Introduction

Metric embedding:
Starting point of many algorithms

Real-world objects
High-dimensional feature vectors
(images, text, etc.)

Goal: Compress vectors while approximately preserving distances.
• Many algorithms for data analysis and machine learning rely on distances
• E.g.: Nearest neighbor queries

Benefits of compression:
• Time: Speed-up linear scan of data
• Space: Fit on memory-limited devices like GPUs

Our algorithm:
• Simple to describe and implement
• Provably pointwise guarantees
• Matches or outperform state-of-the-art in the high-precision regime

Previous work: Either heuristic or impractical.

Heuristic algorithms:
• Lack provable guarantees - may be unsuitable for non-standard datasets
• Optimize for average accuracy - may perform undesirably on individual queries
• Solve a global optimization problem on the dataset (e.g. k-means) - slow or infeasible in high precision regime

Theoretical algorithms:
Unsuitable for implementation despite asymptotic guarantees, due to large hidden constants, underlying combinatorial complexity, etc.

QuadSketch:
Algorithm Description

Construction
• Step 1: Randomly shifted grids
Enclose points in hypercube. Refine into sub-cubes by halving each dimension. Repeat refinement for \( L \) levels. Shift grids by a uniformly random vector.

• Step 2: Quadtree
Construct high-dimensional quadtree from grids:
• The root is the enclosing hypercube.
• For every non-empty sub-hypercube, add child node.

• Step 3: Pruning
For every tree path longer than \( \Lambda \):
Replace the path after the top \( \Lambda \) nodes with a long edge.

The compressed representation is the pruned quadtree.

Recovery
To recover the approximation \( \bar{x} \) of a point \( x \):
• Follow path from root to leaf containing \( x \).
• In each dimension, concatenate bits along edges in path.
• If long edge, concatenate zeros instead.

Experiments

We compare:
• QS: Product QuadSketch
Partition into blocks, QuadSketch in each
• PQ: Product Quantization (Jégou, Douze, Schmid 2011)
Partition into blocks, k-means in each
• Grid: Uniform scalar quantization (baseline)

We report:
• Accuracy - fraction of correct nearest neighbors
• Size - bits per coordinate

Datasets:
$$\begin{array}{c|c|c|c}
\text{Datasets} & n & d & \Phi \\
\hline
\text{SIFT} & 1M & 128 & \geq 83.2 ^* \\
\hline
\text{MNIST} & 60K & 784 & \geq 9.2 ^* \\
\hline
\text{NYC Taxi ridership} & 8,874 & 48 & 49.5 \\
\hline
\text{Diagonal (synthetic)} & 10K & 128 & 20,478,740.2 \\
\end{array}$$

* Estimated on a random sample.
** Random points on a line, embedded in a 128-dimensional space.

Theoretical Results

Parameters:
\( n \) - num. points; \( d \) - dimension; \( \Phi \) - ratio of maximum to minimum distance (captures numerical range)

Theorem: Given \( \epsilon, \delta > 0 \), set
\[ \Lambda = \log \left( \frac{16 \cdot d^{1.5} \cdot \log \Phi}{\epsilon \cdot \delta} \right) \]
and \( L = \Lambda + \log \Phi \).

QuadSketch guarantees: For every point \( x \),
\[ \Pr[|y - \bar{y}| \leq (1 + \epsilon)|x - y|] > 1 - \delta. \]
• In particular, \((1 + \epsilon)\)-approximate nearest neighbors are preserved with probability \(1 - \delta\).
• Construction time: \( O(nd\Lambda) \).
• Compressed size: \( O(nd\Lambda + n\log n) \) bits.

Comparison with prior work:
For \( d = O(\epsilon^{-2}\log n) \) by dimension reduction, and \( \Phi = \alpha \frac{1}{\log n} \)

<table>
<thead>
<tr>
<th>Reference</th>
<th>Bits per coordinate</th>
<th>Construction time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vanilla bound</td>
<td>( O(\log n) )</td>
<td>--</td>
</tr>
<tr>
<td>(Indyk, Wagner 2017)</td>
<td>( O(\log(1/\epsilon)) )</td>
<td>( \tilde{O}(n^{1+\alpha} + \epsilon^{-2}n) ) for ( \alpha \in (0, 1) )</td>
</tr>
<tr>
<td>This work</td>
<td>( O(\log \log n + \log(1/\epsilon)) )</td>
<td>( \tilde{O}(n^{1+\alpha} + \epsilon^{-2}n) ) for ( \alpha \in (0, 1) )</td>
</tr>
</tbody>
</table>

References:
IEEE transactions on pattern analysis and machine intelligence, 2011.