on Management of data*, pages 759–770. ACM, 2009.

E. Xing, A. Ng, M. Jordan, and S. Russell. Distance metric learning, with application to clustering with side-information. *Advances in neural information processing systems*, 15:505–512, 2002.

P. N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *SODA*, volume 93, pages 311–321, 1993.