Minimax-Optimal Semi-supervised Regression

Geodesic kNN on Unknown Manifolds

Ziyue Chu & Keng Xu

University of Edinburgh

Contact Information:
School of Informatics
University of Informatics
10 Crichton St, Edinburgh EH8 9AB, UK

Phone: (+44) 131 650 2957
Email: `zero-cooper@foxmail.com`

Introduction

Previous **defects** on semi-supervised regression and classification methods [1][2]:

- Unlabeled data is not useful with unknown manifolds.
- Require infinity labeled samples.
- The results only achieve asymptotic minimax rate.

Objectives:

- How to use unlabeled data to help the learning process?
- How to design statistically sound and computationally efficient methods with unlabeled data?

Methodology

Algorithm 1 Geodesic KNN

Define: Data space: $\mathcal{X} = \mathcal{L} \cup \mathcal{U}$, where $\mathcal{L} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n}$ are n labeled instances, $\mathcal{U} = \{\mathbf{x}_j\}_{j=1}^{m}$ are m unlabeled instances.

Define: Distance function: $d(\mathbf{x}, \mathbf{x}')$.

Step 1: Construct an undirected graph G whose vertices are all the labeled and unlabeled points, with edge weight $w(\mathbf{x}, \mathbf{x}') = d(\mathbf{x}, \mathbf{x}')$ connecting each pairs of points $\mathbf{x}, \mathbf{x}'$.

Step 2: Compute the shortest-path graph distance $d_G(\mathbf{x}_i, \mathbf{x}_j), \forall \mathbf{x}_i \in \mathcal{L}$ and $\mathbf{x}_j \in \mathcal{L} \cup \mathcal{U}$.

Step 3: Apply standard metric-based supervised learning methods (e.g. kNN) with d_G.

Step 4: Geodesic kNN regressor updates:

$$\hat{f}(\mathbf{x}_i) := \frac{1}{|kNN(\mathbf{x}_i)|} \sum_{(\mathbf{x}_j, y_j) \in kNN(\mathbf{x}_i)} y_j.$$

Step 5: Geodesic kNN predictor:

$$\hat{f}(\mathbf{x}) := \hat{f}(\mathbf{x}^*) = \hat{f}\left(\underset{\mathbf{x}' \in \mathcal{L} \cup \mathcal{U}}{\arg\min} ||\mathbf{x} - \mathbf{x}'|| \right).$$

Claims

- If Regressor $\hat{f} \in$ Lipschitz function:
- Geodesic kNN regressor error expectation is up-bounded and $\hat{f}$ is a minimax-optima.

Theorem 1.

$$\mathbb{E}\left[\left(\hat{f}(\mathbf{x}) - f(\mathbf{x}) \right)^2 \right] \leq cn^{-\frac{2}{2+d}} + c'e^{-c''(n+m)} f_D^2$$

Proof:

$$\mathbb{E}\left[\left(\hat{f}(\mathbf{x}) - f(\mathbf{x}) \right)^2 \right] = \mathbb{E}\left[\left(\hat{f}(\mathbf{x}^*) - f(\mathbf{x}) \right)^2 \right]$$
$$\leq 2\mathbb{E}\left[\left(\hat{f}(\mathbf{x}^*) - f(\mathbf{x}^*) \right)^2 \right] + 2\mathbb{E}\left[\left(f(\mathbf{x}^*) - f(\mathbf{x}) \right)^2 \right]$$

Lemma 1. Second term is up-bounded:

$$\mathbb{E}\left[\left(f(\mathbf{x}^*) - f(\mathbf{x}) \right)^2 \right] \leq \frac{2L^2}{(1 - e^{-Q})^2} n^{-\frac{2}{2+d}} + e^{-QR^d(n+m)} f_D^2$$

Lemma 2. First term is up-bounded:

$$\mathbb{E}\left[\left(\hat{f}(\mathbf{x}^*) - f(\mathbf{x}^*) \right)^2 \right] \leq 4\, c_a e^{-c_b \mu_{min}(n+m)} f_D^2 + \left(2L^2 \left(\frac{1+\delta}{1-\delta} \right)^2 c_1(\mathcal{M}, \mu, \delta) + \sigma^2 \right) n^{-\frac{2}{2+d}}$$

Algorithm Implemention

Algorithm 2 Computation of Geodesic KNN [with faster priority queue handling]

Input: An undirected weighted graph $G = (V, E, w)$ and a set of labeled vertices $\mathcal{L} \in V$.
Output: For every $v \in V$ a list $kNN[v]$ with the k nearest labeled vertices to v and their distances.
$Q \leftarrow$ PriorityQueue()
for $v \in V$ **do**
  [$Q_v \leftarrow$ PriorityQueue()]
  $kNN[v] \leftarrow$ Empty-List(), $S_v \leftarrow \phi$
  if $v \in \mathcal{L}$ **then**
    insert $(Q, (v, v)$ $[(Q, v)\&(Q_v, v)],$ $prio = 0)$
while $Q \neq \phi$ **do**
  (seed, v0, dist) [(v0, dist) & (seed, dist)] $\leftarrow$ pop-minimum(Q [Q_{v_0}]), $S_{v_0} \leftarrow S_{v_0} \cup \{seed\}$
  if length(kNN[v_0]) ¡ k (and $Q_{v_0} \neq \phi$) **then**
    append (seed, dist) **to** kNN[v_0]
    [(newseed, newdist)$\leftarrow$ min(Q_{v_0}) & insert]
  for all $v \in$ neighbors(v_0) **do**
    if length(kNN[v]) ¡ k and seed $\notin S_v$ **then**
      decrease-or-insert(Q, (seed, v), [Q_v, seed & Q,v] priority = dist $+w(v_0, v)$)

Computational Efficiency

The computational efficiency comparison between the two algorithms are shown below:

Algorithms	Runtime
Algorithm 2.1	$O(k\|E\| + N_p log\|V\|)^1$
Algorithm 2.2	$O(k\|E\| + kV log\|V\|)$

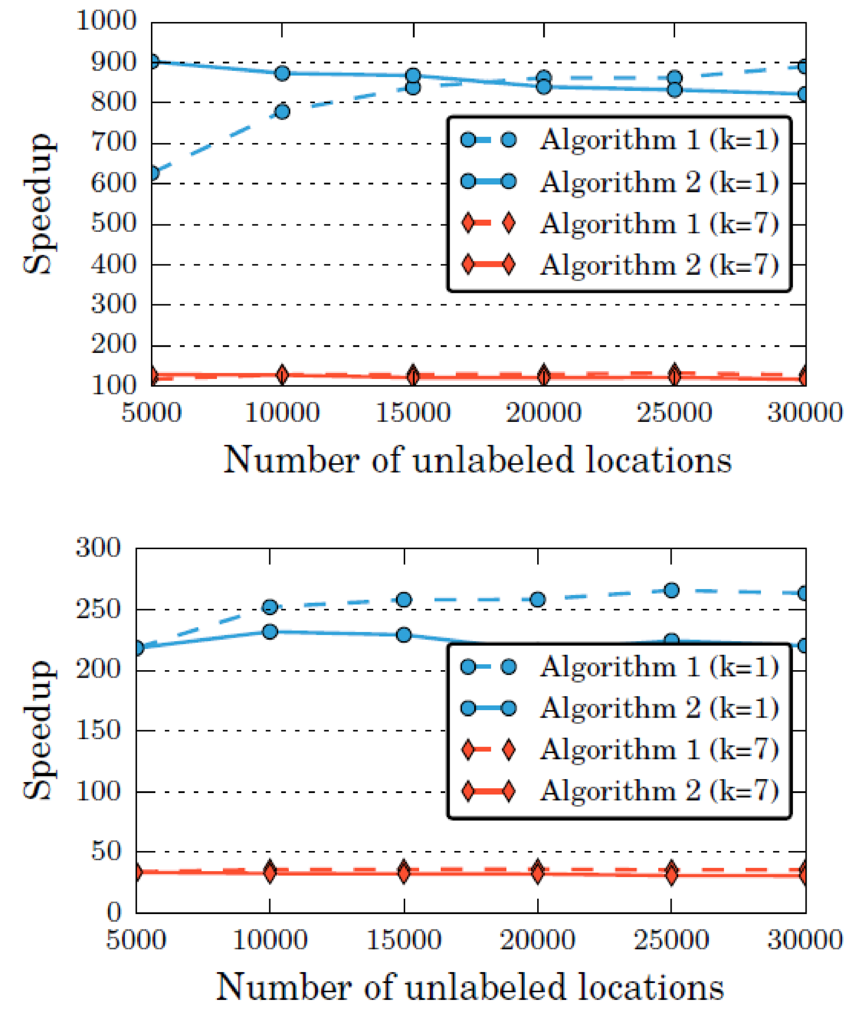

Figure 1: Geodesic 1NN and 7NN with Algorithms 2.1 and 2.2. (Top: 1600 labeled locations on 2m grid; Bottom: 400 labeled on 4m grid)

Applications

Regressor Used

- K nearest neighbor regressor (KNN)
- Geodesic KNN (GNN)
- Semi-supervised Laplacian regressor

Indoor Localization Using Wi-Fi Signals

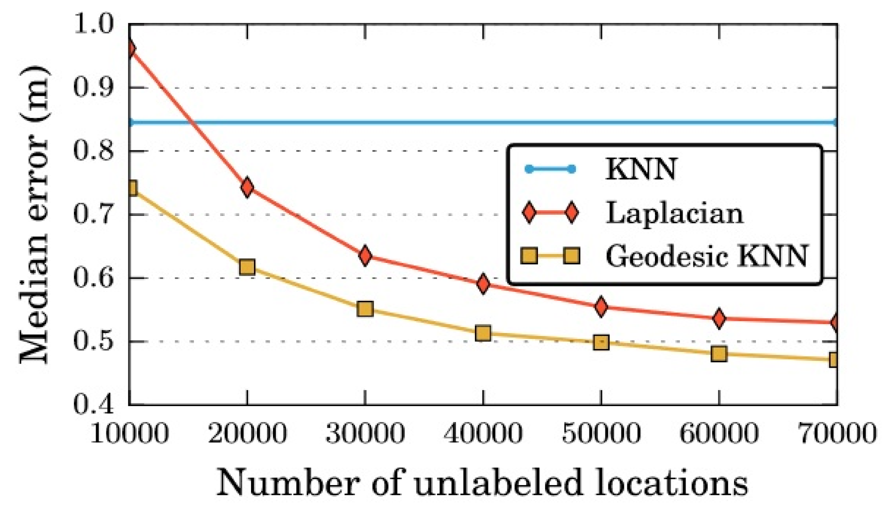

Figure 2: Median localization error vs. number of unlabeled points with 1600 labeled points placed on a regular simulated grid with a side length of 2m. [3]

Labeled grid	n	kNN	GNN	Laplacian
1.5m	73	1.49m	*1.11*m	1.36m
2.0m	48	2.27m	*1.49*m	1.65m
3.0m	23	3.41m	*2.41*m	2.79m

Table 1: Mean accuracy of kNN, geodesic kNN and Laplacian eigenbasis regression on the real data set. [3]

#unlabeled	GNN	Laplacian	graph build
1000	2.3s	7.6s	9s
10000	7s	195s	76s
100000	56s	114min	66min

Table 2: Runtime of Geodesic 7-NN vs. time to com-pute Laplacian eigenvectors. [3]

Facial Pose Estimation

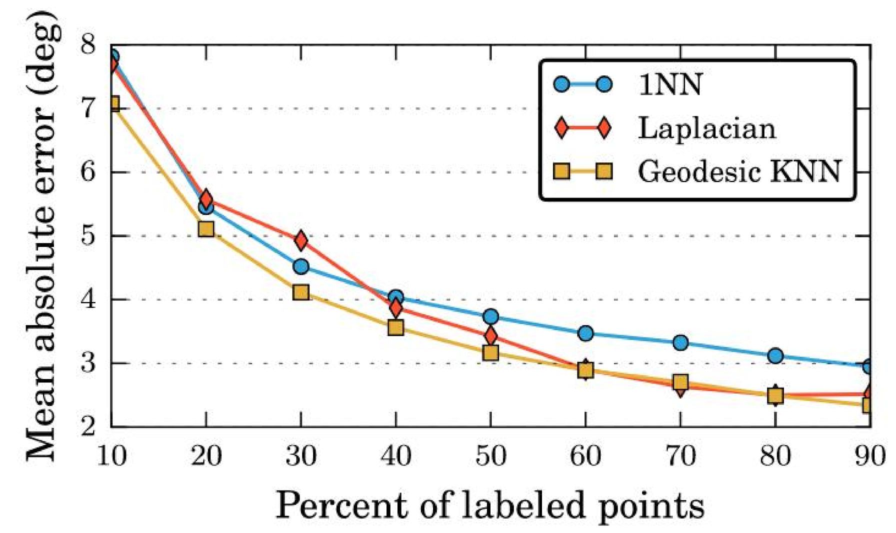

Figure 3: Mean prediction error for the left-rightangle of the face. [3]

References

[1] Peter J. Bickel and Bo Li. Local polynomial regression on unknown manifolds. *InTomography,Networks and Beyond, pages 177186. Institute of Mathematical Statistics*, 2007.

[2] John Lafferty and Larry Wasserman. Statistical analysis of semi-supervised regression. *InNeural Information Processing Systems (NIPS)*, 2007.

[3] Amit Moscovich, Ariel Jaffe, and Boaz Nadler. Minimax-optimal semi-supervised regression on unknown manifolds. *arXiv:1611.02221*, March 2017.

Acknowledgements

We would like to thank Amit Moscovich, Ariel Jaffe, and Boaz Nadler for methodology, algorithm, and empirical results provided in "Minimax-optimal semi-supervised regression on unknown manifolds".