On the Power of Randomization in Big Data Analytics

Ninh Pham

Advisor: Prof. Rasmus Pagh
Theoretical Computer Science Section
IT University of Copenhagen, Denmark

ITU, Oct 7, 2014
Outline

• Challenges of Big Data Analytics

• Randomization

• Thesis Contribution: Happy Marriages between Randomized Algorithms and Big Data Analytics

• Research Direction
What is Big Data?

The Moore's Law of Big Data: “We have generated more than 90% of data in the last two years itself.”
Data Analytics

• Bridge the gap between data and information

• Fundamental operations:
  • Finding elements that meet a specified criterion
  • Modeling data for useful information discovery
Challenges of Big Data Analytics

- Data from different sources in different forms
- High speed of data flow, data change and data processing
- Data volume, query volume
Randomization

- Randomization is the process of making something random.
Why Randomize?

LIFE IS RANDOM

“GOD DOES NOT PLAY DICE.”

ALBERT EINSTEIN

© Lifeshack Quotes
Why Randomize?

- Randomization has played a power role in computer science both in foundational areas (complexity theory, algorithms) and in applied areas (data mining, network).

- Randomization often makes algorithms simpler, faster, easier to program, and parallelizable.

- In many applications, randomized algorithms often achieve the “best” 4-dimensional tradeoff of time, space, correctness (error probability), and programmer time.
Principle in Designing Randomized Algorithms for Big Data Analytics

Data represented as high-dimensional vectors in the Euclidean space

Randomized numerical linear algebra algorithms approximating fundamental properties of data

Relevant approximation properties used to solve big data analytics issues
Thesis Contributions
Thesis Contributions

- Outlier Detection
- Classification
- Similarity Estimation
- Randomized Algorithms
Outlier Detection

- We introduce **FastVOA**, a near-linear time algorithm to approximate the variance of angles between pairs of data points, a robust outlier score to detect high-dimensional outlier patterns.

---

**A Near-linear Time Approximation Algorithm for Angle-based Outlier Detection in High-dimensional Data**

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

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

*Proceedings of 18th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), 2012.*
What Is an Outlier?

**Definition:**
An object *deviates significantly* from *normal* objects

**Applications:**
- Credit card fraud detection
- Network intrusion detection
Outlier Factor

- Measure the degree of outlier-ness
- Traditional outlier factors:
  - kNN distance [RRS’00]
  - Local density [BKNS’00]
- Angle-based outlier factor [KSZ’08]
  - The smaller the variance of angle between a point to other pairs of points is, the more likely it is an outlier.

The variance of angles for different kinds of points
Angle-based Outlier Factor

- Angle-based outlier factor of point $p$:
  
  $$VOA(p) = \text{Var}[\Theta_{apb}] = MOA_2(p) - MOA_1^2(p)$$

  where $MOA_2(p)$ and $MOA_1(p)$ are defined as follows:

  $$MOA_2(p) = \frac{\sum_{a,b \in S \setminus \{p\}} \Theta_{apb}^2}{\frac{1}{2}(n-1)(n-2)}; \quad MOA_1(p) = \frac{\sum_{a,b \in S \setminus \{p\}} \Theta_{apb}}{\frac{1}{2}(n-1)(n-2)}$$

- We propose FastVOA to estimate $MOA_2(p)$, $MOA_1(p)$ and $VOA(p)$ in time near-linear in the size of the data.
FastVOA – An Overview

We project all points into the hyperplane orthogonal to $\vec{r}_i$ and sort them by their dot products. We present each partition $L_p^{(i)}$, $R_p^{(i)}$ as binary vectors $u^{(i)}$, $v^{(i)}$, respectively. Applying AMS Sketches, we can approximate $\|u^{(i)} \otimes v^{(i)}\|_F$, which is then used to estimate VOA.
Random Hyperplane Projection

\[ \Theta_{apb} \]

**Outlier Detection**

\[ u^{(i)} = \{1,1,1,1,1,0,0,0,0\} \quad , \quad AMS(u^{(i)}) = s_l \cdot u^{(i)} \]

\[ v^{(i)} = \{0,0,0,0,0,1,1,1,1\} \quad , \quad AMS(v^{(i)}) = s_r \cdot v^{(i)} \]

- \[ X_{apb}^{(i)} = \begin{cases} 
1 & \text{if } a \cdot r_i < p \cdot r_i < b \cdot r_i \\
0 & \text{otherwise} 
\end{cases} \]

\[ P = \sum_{i=1}^{t} (u^{(i)} \otimes v^{(i)}) \]

\[ \Theta_{apb}^2 = \frac{(2\pi)^2}{t(t-1)} \left( \mathbf{E}[P_{kl}^2] - \frac{t}{2\pi} \Theta_{apb} \right) \]

\[ P_{kl} \] is the number of times that \( a \) locates on the left side and \( b \) locates on the right side after \( t \) projection.
Estimators

- **First moment estimator** $F_1(p)$:
  
  $$MOA_1(p) = \frac{4\pi}{(n-1)(n-2)} \sum_{a,b\in S\setminus\{p\}}^{a\neq b} E[X_{apb}^{(i)}]$$
  
  $$F_1(p) = \frac{2\pi}{t(n-1)(n-2)} \sum_{i=1}^{t} |L_p^{(i)}| |R_p^{(i)}|$$

- **Second moment estimator** $F_2(p)$:
  
  $$MOA_2(p) = \frac{4\pi^2}{t(t-1)(n-1)(n-2)} E\left[\|P\|_F^2\right] - \frac{2\pi}{t-1} MOA_1(p)$$
  
  $$F_2(p) = \frac{4\pi^2}{t(t-1)(n-1)(n-2)} \left(\sum_{i=1}^{t} AMS(u^{(i)})AMS(v^{(i)})\right)^2 - \frac{2\pi}{t-1} F_1(p)$$
FastVOA Algorithm

Algorithm 1 FastVOA(S, t, s_1, s_2)

Ensure: Return the variance estimator for all points

1: \( VAR \leftarrow [0]^n \); \( F_2 \leftarrow [0]^n \)
2: \( \mathcal{L} \leftarrow \text{RandomProjection}(S, t) \)
3: \( F_1 \leftarrow \text{FirstMomentEstimator}(\mathcal{L}, t, n) \)
4: for \( i = 1 \rightarrow s_2 \) do
5: \( Y_i \leftarrow \sum_{j=1}^{s_1} (\text{FrobeniusNorm}(\mathcal{L}, t, n))^2 / s_1 \)
6: end for
7: \( F_{\text{norm}} \leftarrow \text{median} \{ Y_1, \cdots, Y_{s_2} \} \)
8: for \( j = 1 \rightarrow n \) do
9: \( F_2[j] = \frac{4\pi^2}{t(t-1)(n-1)(n-2)} F_{\text{norm}}[j] - \frac{2\pi F_1[j]}{t-1} \)
10: \( VAR[j] = F_2[j] - (F_1[j])^2 \)
11: end for
12: return \( VAR \)

FastVOA runs in \( O(tn(d+\log n+s_1s_2)) \) time with \( n \) points in \( d \)-dimensional space using \( t \) random projections and AMS Sketch size \( s_1 \) and \( s_2 \).
Experiments

- **Accuracy**: The deviation error from expectation of unbiased variance estimators on 5 data sets (50 & 100-dimensional synthetic, Isolet, Multiple Features and Optical Digits datasets).
Experiments

- **Effectiveness**: The capability of algorithms \{Simple VOA, Fast VOA\} vs. \{ABOD, FastABOD\} [KSZ’08] to retrieve the most likely outliers on Isolet and Multiple Features datasets.
Experiments

- **Efficiency**: The CPU time of algorithms \{FastVOA\} vs. \{FastABOD, LB_ABOD\}[KSZ’08] on synthetic datasets.

(a) 100 dimensional synthetic datasets  
(b) Datasets of 20,000 points
Open Problems

- Design and implement FastVOA on MapReduce framework

- **Sampling-based algorithms** for approximating angle-based outlier factor and its variants

- Relationship between angle-based outlier factor and data depth in statistics
We propose **Tensor Sketch**, a fast random feature mapping for approximating non-linear kernels and accelerating the training kernel machines for large-scale classification problems.

**Fast and Scalable Polynomial Kernels via Explicit Feature Maps**

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

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

*Proceedings of 19th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD), 2013.*
Support Vector Machine (SVM)

- **Non-linear SVM**
  - Constructing hyperplane classifiers in high or infinitive dimensional space by using kernel tricks
  - Direct methods run in $O(dn^3)$ time

- **Linear SVM**
  - Constructing hyperplane classifiers in data space because the data space is almost linearly separable.
  - Fast solvers run in $O(dn)$ time
From Linear SVMs to Nonlinear SVMs

- **Random feature mapping** \( f \) [RR’07] from the original data space to the randomized feature space such that:

\[
E \left[ \langle f(x), f(y) \rangle \right] = \langle \Phi(x), \Phi(y) \rangle = \kappa(x, y)
\]

- **Tensor Sketch** - a random feature mapping for polynomial kernels

\[O(dn^3)\] \[O(dn)\]
Tensor Sketch: An Overview

Data space

Tensor Product

Tensor Sketches

Polynomial feature space

Count Sketches

Randomized feature space

Classification
Tensor Sketch: An Overview

Data

\[ x \]

[\text{\( p \) times}]

\[ x \]

\( \mathbb{R}^d \)

Count Sketch

\[ C^{(1)} x \]

\[ C^{(p)} x \]

\( \mathbb{R}^D \)

Tensor Sketch

\[ C x^{(p)} \]

\( \mathbb{R}^D \)

Convolution

Count Sketch

\[ x^{(p)} \]

\( \mathbb{R}^{d^p} \)

Tensor Product
Count Sketches

• **Definition:** Given hash functions \( h : [d] \mapsto [D] \) and \( s : [d] \mapsto [\pm 1] \). Count Sketch of a point \( x = \{x_1, \ldots, x_d\} \in \mathbb{R}^d \) is denoted by \( Cx = \{(Cx)_1, \ldots, (Cx)_D\} \in \mathbb{R}^D \) where \((Cx)_j = \sum_{i: h(i) = j} s(i)x_i \).

• **Example:**

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>+1</td>
<td>-1</td>
<td>-1</td>
<td>+1</td>
</tr>
</tbody>
</table>

\[
\begin{array}{cccc}
    d = 4 &  & D = 3 \\
    \begin{array}{cccc}
        1 & 2 & 3 & 1 \\
    \end{array} & \rightarrow & \begin{array}{ccc}
        +1 & -1 & -3
    \end{array}
\end{array}
\]

• **Properties:**

\[
\begin{align*}
    \mathbb{E} \left[ \langle Cx, Cy \rangle \right] &= \langle x, y \rangle, \\
    \text{Var} \left[ \langle Cx, Cy \rangle \right] &\leq \frac{1}{D} \left( \langle x, y \rangle^2 + \|x\|^2 \|y\|^2 \right).
\end{align*}
\]
Convolution of Count Sketches

- **Observations on outer product domain [Pagh’12]:**
  - View count sketching as the polynomial transforming
    - \( P_1(\omega) = \sum_{i=1}^{d} s_1(i) x_i \omega^{h_1(i)} \) with hash functions \( h_1 \) and \( s_1 \)
    - \( P_2(\omega) = \sum_{i=1}^{d} s_2(i) x_i \omega^{h_2(i)} \) with hash functions \( h_2 \) and \( s_2 \)
  - \( P(\omega) \) is a Count Sketch of outer product with hash functions \( H(i, j) = h_1(i) + h_2(j) \mod D \) and \( S(i, j) = s_1(i)s_2(j) \):
    \[
    P(\omega) = \text{FFT}^{-1}(\text{FFT}(P_1(\omega) \ast \text{FFT}(P_2(\omega))))
    \]
  - \( P(\omega) \) can be seen as an explicit random feature mapping (random projection) for the **degree-2 polynomial kernel** (polynomial feature space).
Convolution of Count Sketches

- **Generalization on tensor product domain:**
  - $P(\omega)$ is a Count Sketch of $p$-level tensor product with hash functions:
    - $H(i_1, \ldots, i_p) = h_1(i_1) + \ldots + h_p(i_p) \mod D$,
    - $S(i_1, \ldots, i_p) = s_1(i_1) \ldots s_p(i_p)$
  - Fast computation $P(\omega) = \text{FFT}^{-1}(\text{FFT}(P_1(\omega) \ast \ldots \ast \text{FFT}(P_p(\omega))))$
  - $P(\omega)$ can be seen as an explicit random feature mapping (random projection) for the degree-$p$ polynomial kernel (polynomial feature space).
Tensor Sketch

Tensor Sketch runs in $O(np(d+D\log D))$ time with $n$ points in $d$-dimensional space using $D$ random features.
Experiments

• **Random feature construction time:**

Comparison of CPU time (s) between Tensor Sketching (TS) and Random Maclaurin (RM) [KK’12] approaches on synthetic, Adult, and Mnist datasets, using $\kappa = \left(1 + \langle x, y \rangle \right)^4$.

(a) CPU Time (s) on Adult and Mnist datasets

(b) CPU Time (s) on synthetic dataset
Experiments

- Accuracy of classification on different kernels:

\[ K = \langle x, y \rangle^2 \]

\[ K = \left( 1 + \langle x, y \rangle \right)^2 \]
Experiments

- **Training time and accuracy of classification:**

  Comparison between Linear SVMs (LIBLINEAR) + Tensor Sketch (TS) or Random Maclaurin (RM) and non-linear SVMs (LIBSVM) on 2 datasets: Mnist ($d = 780, D = 1,000$) and Adult ($d = 123, D = 200$)

<table>
<thead>
<tr>
<th>Kernel</th>
<th>$\kappa$+libsvm</th>
<th>TS+liblinear</th>
<th>RM+liblinear</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\langle x, y \rangle^4$</td>
<td>97.17% 5 mins</td>
<td>92.49±0.22% 2.1 mins</td>
<td>41.45±4.81% 0.5 mins</td>
</tr>
<tr>
<td>$(1 + \langle x, y \rangle)^4$</td>
<td>97.31% 5 mins</td>
<td>92.44±0.04% 2.1 mins</td>
<td>90.07±0.65% 5 mins</td>
</tr>
<tr>
<td>$\langle x, y \rangle^4$</td>
<td>79.34% 2 hours</td>
<td>81.09±0.63% 4.3 secs</td>
<td>58.04±2.37% &lt; 1 sec</td>
</tr>
<tr>
<td>$(1 + \langle x, y \rangle)^4$</td>
<td>79.31% 2 hours</td>
<td>81.89±0.24% 4.5 secs</td>
<td>84.04±0.46% 14.8 secs</td>
</tr>
</tbody>
</table>
Open Problems

• Apply Tensor Sketch on other kernel functions by exploiting Taylor-series approximations of these kernels

• Analyze and evaluate Tensor Sketches on other large-scale learning tasks, such as kernel clustering, multitask learning
Similarity Estimation

- We present **Odd Sketch**, a space-efficient probabilistic data structure for estimating high Jaccard similarities between sets, a central problem in information retrieval applications.

---

**Efficient Estimation for High Similarities using Odd Sketches**

Michael Mitzenmacher  
Harvard University  
Cambridge, MA  
michaelm@eecs.harvard.edu

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

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

Set Similarity and Its Uses

• **Jaccard similarity coefficient:**

\[ J(A, B) = \frac{|A \cap B|}{|A \cup B|}, \quad 0 \leq J \leq 1 \]

• **Some uses:**

  - Web duplication detection
  - Collaborative filtering
### MinHash (Minwise Hashing)

We have sets and their elements, random permutations, rearrangement, and MinHashes:

<table>
<thead>
<tr>
<th>id</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\pi_1$</th>
<th>$\pi_2$</th>
<th>$\pi_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\pi_1$</th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

MinHashes:

<table>
<thead>
<tr>
<th>$S_1$</th>
<th>$S_2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Rearrangement:

$\hat{J} = \frac{|S_1 \cap S_2|}{3} = \frac{2}{3}$

$J(A, B) = \frac{2}{3}$
MinHash (Minwise Hashing)

- **MinHash theorem:**

\[
\Pr\left[ \min_i(\pi_i(A)) = \min_i(\pi_i(B)) \right] = J(A, B)
\]

- Denote \( S_1 \) and \( S_2 \) by minhashes of \( A \) and \( B \) by considering \( k \) independent permutations \( \pi_1, \ldots, \pi_k \)

\[
S_1 = \left\{ \min_i(\pi_i(A)) \mid i = 1, \ldots, k \right\},
\]

\[
S_2 = \left\{ \min_i(\pi_i(B)) \mid i = 1, \ldots, k \right\},
\]

we obtain an unbiased estimator of \( J(A, B) \) and its variance:

\[
\hat{J} = \frac{|S_1 \cap S_2|}{k}, \quad \text{Var} \left[ \hat{J} \right] = \frac{J(1 - J)}{k}.
\]
**b-Bit MinHash**

<table>
<thead>
<tr>
<th>MinHashes</th>
<th>3-Bit MinHashes</th>
<th>1-Bit MinHashes</th>
</tr>
</thead>
<tbody>
<tr>
<td>S₁ S₂</td>
<td>S₁ S₂</td>
<td>S₁ S₂</td>
</tr>
<tr>
<td>3 5</td>
<td>011 101</td>
<td>1 1</td>
</tr>
<tr>
<td>1 1</td>
<td>001 001</td>
<td>1 1</td>
</tr>
<tr>
<td>1 1</td>
<td>001 001</td>
<td>1 1</td>
</tr>
</tbody>
</table>

\[ \hat{J} = \frac{2}{3} \]

\[ \hat{J}^{b=3} = \frac{2}{3} \]

\[ \hat{J}^{b=1} = \frac{|S_1 \cap S_2|}{3 - 1/2} \]

\[ 1 - 1/2 \]

**Intuition:**

- The same hash values give the same lowest \( b \) bits.
- Different hash values give different lowest \( b \) bits with probability \( 1 - 1/2^b \).
**b-Bit MinHash**

- **b-Bit MinHash theorem**
  - Denote $\min_b(\pi(A))$ by the lowest $b$ bits of the hash value $\min(\pi(A))$, we obtain $b$-bit minhashes of $A$ and $B$
    
    $$
    S^b_1 = \{\min_b(\pi_i(A)) \mid i = 1, \ldots, k\},
    $$
    $$
    S^b_2 = \{\min_b(\pi_i(B)) \mid i = 1, \ldots, k\}.
    $$

  We obtain an unbiased estimator for $J(A,B)$ and its variance

  $$
  \hat{J}^b = \frac{|S^b_1 \cap S^b_2|}{k - 1/2^b}, \quad \text{Var} \left[ \hat{J}^b \right] = \frac{1-J}{k} \left( J + \frac{1}{2^b - 1} \right).
  $$

- $b$-bit MinHash uses more permutations than MinHash.
Challenge

- When the Jaccard similarity is high, $b$-bit MinHash offers less information due to the two likely identical $b$-bit summaries.

- Inaccuracy in just a few bit positions (white space) will yield a large relative error of the estimate of $J$. 

$b$-bit MinHash Construction

- Randomized Algorithms
- Similarity Estimation

Different independent hash values

Same independent hash values
Odd Sketch: Intuition

• **The Bloom filter principle:**
  - Wherever a list or set is used, and space is at a premium, consider using a Bloom filter if the effect of false positives can be mitigated.

• **The Odd Sketch:**
  - A Bloom filter using **one** hash function with an “odd” feature that the usual disjunction (OR) is replaced by an exclusive-or operation (XOR).
  - Constructed on the original minhashes ($S_1$ and $S_2$).
Odd Sketch: An Overview

- **Property:**
  \[
  \text{odd}(S_1) \oplus \text{odd}(S_2) = \text{odd}(S_1 \Delta S_2)
  \]

- When \( J \) is close to 1, we can use odd sketches of size \( n \) on minhashes of size **significantly above** \( n \). So the variance induced by the minhash step is reduced.

![Odd Sketch Construction](image)

- **Randomized Algorithms**
- **Similarity Estimation**
Odd Sketch: Estimation

- **Jaccard similarity estimation:**
  1. Estimate $|S_1 \Delta S_2|$ from its sketch $\text{odd}(S_1 \Delta S_2)$
  2. Estimate $J(A, B)$ by

     \[
     \hat{J}^{\text{odd}} = \frac{|S_1 \cap S_2|}{k} = 1 - \frac{|S_1 \Delta S_2|}{2k}.
     \]

- **Problem:** How to estimate the set’s size from its odd sketch?
  - Constructing $\text{odd}(S)$ of size $n$ (bits) of a set $S$ of $m$ elements as independently throwing $m$ balls into $n$ bins, and storing the parity of the number of balls in each bin.
  - We will estimate $m$ based on the observation of the number of “odd” bins in the sketch of size $n$. 
Estimate a Set’s Size from Its Odd Sketch

- **Poisson approximation (independent case):**
  - When $m$ balls are thrown into $n$ bins, this is very approximately the same as independently giving each bin a number of balls that is Poisson distributed with mean $m/n$.

- **Lemma:**
  - Let $Q$ be a random variable that has Poisson distribution with mean $m/n$. The probability $p$ that $Q$ is odd is $\left(1 - e^{-2m/n}\right)/2$.

- Given $Z$ odd bins in the sketch, we obtain an estimate

\[
\hat{m} = -\frac{n}{2} \ln \left(1 - \frac{2Z}{n}\right)
\]
Estimate a Set’s Size from Its Odd Sketch

- **Two-state Markov chain (dependent case):**
  - Changing of the parity of number of balls in any specific bin is a two-state Markov chain with the probability of changing state is $1/n$.
  - Let $p_i$ be the probability that any specific bin has an odd number of balls after $i$ balls have been thrown, we have
    \[ p_i = \frac{1 - (1 - 2/n)^i}{2} \]
  - Given $Z$ odd bins in the sketch, we obtain an estimate
    \[ E[Z] = n \frac{1 - (1 - 2/n)^m}{2} \]
    \[ \hat{m} = \frac{\ln(1 - 2Z/n)}{\ln(1 - 2/n)} \]
Estimate Jaccard Similarity from Odd Sketches

• We construct odd sketches on the original minhashes.

1. \( E\left[|S_1 \Delta S_2|\right] = 2k(1 - J) \),

2. \( \text{odd}(S_1) \oplus \text{odd}(S_2) = \text{odd}(S_1 \Delta S_2) \).

• We rely on the Poisson approximation approach, and estimate the symmetric difference \( |S_1 \Delta S_2| \)

\[
|S_1 \Delta S_2| = -\frac{n}{2} \ln \left( 1 - \frac{2|\text{odd}(S_1) \oplus \text{odd}(S_2)|}{n} \right).
\]

• We estimate the Jaccard similarity

\[
\hat{J}^{\text{odd}} = 1 + \frac{n}{4k} \ln \left( 1 - \frac{2|\text{odd}(S_1) \oplus \text{odd}(S_2)|}{n} \right).
\]
Experiments

- **Parameter setting:**
  - Performance of both odd sketch and $b$-bit MinHash depends on the number of independent permutations $k$.
  - $b$-bit scheme uses $k_b = \frac{n}{b}$ where $n$ (bits) is the size of sketch.
  - Odd sketch uses $k_{odd} = \frac{n}{4(1-J_0)}$ where $J_0$ is the user-defined similarity threshold (heuristic).
  - **Observation:** When $J_0 > 0.75 \rightarrow k_{odd} > k_b$, odd sketch achieves better accuracy than $b$-bit scheme.
Accuracy

Comparison of accuracy between Odd Sketch and $b$-bit MinHash on sparse synthetic dataset with sketch size of 512 bits.

$$k_{\text{odd}} = \frac{n}{4(1-J)} \quad \text{and} \quad k_b = \frac{n}{b}$$
Association Rule Learning

Comparison on precision/recall ratio between Odd Sketch and 1-bit MinHash on the mushroom dataset on detecting pairwise items that have \( J > J_0 = 0.9 \).
Web Duplication Detection

Comparison on precision/recall ratio between Odd Sketch and 1-bit MinHash on the KOS blog entries and Enron emails datasets on detecting pairwise documents that have $J > J_0 = 0.9$. 

KOS blog

Enron emails
Open Problems

- Evaluate Odd Sketch on join operations (e.g. Top-k similarity join, near neighbor join)

- Can we extend Odd Sketch on general set reconciliation problem?
Research Direction

- Exploit the preservation of standard Euclidean structures under random projections.

The standard Euclidean structures of data can be efficiently computed via encodings.
No guide, no realization

- Rasmus Pagh
- Hoang Thanh Lam
- Michael Mitzenmacher
- John Langford & CUSP members
- Algorithms Group members & PhD committee
- My parents – Pham Sy Hy & Le Thi Thanh Ha
- Quang Loc Le, Thai Son Mai, Quoc Viet Hung Nguyen, Thu Thuy Tran
- Friends from DTU, KU & Lieu Quan temple.
- IT University of Copenhagen & Danish National Research Foundation
Mange tak fordi I kom