FALCONN: Practical and Optimal LSH for Angular Distance
Alexandr Andoni (Columbia University), Piotr Indyk (MIT), Thijs Laarhoven (IBM Research), Ilya Razenshteyn (MIT), Ludwig Schmidt (MIT)

Similarity Search
- **Dataset**: n points in a metric space, say $\mathbb{R}^d$ with Euclidean distance.
- **Goal**: preprocess the dataset so that, given a query point, we can quickly return the closest data points.
- **High-dimensional case**: want a nearly-linear dependence on the dimension $d$.

Spherical Case
- All data points and queries lie on a unit sphere in $\mathbb{R}^d$.
- Corresponds to the cosine similarity.

Locality-Sensitive Hashing
- Random partitions of the metric space into cells.
- **Quality**: the gap between collision probabilities for pairs of points that are close ($p_0$) and pairs of points that are far apart ($p_1$)
  $$\rho = \log(1/p_0)/\log(1/p_1).$$
- **Overall**: space usage $O(n^{1+\rho})$ and query time $O(n^\rho)$.

Hyperplane LSH
- To partition a unit sphere, cut it across a random hyperplane through the origin.
- For vectors with angle $\alpha$ between them, the probability of collision is $1 - \alpha/\pi$.
- Widely used in practice for similarity search. However, there is a better theoretical construction that is known to have optimal $\rho$.

Our Contribution
- We theoretically analyze the cross-polytope LSH of [Terasawa, Tanaka 2007] and make it practical.
- **Best of both worlds**: LSH for the cosine similarity that
  - Achieves the optimal theoretical guarantees;
  - Is significantly better than Hyperplane LSH in practice.

Synthetic Datasets
- Random data set with planted queries.
- Target accuracy 0.9 for finding the nearest neighbor.
- Data structure size is roughly the dataset size.

Real Datasets
- Accuracy 0.9 and index size ~ dataset size as above.
- For NYT and Pubmed we limit to interesting queries.

<table>
<thead>
<tr>
<th></th>
<th>n</th>
<th>d</th>
<th>Scan</th>
<th>HP</th>
<th>CP</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIFT</td>
<td>1M</td>
<td>128</td>
<td>36ms</td>
<td>3.7 ms</td>
<td>3.1 ms</td>
</tr>
<tr>
<td>NYT</td>
<td>300K</td>
<td>100K</td>
<td>480 ms</td>
<td>120 ms</td>
<td>35 ms</td>
</tr>
<tr>
<td>Pubmed</td>
<td>8M</td>
<td>140K</td>
<td>3.6 s</td>
<td>857 ms</td>
<td>213 ms</td>
</tr>
</tbody>
</table>

Optimized C++ implementation of Hyperplane LSH and the new LSH family is available at https://falconn-lib.org (released under the MIT License).