Crystallization Learning with the Delaunay Triangulation

Jiaqi Gu [1] Guosheng Yin [1]

Abstract

Based on the Delaunay triangulation, we propose the crystallization learning to estimate the conditional expectation function in the framework of nonparametric regression. By conducting the crystallization search for the Delaunay simplices closest to the target point in a hierarchical way, the crystallization learning estimates the conditional expectation of the response by fitting a local linear model to the data points of the constructed Delaunay simplices. Instead of conducting the Delaunay triangulation for the entire feature space which would encounter enormous computational difficulty, our approach focuses only on the neighborhood of the target point and thus greatly expedites the estimation for high-dimensional cases. Because the volumes of Delaunay simplices are adaptive to the density of feature data points, our method selects neighbor data points uniformly in all directions and thus is more robust to the local geometric structure of the data than existing nonparametric regression methods. We develop the asymptotic properties of the crystallization learning and conduct numerical experiments on both synthetic and real data to demonstrate the advantages of our method in estimation of the conditional expectation function and prediction of the response.

1. Introduction

Consider a regression model,

$$y_i = \mu(\mathbf{x}_i) + \epsilon_i, \quad i = 1, \ldots, n, \tag{1}$$

where $\mathbf{x}_i$ is a d-dimensional feature point in the Euclidean space $\mathscr{R}^d$ $(n > d)$, y_i is the observed response, $\mu(\cdot) = E(Y|\cdot)$ is the conditional expectation function of the response Y, and $\epsilon_1, \ldots, \epsilon_n \in \mathscr{R}$ are independent and identically distributed (i.i.d.) random errors with $E(\epsilon_i) = 0$ and $E(\epsilon_i^2) < \infty$. Nonparametric regression is a collection of methods for estimating the conditional expectation function $\mu(\cdot)$ without rigid assumptions on its shape. Recent decades have witnessed extensive research in the field of nonparametric regression, including nearest-neighbor regression (Nadaraya, 1964; Watson, 1964; Cover & Hart, 1967; Benedetti, 1977; Stone, 1977; Altman, 1992), kernel regression (Priestley & Chao, 1972; Hardle & Gasser, 1984; Hein, 2009) and local linear regression (Cleveland, 1979; Cleveland & Devlin, 1988; Fan & Gijbels, 2018). Although the consistency of these methods has been established under mild conditions, their finite-sample performances are sensitive to the local geometric structure of observed feature points. As these methods only consider the distances from the target point $\mathbf{z}$ to observed feature points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in computing the neighbor data points or assigning weights, it is possible that the directions from $\mathbf{z}$ to its neighbors are not uniformly distributed, especially when $\mathbf{z}$ is close to the boundary of the convex hull of feature points or jump points of the feature data density. As a result, the (weighted) mean of neighbor data points may be far from the target point $\mathbf{z}$, leading to large bias in estimating the conditional expectation $\mu(\mathbf{z})$.

By incorporating the Delaunay triangulation (Delaunay, 1934) into the framework of nonparametric regression, we propose the crystallization learning which mimics the crystallization process in thermodynamics and circumvents the curse-of-dimensionality issue in the Delaunay triangulation. Based on the DELAUNAYSPARSE algorithm (Chang et al., 2020) which locally constructs the Delaunay simplex $\mathcal{S}(\mathbf{z})$ containing the target point $\mathbf{z}$, we develop the crystallization search for the Delaunay simplices closest to $\mathcal{S}(\mathbf{z})$ and estimate $\mu(\mathbf{z})$ by fitting a local linear model to the data points of the obtained Delaunay simplices. Via experiments on synthetic and real data, our method is shown to outperform the existing ones in estimating the conditional expectation function and predicting the response.

2. Methodology

2.1. Delaunay Interpolation

Let $\mathbb{X}$ be a set of n feature points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in the Euclidean space $\mathscr{R}^d$ $(n > d)$. A d-dimensional triangulation of $\mathbb{X}$, $\mathcal{T}(\mathbb{X})$, is a mesh of d-simplices $\{\mathcal{S}_1, \ldots, \mathcal{S}_m\}$ satisfying:

[1]Department of Statistics and Actuarial Science, University of Hong Kong, Hong Kong SAR. Correspondence to: Jiaqi Gu <u3005743@hku.hk>, Guosheng Yin <gyin@hku.hk>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021. Copyright 2021 by the author(s).

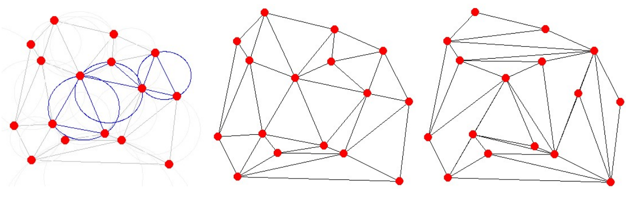


Figure 1. (a) Graphical illustration of the empty-ball property of the Delaunay triangulation; (b) the Delaunay triangulation; (c) a random triangulation.

1. For $j = 1, \dots, m$, the set of $d+1$ vertices of simplex $\mathcal{S}_j$, denoted as $\mathbb{V}(\mathcal{S}_j)$, is a subset of $\mathbb{X}$ and does not lie in any affine hyperplane of $\mathscr{R}^d$.
2. For any $j \neq k$, simplices $\mathcal{S}_j$ and $\mathcal{S}_k$ are disjoint except on their shared boundaries $\mathcal{S}_j \cap \mathcal{S}_k$.
3. The union $\mathcal{S}_1 \cup \dots \cup \mathcal{S}_m$ is the convex hull of $\mathbb{X}$, $\mathcal{H}(\mathbb{X})$.

As the d-simplices $\mathcal{S}_1, \dots, \mathcal{S}_m$ of the triangulation $\mathcal{T}(\mathbb{X})$ fully cover the convex hull $\mathcal{H}(\mathbb{X})$, for each internal point $\mathbf{z} \in \mathcal{H}(\mathbb{X})$, there exists a simplex $\mathcal{S}(\mathbf{z}) \in \mathcal{T}(\mathbb{X})$ such that $\mathbf{z} \in \mathcal{S}(\mathbf{z})$. Let $i_1(\mathbf{z}), \dots, i_{d+1}(\mathbf{z})$ denote the indices corresponding to the data points of $\mathcal{S}(\mathbf{z})$, and then there exist $d+1$ values $\gamma_1, \dots, \gamma_{d+1} \in [0, 1]$ such that $\sum_{k=1}^{d+1} \gamma_k \mathbf{x}_{i_k(\mathbf{z})} = \mathbf{z}$ and $\sum_{k=1}^{d+1} \gamma_k = 1$. Among all triangulations, the Delaunay triangulation has been widely used for multivariate interpolation (de Berg et al., 2008) due to its smoothness property. Let $\mathcal{B}_j$ be the open ball whose boundary is the circumscribed $(d-1)$-sphere of $\mathcal{S}_j$. The Delaunay triangulation of $\mathbb{X}$, denoted as $\mathcal{DT}(\mathbb{X})$, is a triangulation of $\mathbb{X}$ such that $\mathcal{B}_j \cap \mathbb{X} = \varnothing$ for $j = 1, \dots, m$. This is known as the empty-ball property as shown in Figure 1 (a). As the geometric dual of the Voronoi diagram under the L_2 norm, the Delaunay triangulation generates a mesh of simplices that are most regularized in shape. In a 2-dimensional space, $\mathbb{X} \subset \mathscr{R}^2$, the Delaunay triangulation $\mathcal{DT}(\mathbb{X})$ maximizes the minimum angle in all the triangles (2-simplices) $\mathcal{S}_1, \dots, \mathcal{S}_m$ over all possible triangulations (Sibson, 1978), as shown in Figure 1 (b) and (c). The Delaunay triangulation $\mathcal{DT}(\mathbb{X})$ is unique under the assumption that $\mathbb{X}$ is in general position (Delaunay, 1934).

Considering the data $\{(\mathbf{x}_i, y_i) : i = 1, \dots, n\}$ from model (1), the Delaunay interpolation aims to estimate the conditional expectation $\mu(\mathbf{z})$ for all $\mathbf{z} \in \mathcal{H}(\mathbb{X})$. Generally, there are three steps in the Delaunay interpolation: (i) construct the Delaunay triangulation $\mathcal{DT}(\mathbb{X})$; (ii) find the simplex $\mathcal{S}(\mathbf{z}) \in \mathcal{DT}(\mathbb{X})$; and (iii) obtain the estimator $\hat{\mu}(\mathbf{z})$ by optimizing a target function. For most of Delaunay interpolation methods, the first two steps are the same, while the difference mainly lies in the target function. For example, with $\gamma_1, \dots, \gamma_{d+1} \in [0, 1]$ such that $\sum_{k=1}^{d+1} \gamma_k \mathbf{x}_{i_k(\mathbf{z})} = \mathbf{z}$ and

Algorithm 1 DELAUNAYSPARSE (Chang et al., 2020)

```
1:  Input: Feature points $\mathbb{X}$, target point $\mathbf{z} \in \mathcal{H}(\mathbb{X})$ and the
    seed Delaunay simplex $\mathcal{S}_{\text{seed}}$.
2:  Let $\mathcal{S}_{\text{current}} = \mathcal{S}_{\text{seed}}$, $\mathbb{A}_{\text{Frontier}} = \{\mathcal{S}_{\text{seed}}\}$, $\mathbb{A}_{\text{Explored}} = \varnothing$.
3:  while $\mathbf{z} \notin \mathcal{S}_{\text{current}}$ do
4:    Compute the set of facets of $\mathcal{S}_{\text{current}}$ which is visible
      to $\mathbf{z}$[1], denoted as $\mathbb{F}_{\mathbf{z}}(\mathcal{S}_{\text{current}})$.
5:    for each facet $\mathcal{F} \in \mathbb{F}_{\mathbf{z}}(\mathcal{S}_{\text{current}})$ do
6:      Grow a new Delaunay simplex $\mathcal{S}_{\text{new}} \neq \mathcal{S}_{\text{current}}$ on
        the facet $\mathcal{F}$ if it exists.
7:      $\mathbb{A}_{\text{Frontier}} \leftarrow \mathbb{A}_{\text{Frontier}} \cup \{\mathcal{S}_{\text{new}}\}$ if $\mathcal{S}_{\text{new}}$ exists and
        $\mathcal{S}_{\text{new}} \notin \mathbb{A}_{\text{Explored}} \cup \mathbb{A}_{\text{Frontier}}$.
8:    end for
9:    $\mathbb{A}_{\text{Explored}} \leftarrow \mathbb{A}_{\text{Explored}} \cup \{\mathcal{S}_{\text{current}}\}$.
10:   $\mathbb{A}_{\text{Frontier}} \leftarrow \mathbb{A}_{\text{Frontier}} \setminus \{\mathcal{S}_{\text{current}}\}$.
11:   $\mathcal{S}_{\text{current}} \leftarrow$ the first simplex in $\mathbb{A}_{\text{Frontier}}$.
12: end while
13: Output: Simplex $\mathcal{S}_{\text{current}}$.
```

$\sum_{k=1}^{d+1} \gamma_k = 1$, the estimator of de Berg et al. (2008) is

$$\hat{\mu}(\mathbf{z}) = \sum_{k=1}^{d+1} \gamma_k y_{i_k(\mathbf{z})}, \tag{2}$$

which is the minimizer of the squared loss function $\sum_{i=1}^{n} (y_i - g(\mathbf{x}_i))^2$ among all continuous piecewise linear functions, $g(\mathbf{x}) = \sum_{j=1}^{m} 1_{\{\mathbf{x} \in \mathcal{S}_j\}} (\alpha_j + \boldsymbol{\beta}_j^{\mathsf{T}} \mathbf{x})$. Liu & Yin (2020) introduce a regularization function to balance the model fitting and smoothness of the estimator. However, all the aforementioned methods require a complete construction of $\mathcal{DT}(\mathbb{X})$ for the entire feature space, whose size (i.e., m) grows exponentially with the dimension d. As a result, no existing algorithm is feasible when $d > 7$ due to the limitations of computation time/power and memory space (Chang et al., 2020).

Alternatively, several methods have been proposed for medium- to high-dimensional Delaunay interpolation (Chang et al., 2018a;b; 2020). Instead of obtaining the complete $\mathcal{DT}(\mathbb{X})$, these methods only construct the Delaunay simplex $\mathcal{S}(\mathbf{z})$ at each target point $\mathbf{z}$ locally and thus $\mu(\mathbf{z})$ can be estimated at a polynomial cost. For any point $\mathbf{z} \in \mathcal{H}(\mathbb{X})$, the DELAUNAYSPARSE algorithm (Chang et al., 2020) first obtains a seed Delaunay simplex $\mathcal{S}_{\text{seed}}$ close to $\mathbf{z}$. Based on $\mathcal{S}_{\text{seed}}$, Chang et al. (2020) find $\mathcal{S}(\mathbf{z})$ via the breadth first search as described in Algorithm 1 and compute the estimator in (2). Although such an approach is computationally efficient because $\gamma_1, \dots, \gamma_{d+1}$ are simultaneously calculated, it only utilizes the information of $d+1$ data points $\{(\mathbf{x}_{i_k(\mathbf{z})}, y_{i_k(\mathbf{z})}) : k = 1, \dots, d+1\}$ in the esti-

[1]A facet $\mathcal{F}$ of the simplex $\mathcal{S}_{\text{current}}$ is visible to $\mathbf{z}$ if there exists an internal point $\mathbf{z}'$ of $\mathcal{S}_{\text{current}}$ such that the linear segment from $\mathbf{z}$ to $\mathbf{z}'$ intersects $\mathcal{F}$ (Chang et al., 2020).

Algorithm 2 Crystallization search

```
1: Input: Feature points $\mathbb{X}$, target point $\mathbf{z} \in \mathcal{H}(\mathbb{X})$ and topological distance $L$.
2: Compute $\mathcal{S}(\mathbf{z})$ via Algorithm 1.
3: Let $\mathbb{A}_{\text{Frontier}} = \{(\mathcal{S}(\mathbf{z}), 0)\}$ and $\mathcal{N}_L(\mathbf{z}) = \varnothing$.
4: while $\mathbb{A}_{\text{Frontier}} \neq \varnothing$ do
5:   $(\mathcal{S}_{\text{current}}, L_{\text{current}}) \leftarrow$ the first element in $\mathbb{A}_{\text{Frontier}}$.
6:   if $L_{\text{current}} < L$ then
7:     Compute all the facets of $\mathcal{S}_{\text{current}}$, denoted as $\mathcal{F}_1, \dots, \mathcal{F}_{d+1}$.
8:     for $j = 1, \dots, d+1$ do
9:       Grow a new Delaunay simplex $\mathcal{S}_{\text{new}} \neq \mathcal{S}_{\text{current}}$ on the facet $\mathcal{F}_j$ if it exists.
10:      $\mathbb{A}_{\text{Frontier}} \leftarrow \mathbb{A}_{\text{Frontier}} \cup \{(\mathcal{S}_{\text{new}}, L_{\text{current}} + 1)\}$ if $\mathcal{S}_{\text{new}}$ exists and $\mathcal{S}_{\text{new}} \notin \mathcal{N}_L(\mathbf{z}) \cup \mathbb{A}_{\text{Frontier}}$.
11:    end for
12:  end if
13:  $\mathcal{N}_L(\mathbf{z}) \leftarrow \mathcal{N}_L(\mathbf{z}) \cup \{\mathcal{S}_{\text{current}}\}$.
14:  $\mathbb{A}_{\text{Frontier}} \leftarrow \mathbb{A}_{\text{Frontier}} \setminus \{(\mathcal{S}_{\text{current}}, L_{\text{current}})\}$.
15: end while
16: Output: The set of Delaunay simplices $\mathcal{N}_L(\mathbf{z})$.
```

mation. This may lead to overfitting and poor estimation when the simplex $\mathcal{S}(\mathbf{z})$ has a small volume and a poorly regularized shape.

2.2. Crystallization Search for Delaunay Simplices

As one component of $\mathcal{DT}(\mathbb{X})$, $\mathcal{S}(\mathbf{z})$ has $d+1$ facets $\mathcal{F}_1, \dots, \mathcal{F}_{d+1}$, each of which is either a facet of $\mathcal{H}(\mathbb{X})$ or the shared boundary of $\mathcal{S}(\mathbf{z})$ and one neighbor Delaunay simplex.

Definition 1. ***Neighbor Delaunay simplices:*** *Given a set of points $\mathbb{X}$ and the Delaunay triangulation $\mathcal{DT}(\mathbb{X}) = \{\mathcal{S}_1, \dots, \mathcal{S}_m\}$, simplices $\mathcal{S}_j$ and $\mathcal{S}_k$ are neighbors if and only if the intersection $\mathcal{S}_j \cap \mathcal{S}_k$ is a shared facet of $\mathcal{S}_j$ and $\mathcal{S}_k$.*

Inspired by Algorithm 1 (Chang et al., 2020) which searches $\mathcal{S}(\mathbf{z})$ by growing neighbor Delaunay simplices on the facets of the explored ones recursively, we develop the crystallization search (Algorithm 2) to construct all the Delaunay simplices within the topological distance L to $\mathcal{S}(\mathbf{z})$, denoted as $\mathcal{N}_L(\mathbf{z})$. Figures 2 and 3 display the crystallization search of $\mathcal{N}_L(\mathbf{z})$ with respect to a target point $\mathbf{z} \in \mathcal{H}(\mathbb{X})$ for $L = 0, 1, \dots, 5$ in $\mathscr{R}^2$ and $\mathscr{R}^3$, respectively. When $L = 0$, only the simplex $\mathcal{S}(\mathbf{z})$ is constructed. As L increases, Delaunay simplices are constructed in a hierarchical way, such that new simplices would grow on the facets of the explored ones sequentially. The whole process of crystallization search is analogous to the crystallization process in thermodynamics, where the search of $\mathcal{S}(\mathbf{z})$ indexed by line 2 in Algorithm 2 plays the role of nucleation and the remaining steps correspond to crystal growth.

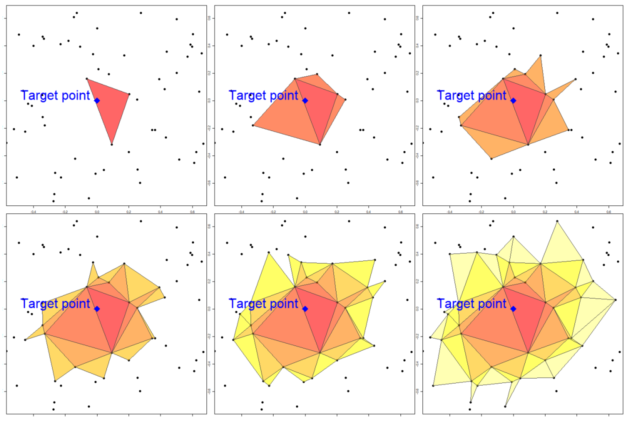


Figure 2. Crystallization search of $\mathcal{N}_L(\mathbf{z})$ with respect to a target point $\mathbf{z} \in \mathcal{H}(\mathbb{X})$ for $L = 0, 1, 2$ (top row) and $L = 3, 4, 5$ (bottom row) in $\mathscr{R}^2$.

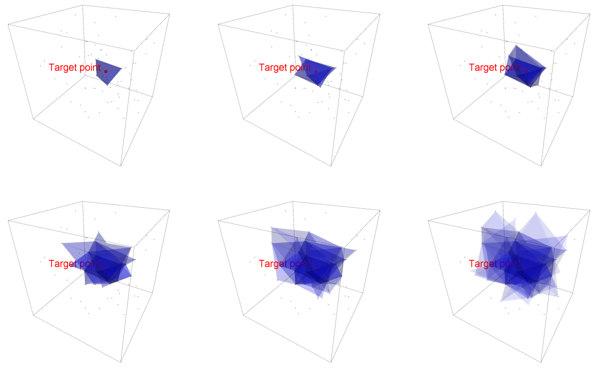


Figure 3. Crystallization search of $\mathcal{N}_L(\mathbf{z})$ with respect to a target point $\mathbf{z} \in \mathcal{H}(\mathbb{X})$ for $L = 0, 1, 2$ (top row) and $L = 3, 4, 5$ (bottom row) in $\mathscr{R}^3$.

2.3. Crystallization Learning

Without loss of generality, let $\mathbb{V}_{\mathbf{z},L} = \cup_{\mathcal{S} \in \mathcal{N}_L(\mathbf{z})} \mathbb{V}(\mathcal{S})$ denote the set of all the data points of the simplices in $\mathcal{N}_L(\mathbf{z})$. Based on the set $\mathcal{N}_L(\mathbf{z})$ of Delaunay simplices topologically closest to the target point $\mathbf{z}$, we propose the crystallization learning to estimate $\mu(\mathbf{z})$ by fitting a local linear model, $\mu(\mathbf{z}) = \alpha + \boldsymbol{\beta}^{\mathsf{T}}\mathbf{z}$, to all the data points in $\mathbb{V}_{\mathbf{z},L}$ instead of only the $d+1$ data points of $\mathcal{S}(\mathbf{z})$. In $\mathcal{DT}(\mathbb{X})$, a vertex shared by more simplices typically has a larger degree in the network formed by Delaunay edges and thus is more informative in the geometric structure of $\mathcal{N}_L(\mathbf{z})$. We estimate α and $\boldsymbol{\beta}$ via the weighted least squares approach,

$$(\hat{\alpha}, \hat{\boldsymbol{\beta}}) = \arg\min_{\alpha, \boldsymbol{\beta}} \sum_{\mathbf{x}_i \in \mathbb{V}_{\mathbf{z},L}} \mathrm{w}_{\mathbf{z},L}(\mathbf{x}_i)(y_i - \alpha - \boldsymbol{\beta}^{\mathsf{T}}\mathbf{x}_i)^2, \quad (3)$$

where the weight function is

$$\mathrm{w}_{\mathbf{z},L}(\mathbf{x}_i) = \left(\sum_{\mathcal{S} \in \mathcal{N}_L(\mathbf{z})} 1_{\{\mathbf{x}_i \in \mathbb{V}(\mathcal{S})\}} \right) \exp\left(- \frac{\|\mathbf{x}_i - \mathbf{z}\|_2^2}{m_L(\mathbf{z})} \right),$$

$$\text{with } m_L(\mathbf{z}) = \left(\sum_{\mathbf{x}_i \in \mathbb{V}_{\mathbf{z},L}} \|\mathbf{x}_i - \mathbf{z}\|_2^2 \right) \Big/ \left(\sum_{i=1}^{n} 1_{\{\mathbf{x}_i \in \mathbb{V}_{\mathbf{z},L}\}} \right).$$

Given the estimators $\hat{\alpha}$ and $\hat{\boldsymbol{\beta}}$, we have $\hat{\mu}(\mathbf{z}) = \hat{\alpha} + \hat{\boldsymbol{\beta}}^{\mathsf{T}}\mathbf{z}$. Similar to the work of Nadaraya (1964) and Watson (1964), our weight function places more weights on the data points closer to $\mathbf{z}$ as well as those shared by more simplices in $\mathcal{N}_L(\mathbf{z})$. For all $\mathbf{x}_i \notin \mathbb{V}_{\mathbf{z},L}$, the weights are set to be zero. In addition, our weight function is scale-invariant due to the existence of the normalization term $m_L(\mathbf{z})$, i.e., multiplying any constant to features would not change the weights. As a result, the estimated conditional expectation function, $\hat{\mu}(\cdot)$, is only piecewise smooth but not piecewise linear in $\mathcal{H}(\mathbb{X})$, as demonstrated by Theorem 1 with the proof given in the supplementary materials.

Theorem 1. *Let $\mathbb{X}$ be a set of n feature points $\mathbf{x}_1, \ldots, \mathbf{x}_n$ in general position and responses $y_1, \ldots, y_n$ are generated from model (1). The estimated conditional expectation function under crystallization learning, $\hat{\mu}(\cdot)$, is smooth in $\mathcal{S}_k$, for $k = 1, \ldots, m$.*

2.4. Selection of L

Similar to many other machine learning methods, the statistical complexity and estimation performance of the crystallization learning is controlled by the hyperparameter L, the maximal topological distance from the generated neighbor Delaunay simplices to $\mathcal{S}(\mathbf{z})$. Because a small L leads to overfitting and a large L makes $\hat{\mu}(\cdot)$ overly smooth, we propose adopting the leave-one-out cross validation (LOO-CV) to select L with respect to the target point $\mathbf{z}$ as follows.

1. Compute the Delaunay simplex $\mathcal{S}(\mathbf{z})$ containing $\mathbf{z}$ and the values of $\gamma_1, \ldots, \gamma_{d+1} \in [0,1]$ such that $\sum_{k=1}^{d+1} \gamma_k \mathbf{x}_{i_k(\mathbf{z})} = \mathbf{z}$ and $\sum_{k=1}^{d+1} \gamma_k = 1$ via Algorithm 1, where $\mathbf{x}_{i_1(\mathbf{z})}, \ldots, \mathbf{x}_{i_{d+1}(\mathbf{z})}$ are the $d+1$ data points of $\mathcal{S}(\mathbf{z})$.

2. For each $\mathbf{x}_{i_k(\mathbf{z})} \in \mathcal{S}(\mathbf{z})$, apply the crystallization learning with different candidate values of L on the leave-one-out data excluding $(\mathbf{x}_{i_k(\mathbf{z})}, y_{i_k(\mathbf{z})})$ to estimate $\mu(\mathbf{x}_{i_k(\mathbf{z})})$. Let $\hat{\mu}(\mathbf{x}_{i_k(\mathbf{z})}; L)$ be the estimator corresponding to the observation $(\mathbf{x}_{i_k(\mathbf{z})}, y_{i_k(\mathbf{z})})$ and candidate value L.

3. Select the optimal $\widetilde{L}$ as

$$\widetilde{L} = \arg\min_{L} \sum_{k=1}^{d+1} \gamma_k \log\{\hat{\mu}(\mathbf{x}_{i_k(\mathbf{z})}; L) - y_{i_k(\mathbf{z})}\}^2.$$

2.5. Computational Complexity

As shown by Chang et al. (2020), the average computational complexity of Algorithm 1 is $\mathcal{O}(d^2 n)$. However, as Algorithm 1 is only implemented once, the dominant cost of Algorithm 2 lies in growing simplices (lines 4–15). In Algorithm 2, the number of generated Delaunay simplices is $\mathcal{O}(d^L)$ and the average computational complexity of growing a new Delaunay simplex on the facet of $\mathcal{S}_{\text{current}}$ is $\mathcal{O}(n)$ with the rank-1 update suggested by Chang et al. (2020). Thus, the average computational complexity of Algorithm 2 is $\mathcal{O}(d^L n)$. Table 1 shows the average runtime in computing $\mathcal{N}_L(\mathbf{z})$ under different configurations.

Table 1. Average runtime (in seconds) in computing $\mathcal{N}_L(\mathbf{z})$ under different values of the maximal topological distance L, sample size n, dimension d.

L	$n = 500$			$n = 1000$			$n = 2000$		
	$d = 6$	8	10	$d = 6$	8	10	$d = 6$	8	10
2	0.05	0.09	0.14	0.06	0.11	0.18	0.07	0.14	0.23
3	0.23	0.51	0.98	0.28	0.63	1.22	0.34	0.80	1.56
4	0.82	2.26	5.20	1.02	2.80	6.51	1.21	3.55	8.27

3. Connection with Other Nonparametric Regression Methods

Similar to the two popular nonparametric regression methods, i.e., the nearest neighbor and the local linear regression, our crystallization learning consists of three steps in estimating the conditional expectation $\mu(\mathbf{z})$ at the target point $\mathbf{z}$: (i) selecting data points from $\mathbb{X}$ as the neighbors of $\mathbf{z}$ according to a specific criterion; (ii) assigning weights to the selected neighbor data points; and (iii) fitting a local model to the selected neighbor data points. As the crystallization learning and existing methods mainly differ in the first two steps, we compare our method with the k-nearest neighbor (k-NN) regression and the local linear regression in selecting the neighbor data points. Our crystallization search of neighbor data points is based on the Delaunay triangulation, which is the geometric dual of the Voronoi diagram under the L_2 norm. As a result, we use the Euclidean distance in k-NN and the Gaussian kernel in the local linear regression.

To find the k nearest neighbor data points, the k-NN regression computes and sorts the Euclidean distances from the target point $\mathbf{z}$ to all the data points in $\mathbb{X}$. This process can be visualized by the left panel of Figure 4, where a circle with center $\mathbf{z}$ is shown. The radius keeps increasing until there are k observed data points falling inside or on the circle, which are returned as the k nearest neighbor points. Because only the Euclidean distance is considered, it is likely that the directions from $\mathbf{z}$ to the k nearest neighbor points are not uniformly distributed, especially when $\mathbf{z}$ is close to the boundary of $\mathcal{H}(\mathbb{X})$ or jump points of the feature data

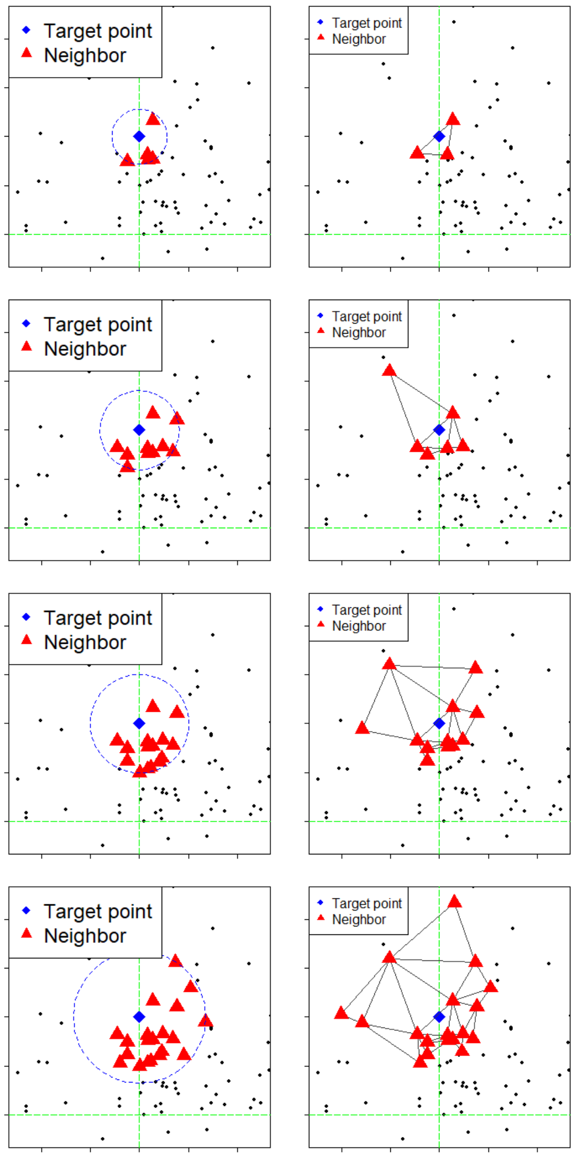
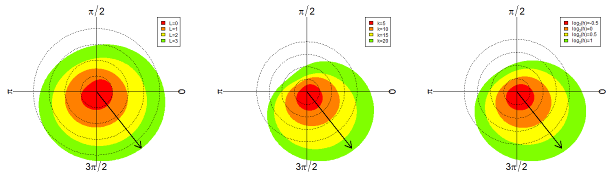


Figure 4. Neighbor data points of the target point $\mathbf{z}$ selected by the k-NN regression with $k = 5, 10, 15, 20$ (left panel) and the crystallization learning with $L = 0, 1, 2, 3$ (right panel).



(a) Crystallization (b) k-NN (c) Local linear

Figure 5. Kernel density estimates of distributions of the directions from the target point $\mathbf{z}$ to its neighbor data points using different methods with different hyperparameter values. The arrow indicates the direction from the target point $\mathbf{z}$ to the sample mean of $\mathbb{X}$.

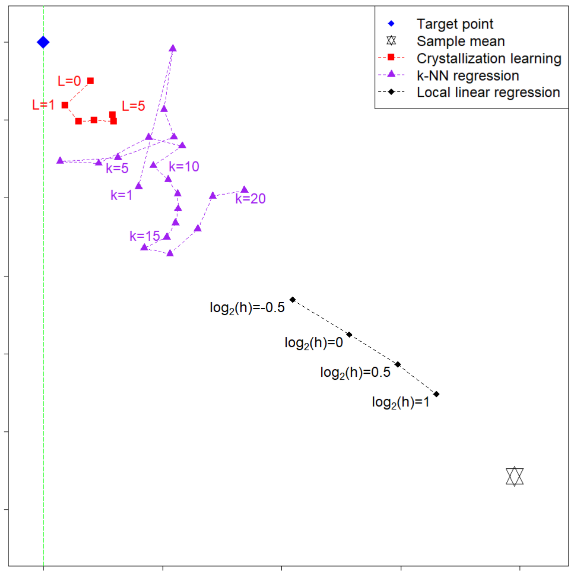


Figure 6. Paths of the (weighted) means of neighbor data points by different methods as the value of the hyperparameter increases under feature data density (4).

density. The same is true for the local linear regression, where more weights are assigned in the direction toward the sample mean. In contrast, the crystallization search identifies neighbor data points $\mathbb{V}_{\mathbf{z},L}$ by constructing Delaunay simplices, whose volumes are adaptive to the density of observed feature data. As a result, the distances from $\mathbf{z}$ to neighbor data points in $\mathbb{V}_{\mathbf{z},L}$ are different for high-density and low-density directions. This can be seen from Figure 4, where we generate $\mathbf{x}_1, \ldots, \mathbf{x}_{100} \in \mathscr{R}^2$ from the density function,

$$f(\mathbf{x}) \propto \prod_{j=1}^{2} \{1 + 0.6 \cdot \text{sign}(x_j)\} \exp(-x_j^2/2), \qquad (4)$$

and use different methods to select the neighbor data points of $\mathbf{z} = (0, 1)^\mathsf{T}$. The density function $f(\mathbf{x})$ is discontinuous at $\mathbf{z}$ with higher density at its right-hand side than its left-hand side. From the left panel of Figure 4, we can see that for all values of k, k-NN identifies more neighbor points at the right-hand side of $\mathbf{z}$ than the left-hand side. However, this is not the case for the crystallization learning as exhibited in the right panel of Figure 4. As L increases, the crystallization learning searches neighbor data points uniformly in all directions, implying its adaptation to the local geometric structure of the data. This can also be observed in Figure 5, where kernel density estimates (KDEs) of distributions of the directions from the target point $\mathbf{z}$ to its neighbor data points are plotted. The neighbor data points selected by k-NN and the weights assigned by the local linear regression concentrate in the direction toward the sample mean of $\mathbb{X}$, while KDEs of the crystallization search are much closer to the uniform distribution. As a result, the (weighted) means of neighbor data points under the crystallization search are closer to the target point $\mathbf{z}$ than existing methods as shown in Figure 6.

4. Asymptotic Theory

We first study the asymptotic geometric properties of the Delaunay triangulation $\mathcal{DT}(\mathbb{X})$ under general distribution of feature points and then prove the consistency of the crystallization learning in estimating $\mu(\cdot)$. All proofs are given in the supplementary materials.

4.1. Asymptotic Geometric Properties

Let $\mathbb{X}$ be a set of n i.i.d. feature data points $\mathbf{x}_1, \dots, \mathbf{x}_n \in \mathscr{R}^d$ from a density $f(\mathbf{x})$, which is bounded away from zero and infinity on $\mathscr{R}^d$.

Lemma 1. *For any target point $\mathbf{z} \in \mathscr{R}^d$, we have that $\mathbb{P}(\mathbf{z} \in \mathcal{H}(\mathbb{X})) \to 1$, as $n \to \infty$.*

By Lemma 1, the target point $\mathbf{z}$ falls inside $\mathcal{H}(\mathbb{X})$ with asymptotic probability one, and thus we only need to consider the inside-hull case.

Theorem 2. *For any target point $\mathbf{z} \in \mathcal{H}(\mathbb{X})$ and any $\rho \in (0,1)$, we have*

$$T(\mathbf{z}) = O_p(n^{-\rho/d}),$$

where $T(\mathbf{z}) = \max\{\|\mathbf{x}_i - \mathbf{z}\|_2; \mathbf{x}_i \in \mathbb{V}_{\mathbf{z},L}\}$ is the maximal L_2 norm between $\mathbf{z}$ and its neighbor feature data points.

Theorem 2 implies that all the feature points of $\mathbb{V}_{\mathbf{z},L}$ converge to $\mathbf{z}$ in probability.

4.2. Consistency

Theorem 3. *Assume the data $\{(\mathbf{x}_i, y_i) : i = 1, \dots, n\}$ are generated from model (1), where the conditional expectation function $\mu(\cdot)$ is differentiable on $\mathscr{R}^d$. The estimated conditional expectation function under crystallization learning, $\hat{\mu}(\cdot)$, satisfies*

$$E\{\hat{\mu}(\mathbf{z}) - \mu(\mathbf{z})\}^2 \to R_{\min}, \quad \text{as } n \to \infty,$$

for all $\mathbf{z} \in \mathcal{H}(\mathbb{X})$ where $R_{\min} = \inf_g E\{Y - g(\mathbf{x})\}^2$ is the minimal value of the L_2 risk over all continuous functions $g : \mathscr{R}^d \to \mathscr{R}$.

5. Numerical Experiments

We conduct experiments on synthetic data under two different scenarios: (i) to illustrate the effectiveness of the crystallization learning in estimating the conditional expectation function $\mu(\cdot)$; (ii) to evaluate the estimation accuracy of our approach in comparison with existing nonparametric regression methods, including the k-NN regression using the Euclidean distance, local linear regression using Gaussian kernel, multivariate kernel regression using Gaussian kernel (Hein, 2009) and Gaussian process models; and (iii) to validate the proposed data-driven procedure for selection of the hyperparameter L. We also apply our method to real data to investigate its empirical performance.

5.1. Synthetic Data

In the experiments on synthetic data, we consider two scenarios to investigate the performance of our crystallization learning: (1) general internal points of $\mathcal{H}(\mathbb{X})$, and (2) jump points of the feature data density. For each scenario, we simulate 100 training datasets $\{(\mathbf{x}_i, y_i) : i = 1, \dots, n\}$ and randomly generate the corresponding sets of target points $\{\mathbf{z}_1, \dots, \mathbf{z}_{100}\}$ under different values of n and d. We use the mean squared error (MSE) under the method $\mathcal{M}$,

$$\text{MSE}_{\mathcal{M}} = \frac{1}{100}\sum_{k=1}^{100}\{\hat{\mu}_{\mathcal{M}}(\mathbf{z}_k) - \mu(\mathbf{z}_k)\}^2,$$

to evaluate the accuracy of the estimator $\hat{\mu}_{\mathcal{M}}(\cdot)$ at the target points $\mathbf{z}_1, \dots, \mathbf{z}_{100} \in \mathcal{H}(\mathbb{X})$.

Scenario 1 (General internal points): For each dataset, $\mathbf{x}_1, \dots, \mathbf{x}_n$ are independently sampled from the multivariate normal distribution $\text{MVN}(\mathbf{0}, \mathbf{I}_d)$ with an identity covariance matrix $\mathbf{I}_d$. The responses $y_1, \dots, y_n$ are generated from an additive model,

$$Y|\mathbf{x} \sim N\left(\sum_{j=1}^{d} c_j g_j(x_j), 1\right), \tag{5}$$

where $\mathbf{x} = (x_1, \dots, x_d)^\mathsf{T}$, $c_1, \dots, c_d \sim N(0,1)$, $g_j(\cdot) = \sum_{l=1}^{10} b_{jl}\phi(\cdot; \nu_{jl}, \sigma_{jl}^2)$, $b_{jl} \sim N(0,1)$, $\nu_{jl} \sim N(0,1)$, $\sigma_{jl}^2 \sim \text{Gamma}(1,1)$, and $\phi(\cdot; \nu_{jl}, \sigma_{jl}^2)$ is the density of a normal distribution $N(\nu_{jl}, \sigma_{jl}^2)$, for $j = 1, \dots, d; l = 1, \dots, 10$. For $k = 1, \dots, 100$, the target point $\mathbf{z}_k$ is generated as $\mathbf{z}_k = \sum_{i=1}^{n} \omega_{ik}\mathbf{x}_i$, with $(\omega_{1k}, \dots, \omega_{nk}) \sim \text{Dirichlet}(1, \dots, 1)$.

Scenario 2 (Jump points of the feature data density): For each dataset, $\mathbf{x}_1, \dots, \mathbf{x}_n$ are sampled from the density,

$$f(\mathbf{x}) = 2^{-d}\prod_{j=1}^{d}(1 + 0.4 \cdot \text{sign}(x_j))\exp(-|x_j|),$$

which has jumps at the point set $\{\mathbf{x} \in \mathscr{R}^d : \prod_{j=1}^{d} x_j = 0\}$. Responses $y_1, \dots, y_n$ are generated from the same additive model in (5). For $k = 1, \dots, 100$, the target point $\mathbf{z}_k$ is generated as $\mathbf{z}_k = \sum_{i=1}^{n} \omega_{ik}\mathbf{x}_i \otimes \mathbf{s}_k$, where $(\omega_{1k}, \dots, \omega_{nk}) \sim \text{Dirichlet}(1, \dots, 1)$, $\otimes$ is the element-wise multiplication operator, $\mathbf{s}_k = (s_{k1}, \dots, s_{kj})^\mathsf{T}$ and $s_{k1}, \dots, s_{kj} \sim \text{Bernoulli}(0.7)$.

Table 2. Averaged values of $\log(\text{MSE})$ and standard deviations in parentheses using crystallization learning (CL) in comparison with k-NN ($k = 5, 10, k^*$, where k^* equals the size of $\mathbb{V}_{\mathbf{z},L}$), local linear (LL) regression, kernel regression (KR) and Gaussian process (GP) in estimating $\mu(\cdot)$ under two scenarios, different sample sizes (n) and different dimensions of the feature space (d).

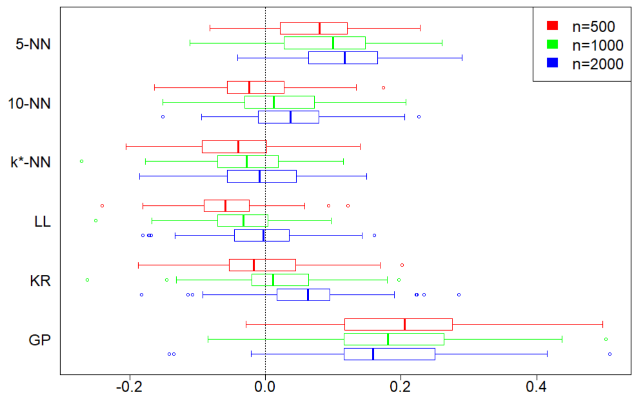
d	n	$\log(\text{MSE}_{\text{CL}})$	$\log\left(\frac{\text{MSE}_{5\text{-NN}}}{\text{MSE}_{\text{CL}}}\right)$	$\log\left(\frac{\text{MSE}_{10\text{-NN}}}{\text{MSE}_{\text{CL}}}\right)$	$\log\left(\frac{\text{MSE}_{k^*\text{-NN}}}{\text{MSE}_{\text{CL}}}\right)$	$\log\left(\frac{\text{MSE}_{\text{LL}}}{\text{MSE}_{\text{CL}}}\right)$	$\log\left(\frac{\text{MSE}_{\text{KR}}}{\text{MSE}_{\text{CL}}}\right)$	$\log\left(\frac{\text{MSE}_{\text{GP}}}{\text{MSE}_{\text{CL}}}\right)$
				Scenario 1 (General internal points)				
5	200	-1.11(0.21)	0.23(0.09)	0.12(0.09)	0.33(0.11)	0.56(0.11)	0.57(0.11)	0.24(0.18)
	500	-2.13(0.18)	0.55(0.13)	0.37(0.11)	0.45(0.13)	0.91(0.17)	0.94(0.17)	0.76(0.18)
	1000	-2.04(0.18)	0.53(0.13)	0.42(0.13)	0.62(0.12)	1.18(0.19)	1.22(0.19)	0.41(0.20)
	2000	-2.21(0.20)	0.48(0.14)	0.38(0.14)	0.59(0.16)	1.06(0.22)	1.08(0.21)	0.81(0.17)
10	200	-0.03(0.16)	0.28(0.09)	0.13(0.07)	0.14(0.08)	0.10(0.07)	0.12(0.07)	-0.08(0.14)
	500	0.01(0.21)	0.43(0.13)	0.31(0.10)	0.29(0.11)	0.47(0.12)	0.47(0.12)	-0.01(0.17)
	1000	-0.50(0.22)	0.37(0.14)	0.30(0.12)	0.43(0.10)	0.54(0.12)	0.53(0.12)	-0.09(0.21)
	2000	-0.67(0.20)	0.42(0.13)	0.33(0.12)	0.51(0.11)	0.59(0.16)	0.60(0.16)	0.10(0.14)
20	200	1.46(0.14)	0.14(0.08)	-0.02(0.06)	-0.01(0.06)	-0.02(0.03)	-0.04(0.06)	0.17(0.15)
	500	1.09(0.15)	0.25(0.10)	0.11(0.07)	-0.01(0.07)	-0.07(0.06)	-0.03(0.06)	-0.18(0.16)
	1000	0.92(0.18)	0.48(0.11)	0.36(0.10)	0.00(0.11)	-0.10(0.08)	-0.02(0.08)	0.22(0.18)
	2000	0.73(0.22)	0.24(0.15)	0.24(0.12)	0.06(0.11)	0.18(0.11)	0.14(0.11)	0.15(0.19)
50	500	2.47(0.14)	0.08(0.09)	-0.02(0.07)	0.02(0.05)	-0.01(0.03)	-0.08(0.11)	0.06(0.19)
	1000	2.32(0.17)	0.08(0.12)	-0.02(0.10)	0.04(0.06)	-0.03(0.03)	-0.13(0.12)	-0.22(0.18)
	2000	2.12(0.17)	0.17(0.13)	0.18(0.10)	-0.01(0.06)	0.02(0.04)	0.00(0.11)	-0.08(0.19)
				Scenario 2 (Jump points of the feature data density)				
5	200	-0.72(0.17)	0.34(0.05)	0.33(0.04)	0.51(0.06)	0.60(0.07)	0.70(0.07)	0.32(0.10)
	500	-1.46(0.15)	0.42(0.05)	0.31(0.05)	0.44(0.06)	0.92(0.09)	1.03(0.09)	0.59(0.11)
	1000	-1.94(0.13)	0.48(0.06)	0.21(0.05)	0.33(0.07)	0.99(0.10)	1.11(0.10)	0.92(0.11)
	2000	-1.87(0.17)	0.46(0.05)	0.26(0.05)	0.33(0.06)	1.43(0.11)	1.53(0.11)	1.10(0.11)
10	200	0.59(0.12)	0.08(0.05)	0.03(0.04)	0.17(0.04)	0.09(0.03)	0.13(0.03)	0.14(0.09)
	500	0.44(0.14)	0.18(0.04)	0.08(0.04)	0.05(0.04)	0.09(0.04)	0.15(0.04)	-0.07(0.08)
	1000	0.27(0.11)	0.18(0.05)	0.11(0.04)	0.18(0.04)	0.29(0.05)	0.38(0.05)	-0.11(0.07)
	2000	0.02(0.13)	0.23(0.04)	0.11(0.04)	0.17(0.04)	0.43(0.05)	0.49(0.05)	-0.12(0.07)
20	200	1.92(0.12)	0.08(0.04)	0.03(0.03)	0.02(0.02)	-0.01(0.01)	-0.04(0.03)	0.04(0.07)
	500	1.77(0.10)	0.14(0.05)	0.01(0.03)	-0.02(0.03)	-0.01(0.04)	-0.02(0.02)	-0.07(0.07)
	1000	1.68(0.13)	0.08(0.05)	0.02(0.03)	-0.05(0.03)	-0.04(0.02)	-0.03(0.03)	-0.09(0.06)
	2000	1.50(0.12)	0.11(0.05)	0.06(0.03)	0.08(0.03)	0.02(0.02)	0.09(0.03)	-0.11(0.07)
50	500	2.85(0.09)	0.16(0.06)	0.05(0.04)	-0.01(0.04)	0.09(0.03)	0.14(0.06)	-0.04(0.08)
	1000	2.90(0.09)	0.20(0.05)	0.08(0.04)	-0.03(0.02)	0.03(0.02)	0.19(0.06)	-0.10(0.07)
	2000	2.82(0.10)	0.15(0.04)	0.08(0.03)	-0.01(0.01)	-0.01(0.01)	0.10(0.04)	-0.12(0.07)

In both scenarios, we apply the crystallization learning and existing methods to estimate $\mu(\mathbf{z}) = \sum_{j=1}^{d} c_j g_j(z_j)$ at the target points $\mathbf{z}_1, \ldots, \mathbf{z}_{100}$. We implement the crystallization learning with $L = 3$ for $d = 5, 10$ and $L = 2$ for $d = 20, 50$, and obtain $\hat{\mu}(\mathbf{z}_1), \ldots, \hat{\mu}(\mathbf{z}_{100})$. We implement the k-NN regression with $k = 5, 10, k^*$, where k^* equals the size of $\mathbb{V}_{\mathbf{z},L}$, and the local linear regression and kernel regression with bandwidth $h = 1$.

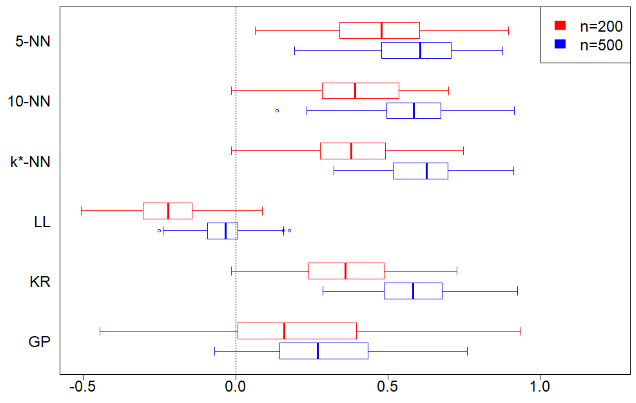
Table 2 presents the estimation results averaged over 100 simulations under two scenarios with different values of n and d. For each d, the estimation accuracy of the crystallization learning improves as the sample size increases, indicating its consistency in estimating $\mu(\cdot)$ in the convex hull $\mathcal{H}(\mathbb{X})$. For lower dimensional cases ($d = 5, 10$), the crystallization learning generally outperforms the existing methods, demonstrating that our approach is more efficient. For the higher dimensional cases ($d = 20, 50$), although our method cannot completely dominate the existing ones, the performances of different approaches are comparable. The results under Scenario 2 suggest the robustness of our method to the variation or sudden change in the feature data density. Overall, the crystallization learning performs well and is stable in estimating $\mu(\cdot)$ at general internal points of $\mathcal{H}(\mathbb{X})$ and jump points of the feature data density.

5.2. Real Data Application

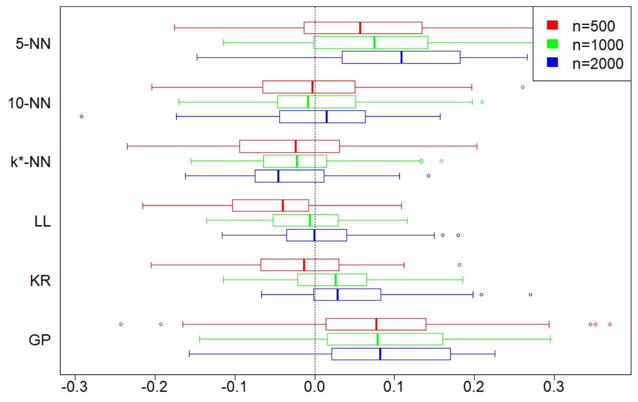
We apply the crystallization learning to several real datasets from the UCI repository. The critical assessment of pro-



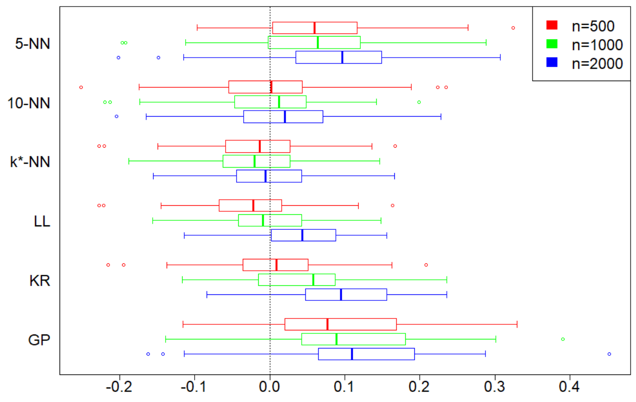
(a) CASP dataset



(b) Concrete dataset



(c) Parkinson's motor UPDRS



(d) Parkinson's total UPDRS

Figure 7. Boxplots of $\log(\text{MPSE}_{\mathcal{M}}/\text{MPSE}_{\text{CL}})$ corresponding to k-NN ($k = 5, 10, k^*$, where k^* equals the size of $\mathbb{V}_{\mathbf{z},L}$), local linear (LL) regression, kernel regression (KR) and Gaussian process (GP) in estimating $\mu(\cdot)$ under different datasets and sizes of the training set (n).

tein structure prediction (CASP) dataset[2] (Betancourt & Skolnick, 2001) contains experimental records on protein structure prediction. The CASP dataset includes 45730 records of 9 features, where the response is the root mean squared deviation (RMSD) of the residues. The Concrete dataset[3] (Yeh, 1998) consists of 1030 experimental records of concrete compressive strength measurement. We use the content of 7 concrete ingredients and the age of a concrete sample to predict its compressive strength. Parkinson's telemonitoring dataset[4] (Tsanas et al., 2010) is composed of 5875 voice recordings of 16 biomedical voice measures from 42 patients with early-stage Parkinson's disease in a six-month trial. We use these 16 biomedical voice measures to predict the motor and total UPDRS (unified Parkinson's disease rating scale) scores.

For each dataset, we take 100 bootstrap samples without replacement of size n (n = 200, 500, 1000 or 2000) for training and 100 bootstrap samples of size 100 for testing. To eliminate the impact of feature correlations and scales, we standardize the principal components of features in the training set and take them as the feature points $\mathbf{x}_1, \ldots, \mathbf{x}_n$. The same transformation is applied to the features in the testing set to obtain the target points $\mathbf{z}_1, \ldots, \mathbf{z}_{100}$. We take $L = 3$ for crystallization learning, and implement the k-NN regression with $k = 5, 10, k^*$, where k^* equals the size of $\mathbb{V}_{\mathbf{z},L}$, and the local linear regression and kernel regression with bandwidth $h = 1$. Based on the testing set, we quantify the performance of the method $\mathcal{M}$ by the mean predictive

[2]https://archive.ics.uci.edu/ml/datasets/Physicochemical+Properties+of+Protein+Tertiary+Structure

[3]https://archive.ics.uci.edu/ml/datasets/Concrete+Compressive+Strength

[4]https://archive.ics.uci.edu/ml/datasets/Parkinsons

squared error (MPSE),

$$\text{MPSE}_{\mathcal{M}} = \frac{1}{100}\sum_{k=1}^{100}\{\hat{\mu}_{\mathcal{M}}(\mathbf{z}_k) - y_k\}^2,$$

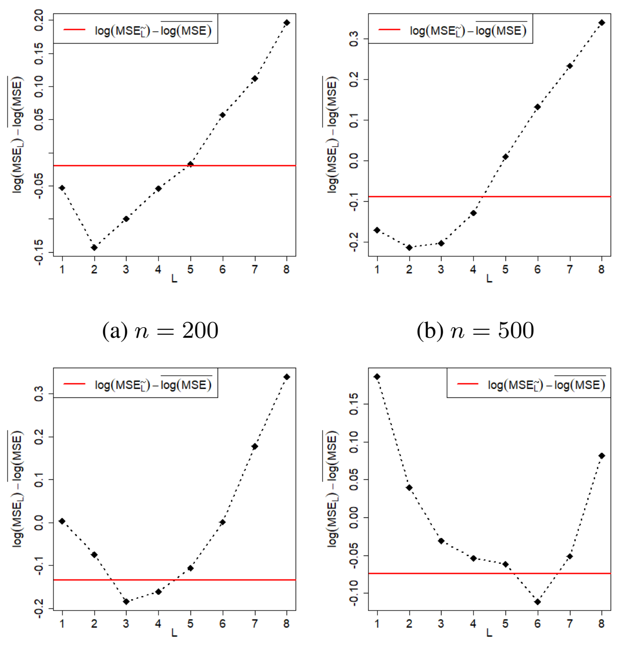
where y_k's are responses corresponding to $\mathbf{z}_k$'s.

Figure 7 shows the comparison results averaged over 100 bootstrap samples between our method and existing ones under different datasets and sizes of the training set (n). It is clear that as n increases, the advantage of the crystallization learning over the existing methods amplifies. Overall, the crystallization learning dominates all the existing methods in most of the cases.

5.3. Selection of L

To examine the data-driven selection procedure for L proposed in Section 2.4, we conduct experiments under Scenario 1 of Section 5.1. With candidate values $L = 1, \ldots, 8$ and $d = 5$, we simulate 100 training datasets with sample sizes $n = 200, 500, 1000, 2000$ respectively and generate the corresponding sets of target points.

Figure 8 shows the estimation results of our method averaged over 100 simulations under different sample sizes, when using different candidate values of L and the selected $\tilde{L}$. It is clear that as the sample size n increases, the optimal value of L, which results in the smallest averaged value of $\log(\text{MSE}_L)$, increases. This is reasonable because a larger n would lead to smaller volumes of simplices in $\mathcal{N}_L(\mathbf{z})$ and thus a larger L is needed for more accurate estimation. In addition, the averaged value of $\log(\text{MSE}_{\tilde{L}})$ is closer to the smallest averaged value of $\log(\text{MSE}_L)$ when n is larger, suggesting the effectiveness of our LOO-CV procedure in improving the estimation accuracy.



(a) $n = 200$ (b) $n = 500$

(c) $n = 1000$ (d) $n = 2000$

Figure 8. Averaged values of $\overline{\log(\text{MSE}_L)} - \overline{\log(\text{MSE})}$ ($L = 1, \ldots, 8$) and $\log(\text{MSE}_{\tilde{L}}) - \overline{\log(\text{MSE})}$ under different sample sizes (n), where MSE_L is the MSE using the hyperparameter L and $\overline{\log(\text{MSE})} = \sum_{L=1}^{8}\log(\text{MSE}_L)/8$.

6. Conclusions

The Delaunay triangulation is a powerful tool to partition the feature space in a data-driven way, which has the least roughness for smooth surface reconstruction. We incorporate the Delaunay triangulation into the framework of nonparametric regression and develop the crystallization learning procedure. Without the need to triangulate the entire feature space which becomes infeasible for high-dimensional cases, our method conducts the Delaunay triangulation locally at each specific target point like crystal growth. The conditional expectation $\mu(\mathbf{z})$ at the target point $\mathbf{z} \in \mathcal{H}(\mathbb{X})$ is estimated by fitting a local linear model to the data points of the Delaunay simplices identified by the crystallization search. Compared with existing nonparametric regression methods, our method is more adaptive to the local geometric structure of the data, which selects the neighbor data points uniformly in all directions and their weighted mean is closer to the target point $\mathbf{z}$. Both theoretical studies and numerical experiments show that the crystallization learning is consistent in estimating $\mu(\cdot)$ and it generally outperforms the existing methods.

Given that the crystallization learning is a local approach, it is possible to combine it with the uniform design and develop a global version of crystallization learning in a hierarchical way. As our method searches $\mathcal{N}_{\mathbf{z},L}$ in a deterministic way, we can also develop the stochastic crystallization search to reduce the boundary effect on the estimated conditional expectation function $\hat{\mu}(\cdot)$. Other possible extensions include extrapolation of $\mu(\mathbf{z})$ at $\mathbf{z} \notin \mathcal{H}(\mathbb{X})$ with the Möbius transformation (Zhou et al., 2019), regression problems in other metric spaces (e.g., manifold regression and structured output) as discussed in Hein (2009) and online regression problems (Kuzborskij & Cesa-Bianchi, 2017).

Acknowledgement

We thank the four anonymous reviewers for insightful suggestions that have significantly improved the paper. This research was supported by funding from the Research Grants Council of Hong Kong (17308420) and TCL Corporate Research (Hong Kong).

References

Altman, N. S. An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185, 1992.

Benedetti, J. K. On the nonparametric estimation of regression functions. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(2):248–253, 1977.

Betancourt, M. R. and Skolnick, J. Universal similarity measure for comparing protein structures. *Biopolymers*, 59(5):305–309, 2001.

Chang, T. H., Watson, L. T., Lux, T. C. H., Bernard, J., Li, B., Xu, L., Back, G., Butt, A. R., Cameron, K. W., and Hong, Y. Predicting system performance by interpolation using a high-dimensional Delaunay triangulation. In *Proceedings of the High Performance Computing Symposium*, HPC '18, San Diego, CA, USA, 2018a. Society for Computer Simulation International.

Chang, T. H., Watson, L. T., Lux, T. C. H., Li, B., Xu, L., Butt, A. R., Cameron, K. W., and Hong, Y. A polynomial time algorithm for multivariate interpolation in arbitrary dimension via the Delaunay triangulation. In *Proceedings of the ACMSE 2018 Conference on - ACMSE 18*. ACM Press, 2018b.

Chang, T. H., Watson, L. T., Lux, T. C. H., Butt, A. R., Cameron, K. W., and Hong, Y. Algorithm 1012: DELAUNAYSPARSE: Interpolation via a sparse subset of the Delaunay triangulation in medium to high dimensions. *ACM Transactions on Mathematical Software*, 46 (4):1–20, 2020.

Cleveland, W. S. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368):829–836, 1979.

Cleveland, W. S. and Devlin, S. J. Locally weighted regression: an approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83(403): 596–610, 1988.

Cover, T. and Hart, P. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

de Berg, M., Cheong, O., van Kreveld, M., and Overmars, M. Delaunay triangulations. In *Computational Geometry*, pp. 191–218. Springer Berlin Heidelberg, 2008.

Delaunay, B. Sur la sphère vide. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na*, 6:793–800, 1934.

Fan, J. and Gijbels, I. *Local Polynomial Modelling and Its Applications*. Routledge, 2018.

Hardle, W. and Gasser, T. Robust non-parametric function fitting. *Journal of the Royal Statistical Society. Series B (Methodological)*, 46(1):42–51, 1984.

Hein, M. Robust nonparametric regression with metric-space valued output. In *Advances in Neural Information Processing Systems*, volume 22. Curran Associates, Inc., 2009.

Kuzborskij, I. and Cesa-Bianchi, N. Nonparametric online regression while learning the metric. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Liu, Y. and Yin, G. The Delaunay triangulation learner and its ensembles. *Computational Statistics & Data Analysis*, 152:107030, 2020.

Nadaraya, E. A. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964.

Priestley, M. B. and Chao, M. T. Non-parametric function fitting. *Journal of the Royal Statistical Society. Series B (Methodological)*, 34(3):385–392, 1972.

Sibson, R. Locally equiangular triangulations. *The Computer Journal*, 21(3):243–245, 1978.

Stone, C. J. Consistent nonparametric regression. *The Annals of Statistics*, 5(4):595–620, 1977.

Tsanas, A., Little, M., McSharry, P., and Ramig, L. Accurate telemonitoring of Parkinson's disease progression by non-invasive speech tests. *IEEE Transactions on Biomedical Engineering*, 57(4):884–893, 2010.

Watson, G. S. Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A (1961-2002)*, 26(4): 359–372, 1964.

Yeh, I.-C. Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete Research*, 28(12):1797–1808, 1998.

Zhou, Z., Tan, S., Xu, Z., and Li, P. Möbius transformation for fast inner product search on graph. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.