Learning to Route in Similarity Graphs

Dmitry Baranchuk

joint work with Dmitry Persiyanov, Anton Sinitsin and Artem Babenko

Yandex
Overview

The Budgeted Nearest Neighbor Search Problem

Similarity Graphs

Learning to Route in Similarity Graphs

Evaluation
The Budgeted Nearest Neighbor Search Problem

- \{x_1, ..., x_N\} \subset \mathbb{R}^D — search database
- q \in \mathbb{R}^D — query
- DCS — maximal number of distance computations
- Recall@1 — a rate of queries for which the actual nearest neighbor is successfully found
Similarity Graphs

- **Vertices** correspond to the database items
- **Edges** connect (mostly) nearest neighbors

Several state-of-the-art methods exist e.g. HNSW\textsuperscript{1}, NSG\textsuperscript{2}

---

\textsuperscript{1}Malkov, Y., Yashunin, D. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. TPAMI 2018

\textsuperscript{2}Cong Fu, Chao Xiang, Changxu Wang, and Deng Cai. Fast approximate nearest neighbor search with the navigating spreading-out graph. PVLDB 2019
Routing Algorithms

- **Greedy routing**: Pick the best neighbor of the current vertex
- **Beam search**: Expand the most promising vertex in the candidate pool
- **Our method**: Learn a routing algorithm directly from data
1. **Imitation Learning**: Train the agent to imitate expert decisions

2. **Agent** is a beam search based on learned vertex representations

3. **Expert** encourages the agent to follow a shortest path to the actual nearest neighbor $v^*$

---

Ross, S., Gordon, G. J., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. AISTATS 2011
Model Architecture

**Graph Convolutional Network** learns representations for vertices that account for the underlying structure of the similarity graph.

Graph Convolutional Block

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. ICLR 2017
Evaluation

- Datasets with $10^5$ points
- No additional cost in run-time
- PyTorch implementation\(^3\)

<table>
<thead>
<tr>
<th>DCS budget</th>
<th>Vertex Representations</th>
<th>SIFT100K Recall@1</th>
<th>DEEP100K Recall@1</th>
<th>GloVe100K Recall@1</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>Original</td>
<td>0.239</td>
<td>0.386</td>
<td>0.198</td>
</tr>
<tr>
<td></td>
<td>Learned</td>
<td><strong>0.371</strong></td>
<td><strong>0.474</strong></td>
<td><strong>0.305</strong></td>
</tr>
<tr>
<td>256</td>
<td>Original</td>
<td>0.672</td>
<td>0.795</td>
<td>0.400</td>
</tr>
<tr>
<td></td>
<td>Learned</td>
<td><strong>0.799</strong></td>
<td><strong>0.811</strong></td>
<td><strong>0.526</strong></td>
</tr>
<tr>
<td>512</td>
<td>Original</td>
<td>0.936</td>
<td>0.940</td>
<td>0.582</td>
</tr>
<tr>
<td></td>
<td>Learned</td>
<td><strong>0.949</strong></td>
<td><strong>0.945</strong></td>
<td><strong>0.676</strong></td>
</tr>
</tbody>
</table>

Search performance Recall@1 for distance computation (DCS) budgets

\(^3\)https://github.com/dbaranchuk/learning-to-route