Dimension Reduction with Certainty

Rasmus Pagh
IT University of Copenhagen

ECML-PKDD
September 21, 2016

Slides: goo.gl/hZoRWo
What can we say about high-dimensional objects from a low-dimensional representation?
What can we say with certainty about high-dimensional objects from a low-dimensional representation?
Outline

Part I:
Tools for randomized dimension reduction - greatest hits

Part II:
Transparency and interpretability

Part III:
Dimension reduction with certainty?
Dimension reduction

Technique for mapping objects from a large space into a small space, while preserving essential relations.

Theorem 1

which any embedding... decades, its optimality has still not been settled. In the original pap... clustering... numerical linear algebra... compressed sensing... Such schemes have been applied to good e... short of the JL lemma for any...
Oblivious dimension reduction

Technique for mapping objects from a large space into a small space, while preserving essential relations, that is data-independent and does not need to be trained.
Oblivious dimension reduction

Technique for mapping objects from a large space into a small space, while preserving essential relations, that is data-independent and does not need to be trained.

Generally applicable.

As good as data-dependent methods in many cases.
Oblivious dimension reduction

Technique for mapping objects from a large space into a small space, while preserving essential relations, that is data-independent and does not need to be trained.

- Generally applicable.
- As good as data-dependent methods in many cases.
- Easier to parallelize.
- Data does not need to be available in advance - works for “on-line” data.
Next: Three tools

🛠 Random projection
🛠 Random feature mapping
🛠 1-bit minwise hashing
Random projections

Figure courtesy of Suresh Venkatasubramanian

Photo by Giovanni Dall’Orto
Johnson-Lindenstrauss Transformation

• To preserve $n$ Euclidean distances?

$$||\hat{x}_i - \hat{x}_j||_2 = (1 \pm \varepsilon)||x_i - x_j||_2$$
Johnson-Lindenstrauss Transformation

- To preserve $n$ Euclidean distances?

$$||\hat{x}_i - \hat{x}_j||_2 = (1 \pm \varepsilon)||x_i - x_j||_2$$

- Use a random* linear mapping!

$$x = \hat{x}$$ $m = O(\log(n) / \varepsilon^2)$

dimensions

[Johnson & Lindenstrauss ‘84]
Johnson-Lindenstrauss Transformation

• To preserve $n$ Euclidean distances?

$$||\hat{x}_i - \hat{x}_j||_2 = (1 \pm \varepsilon)||x_i - x_j||_2$$

• Use a random* linear mapping!

$$A \times x = \hat{x}$$

$m = O(\log(n)/\varepsilon^2)$
dimensions

[Johnson & Lindenstrauss '84]

Dot products?

Yes, but error depends on vector lengths

*random
Johnson-Lindenstrauss Transformation

• To preserve $n$ Euclidean distances?
  \[ \| \hat{x}_i - \hat{x}_j \|_2 = (1 \pm \varepsilon) \| x_i - x_j \|_2 \]

• Use a random* linear mapping!

\[ m = O(\log(n)/\varepsilon^2) \]

Still too many dimensions!

[Johnson & Lindenstrauss ‘84]
Johnson-Lindenstrauss Transformation

• To preserve \( n \) Euclidean distances?

\[ \| \hat{x}_i - \hat{x}_j \|_2 = (1 \pm \varepsilon) \| x_i - x_j \|_2 \]

• Use a random* linear mapping!

\[ x = A \hat{x} \]

\[ m = O(\log(n) / \varepsilon^2) \]

Dimensions

[Johnson & Lindenstrauss ‘84]

Optimality of the Johnson-Lindenstrauss Lemma

Kasper Green Larsen*  Jelani Nelson†

September 8, 2016

Dot products?

Yes, but error depends on vector lengths

Still too many dimensions!
Oblivious subspace embeddings

- Do better if data has nice structure? For example constrained to $d$-dim. subspace.
Oblivious subspace embeddings

• Do better if data has nice structure?
  For example constrained to $d$-dim. subspace.
  - Principal component analysis (PCA) works,
    but mapping is data-dependent.
Oblivious subspace embeddings

• Do better if data has nice structure? For example constrained to $d$-dim. subspace.
  
  - Principal component analysis (PCA) works, but mapping is data-dependent.
  
  - A suitable random linear map works with $m = O(d / \varepsilon^2)$ dimensions! [Sarlós ‘06]
Oblivious subspace embeddings

• Do better if data has nice structure? For example constrained to $d$-dim. subspace.
  
  - Principal component analysis (PCA) works, but mapping is *data-dependent*.
  
  - A suitable random linear map works with $m = O(d / \varepsilon^2)$ dimensions! \[\text{[Sarlós '06]}\]
  
  - Sparse matrices almost as good \[\text{[Cohen '16]}\].

Key tool in randomized numerical linear algebra (RandNLA) methods
Randomization offers new benefits for large-scale linear algebra computations.

BY PETROS DRINEAS AND MICHAEL W. MAHONEY

RandNLA: Randomized Numerical Linear Algebra

Matrices are ubiquitous in computer science, statistics, and applied mathematics. An \( m \times n \) matrix can encode information about \( m \) objects (each described by \( n \) features), or the behavior of a discretized differential operator on a finite element mesh; an \( n \times n \) positive-definite matrix can encode the correlations between all pairs of \( n \) objects, or the edge-connectivity between all pairs of nodes in a social network; and so on. Motivated largely by technological developments that generate extremely large scientific and Internet datasets, recent years have witnessed exciting developments in the theory and practice of matrix algorithms. Particularly remarkable is the use of randomization—typically assumed to be a property of the input data due to, for example, noise in the data generation mechanisms—as an algorithmic or computational resource for the development of improved algorithms for fundamental matrix problems such as matrix multiplication, least-squares (LS) approximation, low-rank matrix approximation, and Laplacian-based linear equation solvers.

Randomized Numerical Linear Algebra (RandNLA) is an interdisciplinary research area that exploits randomization as a computational resource to develop improved algorithms for large-scale linear algebra problems.\(^3\) From a foundational perspective, RandNLA has its roots in theoretical computer science (TCS), with deep connections to mathematics (convex analysis, probability theory, metric embedding theory) and applied mathematics (scientific computing, signal processing, numerical linear algebra). From an applied perspective, RandNLA is a vital new tool for machine learning, statistics, and data analysis. Well-engineered implementations have already outperformed highly optimized software libraries for ubiquitous problems such as least-squares,\(^4,5\) with good scalability in parallel and distributed environments.\(^5\) Moreover, RandNLA promises a sound algorithmic and statistical foundation for modern large-scale data analysis.

key insights

- Randomization isn’t just used to model noise in data; it can be a powerful computational resource to develop algorithms with improved running times and stability properties as well as algorithms that are more interpretable in downstream data science applications.
- To achieve best results, random sampling of elements or columns/rows must be done carefully, but random projections can be used to transform or rotate the input data to a random basis where simple uniform random sampling of elements or rows/columns can be successfully applied.
- Random sketches can be used directly to get low-precision solutions to data science applications; or they can be used indirectly to construct preconditioners for traditional iterative numerical algorithms to get high-precision solutions in scientific computing applications.
Random feature mappings

\[ \begin{align*}
\text{data vectors} & \quad \text{kernel expansion} \quad \text{feature space} \\
00101110111010101 & \quad 0010000001010000 \ldots 00001000100000010 \\
\hat{x} & \quad \hat{x} \cdot \hat{y} \approx \phi(x) \cdot \phi(y)
\end{align*} \]

[Rahimi & Recht ‘07]
Random feature mappings

Can use linear kernel methods (e.g. linear SVM)

\[ \hat{x} \cdot \hat{y} \approx \varphi(x) \cdot \varphi(y) \]
Random feature mappings

Efficient mappings:
• Random kitchen sinks (2007)

Can use linear kernel methods (e.g. linear SVM)

\[ \hat{x} \cdot \hat{y} \approx \varphi(x) \cdot \varphi(y) \]
Random feature mappings

Efficient mappings:
• Random kitchen sinks (2007)
• FastFood (2013)
• Tensor Sketching (2013)

Can use linear kernel methods (e.g. linear SVM)
Sparse vectors
Sparse vectors

Term vector (or TF/IDF)

0010000001010000 … 00001000100000010

> 10^5 dimensions

001011101110101010010010

dimension-reduced representation
Sparse vectors

Sparse, non-negative entries

Term vector (or TF/IDF)

0010000001010000 ... 00001000100000010

> 10^5 dimensions

001011101110101010101001010

dimension-reduced representation
Optimality of the Johnson-Lindenstrauss Lemma
Kasper Green Larsen
Mads Nielsen
September 8, 2016

Abstract
For any integers n, d, ε > 0, there exists a map f : R^n → R^d satisfying
\( \|x - y\| \leq (1 + \epsilon) \|x - y\| \)
for all x, y ∈ R^n. This was later improved by Johnson and Lindenstrauss [JL84],
who showed the existence of an embedding into dimension min
\( \Omega \left( \frac{n}{\epsilon^2} \right) \)
for some constant c > 0.

1 Introduction
In many algorithm design, when data is high-dimensional, and one seeks to first pre-process the data
so that the distance between in the Euclidean space can be preserved in such a way that is acceptable for
later algorithms, one typically has to sacrifice time to be analyzed by later algorithms. The Johnson
Lindenstrauss (JL) lemma [JL84] provides (1) must have
\( \|x - y\| \leq (1 + \epsilon) \|x - y\| \)
for some constant c > 0. This bound falls
\( \frac{1}{2a n} \frac{\epsilon}{n, d} \Omega \left( \frac{n}{\epsilon^2} \right) \)
for any positive integer a.

1.1 Term vector (or TF/IDF)

- Min-wise hashing (1997)
- b-bit min-wise hashing (2010)

Sparse, non-negative entries

0010000001010000 ... 00001000100000010

> 10^5 dimensions

0010111011101010101010010101

dimension-reduced representation

Sparse vectors
1-bit minwise hashing

• Min-max kernel: \( k(x, y) = \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)} \)
1-bit minwise hashing

- Min-max kernel: \( k(x, y) = \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)} \)

Now: Binary vectors/Jaccard similarity
1-bit minwise hashing

- Min-max kernel: \( k(x, y) = \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)} \)

Now: Binary vectors/Jaccard similarity

- Random hash functions \( h_i: \mathbb{N} \rightarrow [0;1] \)
  
  - Min-hash: \( z_i(x) = \arg \min_{x_j \neq 0} h_i(j) \)
1-bit minwise hashing

• Min-max kernel: \( k(x, y) = \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)} \)

Now: Binary vectors/Jaccard similarity

• Random hash functions \( h_i: \mathbb{N} \rightarrow [0;1] \)
  - Min-hash: \( z_i(x) = \arg \min_{x_j \neq 0} h_i(j) \)
  - 1-bit min-hash: \( b_i(z_i(x)), \) for random \( b_i: \mathbb{N} \rightarrow \{0,1\} \)
1-bit minwise hashing

- Min-max kernel: \( k(x, y) = \frac{\sum_i \min(x_i, y_i)}{\sum_i \max(x_i, y_i)} \)

Now: Binary vectors/Jaccard similarity

- Random hash functions \( h_i: \mathbb{N} \rightarrow [0;1] \)
  - Min-hash: \( z_i(x) = \arg \min_{x_j \neq 0} h_i(j) \)
  - 1-bit min-hash: \( b_i(z_i(x)) \), for random \( b_i: \mathbb{N} \rightarrow \{0,1\} \)

- Binary dimension-reduced representation: 
  \( b_1(z_1(x)) \ldots b_m(z_m(x)) \)
Theory and Applications of $b$-Bit Minwise Hashing

By Ping Li and Arnd Christian König

Abstract

Efficient (approximate) computation of set similarity in very large datasets is a common task with many applications in information retrieval and data management. One common approach for this task is minwise hashing. This paper describes $b$-bit minwise hashing, which can provide an order of magnitude improvements in storage requirements and computational overhead over the original scheme in practice.

We give both theoretical characterizations of the performance of the new algorithm as well as a practical evaluation on large real-life datasets and show that these match very closely. Moreover, we provide a detailed comparison with other important alternative techniques proposed for estimating set similarities. Our technique yields a very simple algorithm and can be realized with only minor modifications to the original minwise hashing scheme.

1. INTRODUCTION

With the advent of the Internet, many applications are faced with very large and inherently high-dimensional datasets. A common task on these is similarity search, that is, given a high-dimensional data point, the retrieval of data points that are close under a given distance function. In many scenarios, the storage and computational requirements for computing exact distances between all data points are prohibitive, making data representations that allow compact storage and efficient approximate distance computation necessary.

In this paper, we describe $b$-bit minwise hashing, which leverages properties common to many application scenarios to obtain order-of-magnitude improvements in the storage space and computational overhead required for a given level of accuracy over existing techniques. Moreover, while the theoretical analysis of these gains is technically challenging, the resulting algorithm is simple and easy to implement.

To describe our approach, we first consider the concrete task of Web page duplicate detection, which is of critical importance in the context of Web search and was one of the motivations for the development of the original minwise hashing algorithm by Broder et al. $^4$ Here, the task is to identify pairs of pages that are textually very similar. For this purpose, Web pages are modeled as "a set of contiguous words occurring on the page. Now, given two such sets $S_i, S_j \subseteq \Omega$, the normalized similarity known as resemblance or jaccard similarity, denoted by $R$, is

$$R = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} = \frac{a}{f_i + f_j - a} \quad \text{where } f_i = |S_i|, f_j = |S_j|.$$

Duplicate detection now becomes the task of detecting pairs of pages for which $R$ exceeds a threshold value. Here, $w$ is a tuning parameter and was set to be $w=5$ in several studies. $^1,2,4$ Clearly, the total number of possible shingles is huge. Considering $10^5$ unique English words, the total number of possible 5-shingles should be $D = (10^5)^2 = 10^{10}$. A prior study used $D = 2^m$ and even earlier studies used $D = 2^n$. Due to the size of $D$ and the number of pages crawled as part of Web search, computing the exact similarities for all pairs of pages may require prohibitive storage and computational overhead, leading to approximate techniques based on more compact data structures.

1.1. Minwise hashing

To address this issue, Broder and his colleagues developed minwise hashing in their seminal work. $^1,4$ Here, we give a brief introduction to this algorithm. Suppose a random permutation $\pi$ is performed on $\Omega$, that is,

$$\pi : \Omega \rightarrow \Omega, \quad \text{where } \Omega = \{0,1,\ldots,D-1\}.$$

An elementary probability argument shows that

$$\Pr(\min(\pi(S_i) = \min(\pi(S_j)))) = \frac{|S_i \cap S_j|}{|S_i \cup S_j|} = R. \quad \text{(1)}$$

After $k$ minwise independent permutations, $\pi_1, \pi_2, \ldots, \pi_k$, one can estimate $R$ without bias, as a binomial probability:

$$\hat{R}_w = \frac{1}{k} \sum_{i=1}^k \min(\pi_i(S_i)) = \min(\pi_i(S_i)). \quad \text{(2)}$$

$$\text{Var}(\hat{R}_w) = \frac{1}{k} R(1-R). \quad \text{(3)}$$

We will frequently use the terms "sample" and "sample size" (i.e., $k$). For minwise hashing, a sample is a hashed value, $\min(\pi(S))$, which may require, for example, 64 bits. $^7$

Since the original minwise hashing work, $^1,4$ there have been considerable theoretical and methodological developments. $^4,5,10,11,12,13$ Applications: As a general technique for estimating set similarity, minwise hashing has been applied to a wide range of applications, for example, content matching for online advertising, $^2$ detection of redundancy in enterprise applications, and so on.

The previous version of this paper, entitled "b-Bit Minwise Hashing for Estimating Three-way Similarities," was published in Proceedings of the Neural Information Processing Systems: NIPS 2010 (Vancouver, British Columbia, Canada).
Part II: Transparency and interpretability
EU regulations on algorithmic decision-making and a “right to explanation”

Bryce Goodman
Oxford Internet Institute, Oxford

Seth Flaxman
Department of Statistics, Oxford

Abstract
We summarize the potential impact that the European Union’s new General Data Protection Regulation will have on the routine use of machine learning algorithms. Slated to take effect as law across the EU in 2018, it will restrict automated individual decision-making (that is, algorithms that make decisions based on user-level predictors) which “significantly affect” users. The law will also create a “right to explanation,” whereby a user can ask for an explanation of an algorithmic decision that was made about them. We argue that while this law will pose large challenges for industry, it highlights opportunities for machine learning researchers to take the lead in designing algorithms and evaluation frameworks which avoid discrimination.

1. Introduction
On 14 April 2016, for the first time in over two decades, the European Parliament adopted a set of comprehensive regulations for the collection, storage and use of personal information, the General Data Protection Regulation (see figure 1). The new regulation has been described as a “Copernican Revolution” in data protection law, “seeking to shift its focus away from paper-based, bureaucratic requirements and towards compliance in practice, harmonization of the law, and individual empowerment” (Kuner, 2012). Much of the regulations are clearly aimed at perceived gaps and inconsistencies in the EU’s current approach to data protection. This includes, for example, the codification of the “right to be forgotten” (Article 16), and regulations for foreign companies collecting data from European citizens (Article 35).

However, while the bulk of language deals with how data is collected and stored, the regulation contains a short article entitled “Automated individual decision-making” (see figure 1) potentially prohibiting a wide swath of algorithms currently in use in, e.g. recommendation systems, credit and insurance risk assessments, computational advertising, and social networks. This raises important issues that are of particular concern to the machine learning community. In its current form, the GDPR’s requirements could require a complete overhaul of standard and widely used algorithmic techniques. The GDPR’s policy on the right of citizens to receive an explanation for algorithmic decisions highlights the pressing importance of human interpretability in algorithm design. If, as expected, the GDPR takes effect in its current form in mid-2018, there will be a pressing need for effective algorithms which can operate within this new legal framework.

Figure 1. Excerpt from the General Data Protection Regulation, Council of the European Union (2016)
We summarize the potential impact that the European Union’s new General Data Protection Regulation will have on the routine use of machine learning algorithms. Slated to take effect as law across the EU in 2018, it will restrict automated individual decision-making (that is, algorithms that make decisions based on user-level predictors) which “significantly affect” users. The law will also create a “right to explanation,” whereby a user can ask for an explanation of an algorithmic decision that was made about them. We argue that while this law will pose large challenges for industry, it highlights opportunities for machine learning researchers to take the lead in designing algorithms and evaluation frameworks which avoid discrimination.
We summarize the potential impact that the European Union’s new General Data Protection Regulation will have on the routine use of machine learning algorithms. Slated to take effect as law across the EU in 2018, it will restrict automated individual decision-making (that is, algorithms that make decisions based on user-level predictors) which “significantly affect” users. The law will also create a “right to explanation,” whereby a user can ask for an explanation of an algorithmic decision that was made about them. We argue that while this law will pose large challenges for industry, it highlights opportunities for machine learning researchers to take the lead in designing algorithms and evaluation frameworks that avoid discrimination.
Biff Tannen in Back to the Future
Issues for dimension reduction

• Dimension reduction creates features that are not easy to describe in terms of original data.
Issues for dimension reduction

• Dimension reduction creates features that are not easy to describe in terms of original data.

• Use of randomization causes issues of:
  - Trust. “Is it a coincidence that my feature vector is similar to Donald Trump’s, or was this arranged?”
  - Fairness. “Would I have gotten a loan if the random choices had been different?”
  - …
Part III: Dimension reduction with certainty?
kNN classifier
kNN classifier
kNN classifier

“Explanation”
kNN classifier

“Explanation”
Getting *rid* of randomness?

- Easy to argue that a *deterministic* dimension reduction cannot work without assumptions on data (“incompressibility”).
Getting *rid* of randomness?

- Easy to argue that a *deterministic* dimension reduction cannot work without assumptions on data ("incompressibility").

- Second best option? Randomized algorithms whose output is guaranteed, but may fail to produce result within a given time/space usage. ("Las Vegas", a la quicksort.)
CoveringLSH: Locality-sensitive Hashing without False Negatives

RASMUS PAGH, IT University of Copenhagen

We consider a hash family \( H \) in the sense of Indyk and Motwani [17], to be a family of \( m \)-bit hash functions \( h : S \to \{0,1\}^m \) from a domain \( S \) to \( \{0,1\}^m \). The construction of a locality-sensitive hash family \( H \) is defined in terms of a function \( c(\cdot,\cdot) \) that is an integer called the locality-sensitive function. We consider a hash family \( H \) to be \( (r, \rho) \)-locality-sensitive if the locality-sensitive function \( c(\cdot,\cdot) \) of the new construction is \( c(r, \rho) = o(\log(n)) \), for an integer \( c \) of the new construction. One consequence is that nearly all of the collisions have little or no collisions.

Scalability and Total Recall with Fast CoveringLSH

Ninh Pham
IT University of Copenhagen
Denmark
ndap@itu.dk

Rasmus Pagh
IT University of Copenhagen
Denmark
pagh@itu.dk

ABSTRACT

Locality-sensitive hashing (LSH) has emerged as the dominant algorithmic technique for similarity search with strong performance guarantees in high-dimensional spaces. A drawback of traditional LSH schemes is that they can have false negatives, i.e., the recall is less than 100%. This limits the applicability of LSH in settings requiring precise performance guarantees. Building on the recent theoretical “CoveringLSH” construction that eliminates false negatives, we propose a fast and practical covering LSH scheme for Hamming space called Fast CoveringLSH (fcLSH). Inheriting the design benefits of CoveringLSH, our method avoids false negatives and always reports all near neighbors. Compared to CoveringLSH, we achieve an asymptotic improvement to the hash function computation time from \( O(dL) \) to \( O(d + L \log L) \), where \( d \) is the dimensionality of data and \( L \) is the number of hash tables. Our experiments on synthetic and real-world data sets demonstrate that fcLSH is competitive on data [7]. The emergence of big data adds to both research and commercial applications the challenges of scale and accuracy for efficient similarity search.

In most such applications, data can be represented or approximated as high-dimensional binary vectors, and Hamming distance is used as a similarity measure. For instance, a near-duplicate detection system uses hashing techniques [6, 17, 23] to represent documents as binary vectors, and identifies them as near-duplicates if their Hamming distances are smaller than a threshold radius. In content-based image retrieval systems, a standard approach is to learn short binary codes to represent image objects such that the Hamming distance between codes reflects their neighborhood or semantic similarity in the original space [16, 30, 36, 38]. Retrieving similar images can be efficiently done by simply returning all images with codes within a small Hamming distance of the code of the query image.

Similarity search in Hamming space dates back to Min-
Bloom filters

$h(S)$ ➟ $S$
Bloom filters

- **Use cases:**
  - Detecting identical data in a remote server.
  - Make it possible to test for inclusion in $S$ while revealing very little about $S$. 

$h(S)$ ➟ $S$
Bloom filters

- Use cases:
  - Detecting identical data in a remote server.
  - Make it possible to test for inclusion in $S$ while revealing very little about $S$.

Allow $\epsilon$ fraction “false positives”
Distance sensitive Bloom filters

- Use cases:
  - Detecting nearly identical vectors in a remote server.
  - Make it possible to test for proximity to a vector $x$ in $S$ while revealing very little about $S$. 

Allow $\epsilon$ fraction “false positives”
Distance sensitive Bloom filters

- Store collection of $S$ bit vectors
- Given query vector $y$ determine distinguish
  1. exists $x \in S$ within distance $r$ from $y$, and
  2. all vectors in $S$ have distance at least $cr$ from $y$

[Kirsch & Mitzenmacher ‘06]
Distance sensitive Bloom filters

- Store collection of $S$ bit vectors
- Given query vector $y$ determine distinguish
  1. exists $x \in S$ within distance $r$ from $y$, and
  2. all vectors in $S$ have distance at least $cr$ from $y$

[Kirsch & Mitzenmacher ‘06]
Distance sensitive Bloom filters

- Store collection of $S$ bit vectors

- Given query vector $y$ determine distinguish
  1. exists $x \in S$ within distance $r$ from $y$, and
  2. all vectors in $S$ have distance at least $cr$ from $y$
Distance sensitive Bloom filters

• Store collection of $S$ bit vectors

• Given query vector $y$ determine distinguish

1. exists $x \in S$ within distance $r$ from $y$, and

2. all vectors in $S$ have distance at least $cr$ from $y$

• No requirement outside of these cases; also, in case 2 we allow probability $\varepsilon$ of “false positive”

[Kirsch & Mitzenmacher ‘06]
Distance sensitive Bloom filters

- Store collection of $S$ bit vectors
- Given query vector $y$ determine distinguish
  1. exists $x \in S$ within distance $r$ from $y$, and
  2. all vectors in $S$ have distance at least $cr$ from $y$
- No requirement outside of these cases; also, in case 2 we allow probability $\varepsilon$ of “false positive”

But: Existing solutions also have false negatives…
Distance sensitive Bloom filters

without false negatives

• Consider the single-item case, $S=\{x\}$

• **Basic idea:** For random $a \in \{-1,+1\}^d$, store $a \cdot x$

  Query $y$: If $|a \cdot x - a \cdot y| \leq r$ answer ‘1’, otherwise ‘2’
Distance sensitive Bloom filters

without false negatives

• Consider the single-item case, $S = \{x\}$

• Basic idea: For random $a \in \{-1,+1\}^d$, store $a \cdot x$

Query $y$: If $|a \cdot x - a \cdot y| \leq r$ answer ‘1’, otherwise ‘2’
Distance sensitive Bloom filters

without false negatives

- Consider the single-item case, $S=\{x\}$

- **Basic idea:** For random $a \in \{-1,+1\}^d$, store $a \cdot x$

Query $y$: If $|a \cdot x - a \cdot y| \leq r$ answer ‘1’, otherwise ‘2’

Distinguishes distance $r$ and $r^2$ with constant probability and no false positives
Distance sensitive Bloom filters
without false negatives

• Consider the single-item case, \( S=\{x\} \)

• **Basic idea:** For random \( a \in \{-1,+1\}^d \), store \( a \cdot x \)

Query \( y \): If \( |a \cdot x - a \cdot y| \leq r \) answer ‘1’, otherwise ‘2’

In general, distinguish between distance \( r \) and \( cr \) using space \( \tilde{O}(r/(c-1)) \)
Distance sensitive Bloom filters

without false negatives

• Consider the single-item case, $S = \{x\}$

• Basic idea: For random $a \in \{-1,+1\}^d$, store $a \cdot x$

Query $y$: If $|a \cdot x - a \cdot y| \leq r$ answer ‘1’, otherwise ‘2’

In general, distinguish between distance $r$ and $cr$ using space $\tilde{O}(r/(c-1))$

Space usage essentially optimal

[ Goswami et al. ‘16 ]
Deterministic feature mappings for the polynomial kernel

data vectors

$00101110111010101$
$x$

feature space

$0010000001010000 \ldots 00001000100000010$
$\varphi(x)$

$\hat{x}$

dimension-reduced representation

$\hat{x} \cdot \hat{y} \approx (x \cdot y)^k$
Deterministic feature mappings for the polynomial kernel

\[ \varphi(x) \approx (x \cdot y)^k \]

[Karppa et al. '16]
Deterministic feature mappings for the polynomial kernel

How is this possible!?
How it works

• Considers the kernel \( k(x,y) = (x \cdot y)^2 \)

• Feature space: 1 feature per edge in biclique

[Karppa et al. '16]
How it works

• Considers the kernel $k(x,y) = (x \cdot y)^2$

• Feature space: 1 feature per edge in biclique edge in constant degree expander graph

[Karppa et al. ‘16]
How it works

• Considers the kernel \( k(x,y) = (x \cdot y)^2 \)

• Feature space:
  1 feature per edge in biclique
  edge in constant degree expander graph

[Proofs only for vectors in \([-1, +1]^d\)]

[Karppa et al. ‘16]
Some open questions

What machine learning or knowledge discovery aided decisions can be made “explainable”? 
Some open questions

- What machine learning or knowledge discovery aided decisions can be made “explainable”?
- What ML/KDD algorithms can be sped up by RandNLA methods?
Some open questions

What machine learning or knowledge discovery aided decisions can be made “explainable”?

What distance/kernel approximations are possible with one-sided error?

What ML/KDD algorithms can be sped up by RandNLA methods?
Some open questions

What machine learning or knowledge discovery aided decisions can be made “explainable”?

What ML/KDD algorithms can be sped up by RandNLA methods?

What kernels expansions have efficient *deterministic* approximations?

What distance/kernel approximations are possible with *one-sided* error?
Thank you for your attention!

Acknowledgement of economic support:

PS. Seeking a post-doc to start in 2017!