Article

Clustering and Curve Fitting by Line Segments

Hrishikesh D. Vinod [1,†,‡] and Fred Viole [1,‡]

[1] Fordham University; vinod@fordham.edu; fviole@fordham.edu
* Correspondence: vinod@fordham.edu; Tel.: +1-718-817-4065
† Current address: 441 E Fordham Rd, Bronx, NY 10458

Abstract: Nonlinear nonparametric statistics (NNS) algorithm offers new tools for curve fitting. A relationship between k-means clustering and NNS regression points is explored with graphics showing a perfect fit in the limit. The goal of this paper is to demonstrate NNS as a form of unsupervised learning, and supply a proof of its limit condition. The procedural similarity NNS shares with vector quantization is also documented, along with identical outputs for NNS and a k nearest neighbours classification algorithm under a specific NNS setting. Fisher's iris data and artificial data are used. Even though a perfect fit should obviously be reserved for instances of high signal to noise ratios, NNS permits greater flexibility by offering a large spectrum of possible fits from linear to perfect.

Keywords: clustering; curve fitting; nonparametric regression; smoothing data; polynomial approximation

1. Introduction

In a recent paper [1] demonstrate a nonlinear regression (NNS) algorithm comprised of partial moment quadrant means originally presented in [2]. NNS quadrant means are generated according to an order parameter O, such that $4^{(O-1)}$ quadrant means are calculated from a hierarchical partitioning internally used with partial moment statistics. Central to this technique is the claim and proof that linearly connecting these quadrant means will perfectly fit any underlying $f(x)$ in the limit at some finite O.

Our motivation for writing this paper is as follows. Polynomial type curve fitting methods claim only approximate results. Any bandwidth-based nonparametric regression, illustrated by the popular R package (np), [3] does not enjoy perfection because bandwidths apply to variable x for all values of x–not tailor-made for each observation. Nadaraya-Watson type kernel regression users must choose both a kernel function (e.g. Gaussian) and a bandwidth parameter h. These choices are subject to a well-known trade-off. As h tends to zero, the bias diminishes but variance increases. NNS algorithm, also available as an R package is subject to a similar trade-off, but contains features mitigating both concerns while retaining the perfect fit ability. The variance in NNS is reduced due to the use of linear segments between regression points. The bias can be reduced by increasing the number of regression points (and by extension linear segments) used in the estimate. Of course this bias reduction comes at the expense of increasing variance, but the linear segments do not permit much variability of estimates.

The claim that kernel methods are intrinsically approximate can be verified as follows. Assume we want to approximate a density $\phi(x)$ defined as the limit of the central difference among (cumulative) distribution functions Φ:

$$\phi(x) = \lim_{h\to 0}\left(\frac{1}{h}\right)\left[\Phi\left(x+\frac{h}{2}\right) - \Phi\left(x-\frac{h}{2}\right)\right]. \tag{1}$$

The histogram method of density estimation counts the number of data points of the underlying random variable X lying in the neighbourhood of a specific value x and divides by the bandwidth h. Rosenblatt suggested replacing the central difference with a kernel weight function, which is symmetric with positive variance and integrates to unity. A normal kernel defined by weights w_t obtained from the standard normal density, $K(w_t) \sim N(0, \sigma^2)$, also has positive variance and integrates to unity, besides being so simple. Hence, a popular method in kernel density estimation is to substitute $w_t = \frac{(x_t - x)}{h}$, where x is the point at which the function is evaluated and x_t are nearby observed data points, into the familiar normal density formula.

Since all kernel functions are continuous probability distributions, the probability that a point exactly equals a particular value (limiting data value) x_t is always zero. Even if the kernel based regression asymptotically approaches the correct value for some x, it can never achieve the exact x for all observations at the same time. Furthermore, if one employs the popular leave-one-out-cross validation, x_t is removed from the analysis, thus guaranteeing the estimate will not exactly equal x_t since, again, the probability of a specific point in a continuous distribution of surrounding points is equal to zero. This manuscript explains how (i) NNS converges to the exact fit for all observations simultaneously in both uni- and multivariate cases, and (ii) NNS limit is achieved in finite steps, not asymptotically.

We admit that a perfect fit versus an exceptionally good fit may seem like splitting hairs. However, the former affords greater flexibility by offering a large spectrum of possible fits from linear to perfect. Such flexibility in turn allows NNS to use an internal dependence measure to ascertain the signal to noise ratio (SNR) and restrict the order accordingly. Spline interpolation shares this approach, albeit with different techniques. Splines can fit any $f(x)$ in a similar limit condition whereby the number of knots (analogous to NNS quadrant means) will equal the number of observations. One popular method to avoid overfitting with splines is to impose a penalization upon the piecewise polynomial components to optimize the fit. A linear spline is defined as

$$f(x) = \beta_0 + \beta_1 x + \sum_{i=1}^{K} b_i (x - k_i)_+,$$

where b_i refers to the weight of each linear function and $(x - k_i)_+$ refers to the ith linear function with a knot at k_i. The weights are then chosen by satisfying

$$\sum_{i=1}^{K} b_i^2 < C,$$

where C is a penalization criteria. But again, there is no agreed upon best method to determine an optimal penalization criterion. NNS's proposed use of dependence for this task has the benefit of being objective and admitting replicable results.

Our free parameter SNR also permits future independent research to be seamlessly incorporated into the NNS algorithm, should further advances present themselves. Bandwidth selection and kernel functions are comparatively mature in their development cycles without much room for theoretical advancement. When SNR is large (small) the NNS algorithm will use a larger (smaller) parameter O implying shorter (longer) linear segments. Low SNR requires a greater need to avoid overfitting.

[1] offer a complete analysis of goodness of fits and partial derivatives against varying degrees of noise, while noting the NNS dependence with each experiment. They find NNS R^2 results are not uniformly equal to 1, even though they have the capability to be, and note how partial derivative estimation is better served with lower orders in the presence of noise, that NNS dependence compensates for. Our section 3 illustrates practical advantages of NNS over the myriad of competitors in regression problems, emanating from its perfect fit capability achieved relatively fast and rather simply. This section also notes the similarities NNS shares with vector quantization per [4] and [5].

One illustration uses a progression of partial moment quadrants alongside their linear segments to highlight the piece-wise linearity of the NNS fit which enables interpolation and extrapolation along the fitted lines. Thus, the NNS linearity fills a long-standing gap in the literature since inter-extra-polations of kernel type nonparametric regressions are quite unstable and highly sensitive to noise structure.

We begin by examining the strong similarity between NNS and k-means clustering objective functions despite subtle difference in initial parameter specification.

1.1. *k-means Objective*

The k-means objective is to identify k sets S_i of clusters and points x belonging to each cluster which minimizes the within-cluster sum of squares:

$$\arg\min_{S} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2. \tag{2}$$

1.2. *NNS Objective*

The NNS partition dual objective is to identify set S_i where each cluster results from a partial moment quadrant and to identify point z that minimizes the within quadrant sum of squares.

$$\arg\min_{z} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - z_i||^2. \tag{3}$$

Since the arithmetic mean (μ) is a least-squares estimator, this satisfies the minimization of the within-quadrant sum of squares objective, thus $z_i = \mu_i$ for any given quadrant.

1.3. *Weirstrass Approximation Theorem*

Weierstrass' (1885) famous *Approximation Theorem* states that any continuous function ($f(x)$) can be approximated arbitrarily closely by a polynomial ($p_n(x)$) of a sufficiently high degree (n). That is, given a compact set K so that $x \in K$, and $\epsilon > 0$, there exists an n offering a "close" approximation defined by:

$$d(f, p_n) = \sup_{x \in K} |f(x) - p_n(x)| < \epsilon. \tag{4}$$

1.4. *Implications*

Unlike polynomial approximations, NNS offers a perfect, not approximate, limiting fit to $f(x)$ denoted by $f_O(x)$, for a finitely large order parameter O. That is, given a compact set K so that $x \in K$, there exists a finitely large O satisfying:

$$d(f, f_O) = \sup_{x \in K} |f(x) - f_O(x)| \to 0. \tag{5}$$

The proof is straightforward. We start with observed values of x and $f(x)$. As O increases, the number of sets and line segments joining set means increase exponentially according to $4^{(O-1)}$, until each observation becomes its own set mean in the limit.

Because NNS increases exponentially from a base of 4, this finite limit condition O will occur much more quickly than a corresponding polynomial degree. Then, $f(x) = f_O(x)$ must hold, implying a perfect fit in the limit. Similarly in a limit condition, k-means will have every observation occupy its own cluster, demonstrating an equivalence to NNS quadrants. Uni- and multivariate examples demonstrating NNS fitting and quadrants (clusters) in this limit condition follow from [6].

The plan of the remaining paper is as follows. Section 2 considers a somewhat hard nonlinear univariate curve-fitting problem for a distinctly clustered dataset. It conveys through a series of images, the relationship between k-means and NNS clusters. Section 3 explores the relation between clusters and curve fitting via linear segments. We also note the procedural similarities NNS shares with vector quantization after the partitioning of the data. Section 4 considers the multivariate case.

doi:10.20944/preprints201801.0090.v1

2. NNS and *k*-means Clusters Visualization

We created a distinctly clustered dataset to present a progression of clusters in Figures 1:3 using the same *k* for both *k*-means and NNS.

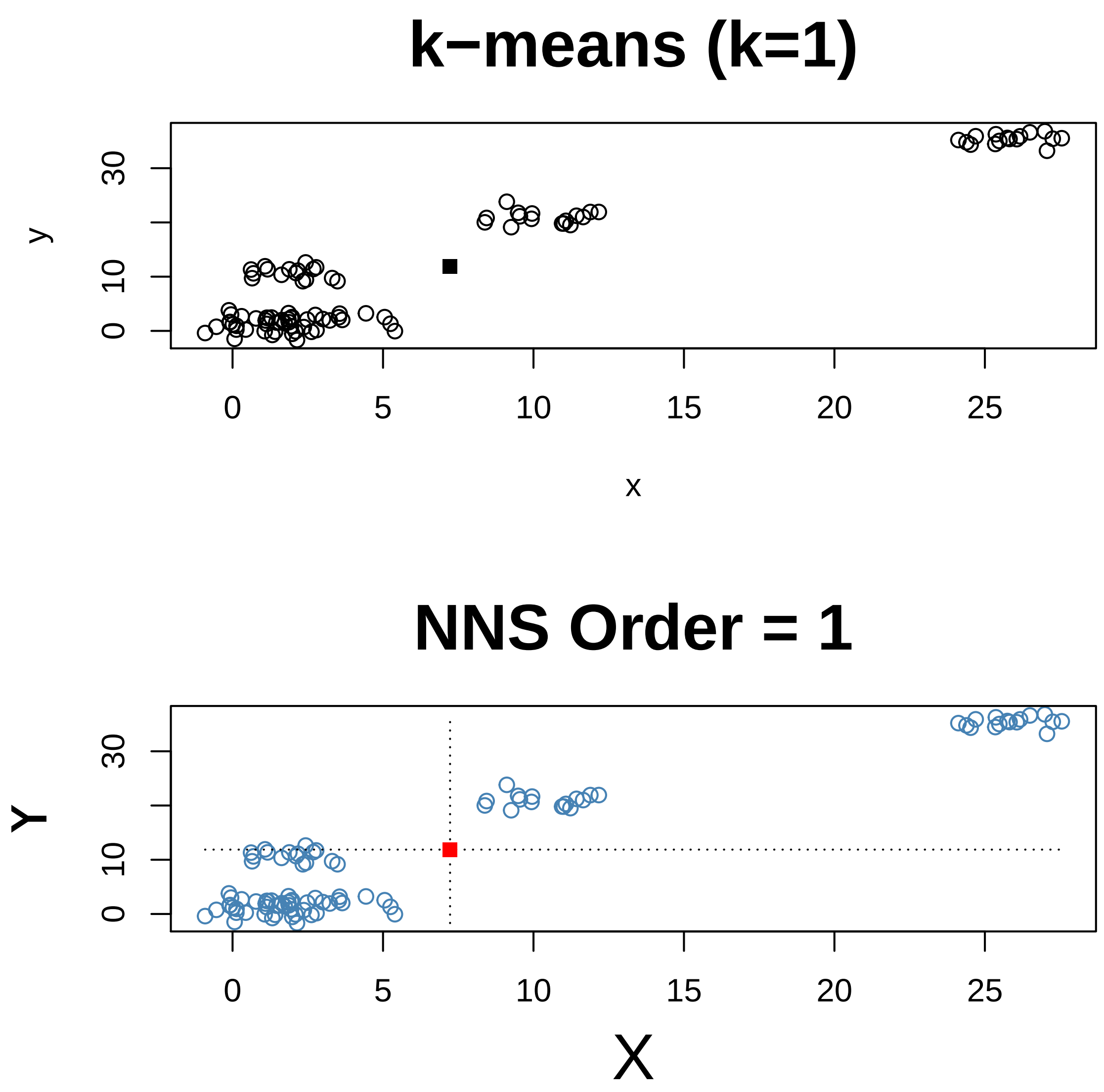

Figure 1. *k*=1 for *k*-means and NNS partitioning. The single point shown as a square box is identical for both methods.

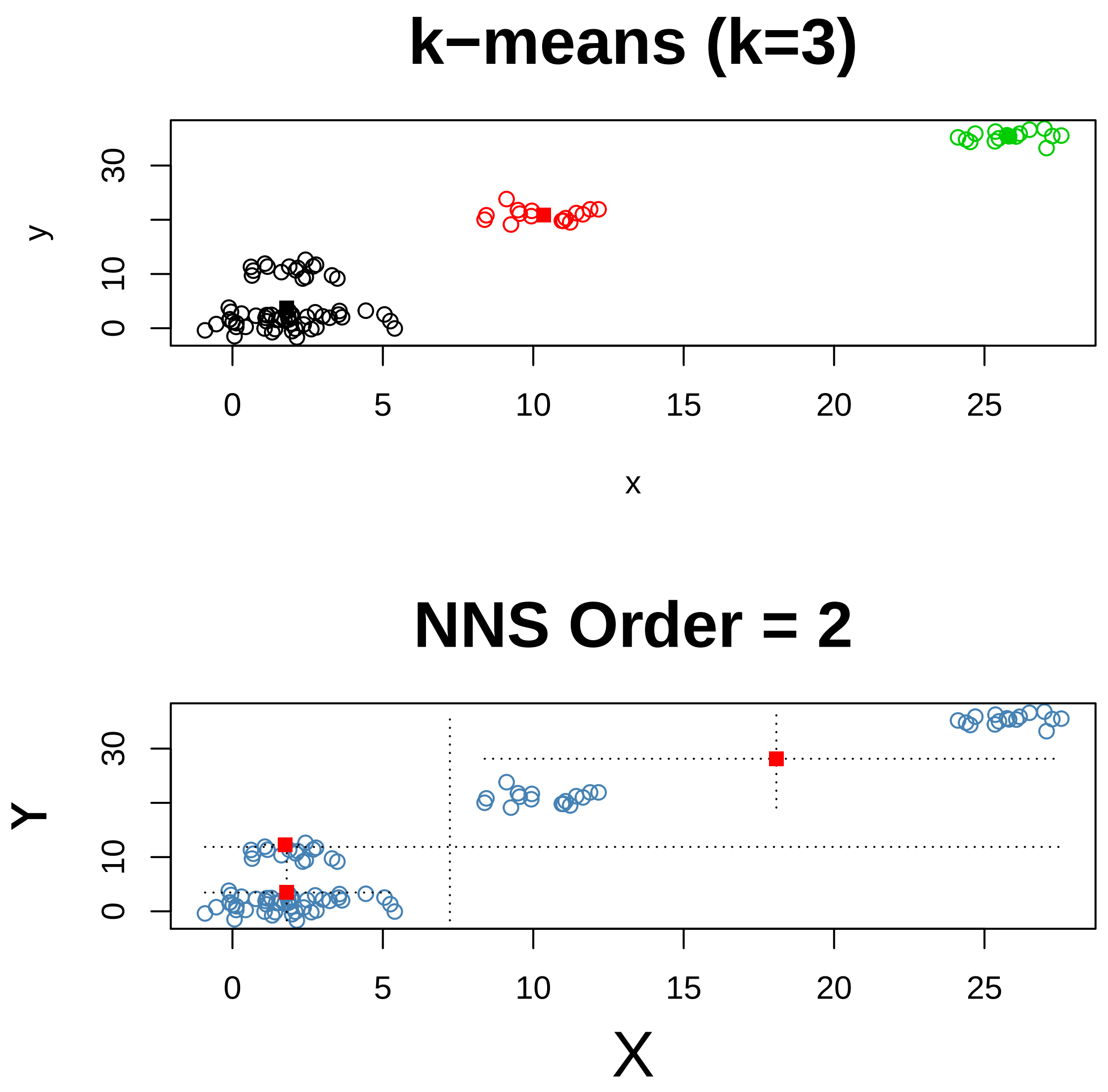

Figure 2. *k*=3 for *k*-means and NNS partitioning. NNS further partitions from the 1st point and shares no points in common with *k*-means.

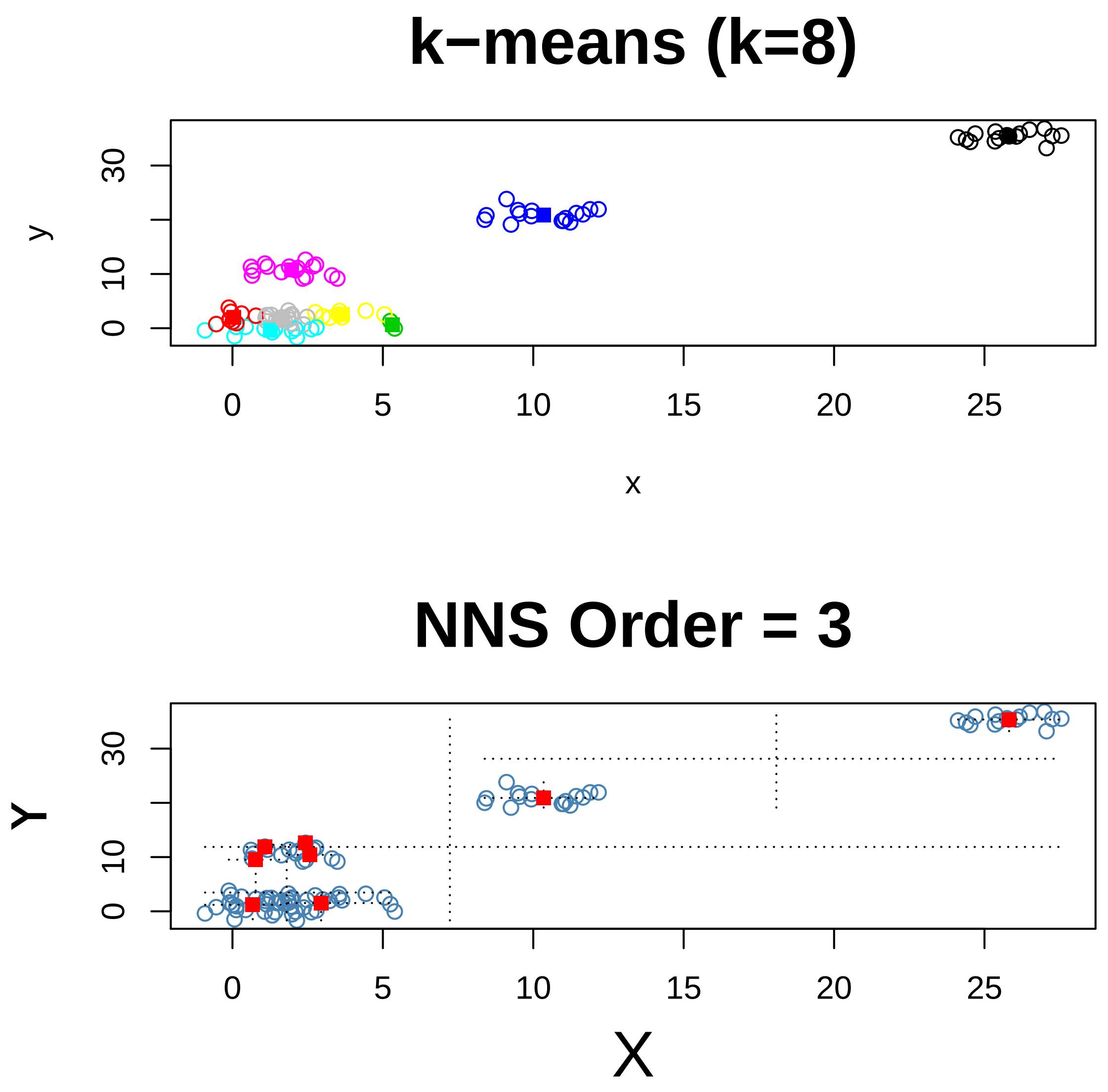

Figure 3. k=8 for k-means and NNS partitioning. As the number of k increases, NNS and k-means will generate more identical points.

Figure 1 reveals that the k-means cluster and NNS clusters are identical for the $k = 1$ case. Figures 2 and 3 depict the $k = 3, 8$ cases. Several shared points in the upper right section of Figures 1 and 3 are visible.

3. NNS Clusters and NNS Regression

This section explores the relation between NNS clusters and NNS line segments used for curve fitting. Figure 4 displays two columns having three figures each. The original univariate data (provided in the Supplementary Material A) are displayed in all six figures, showing a large dip after a peak at $x = 25$. The filled-in squares in all figures represent the cluster means. The right hand figures show regression fit as straight lines joining NNS quadrant (cluster) means equal to $\bar{X}, \bar{Y}$ and end points which are determined from another internal NNS algorithm described in [1].

Figure 5 extends the right hand panel of Figure 4 for orders $O = 1, \ldots, 5$, so that it ultimately reaches perfect fit.

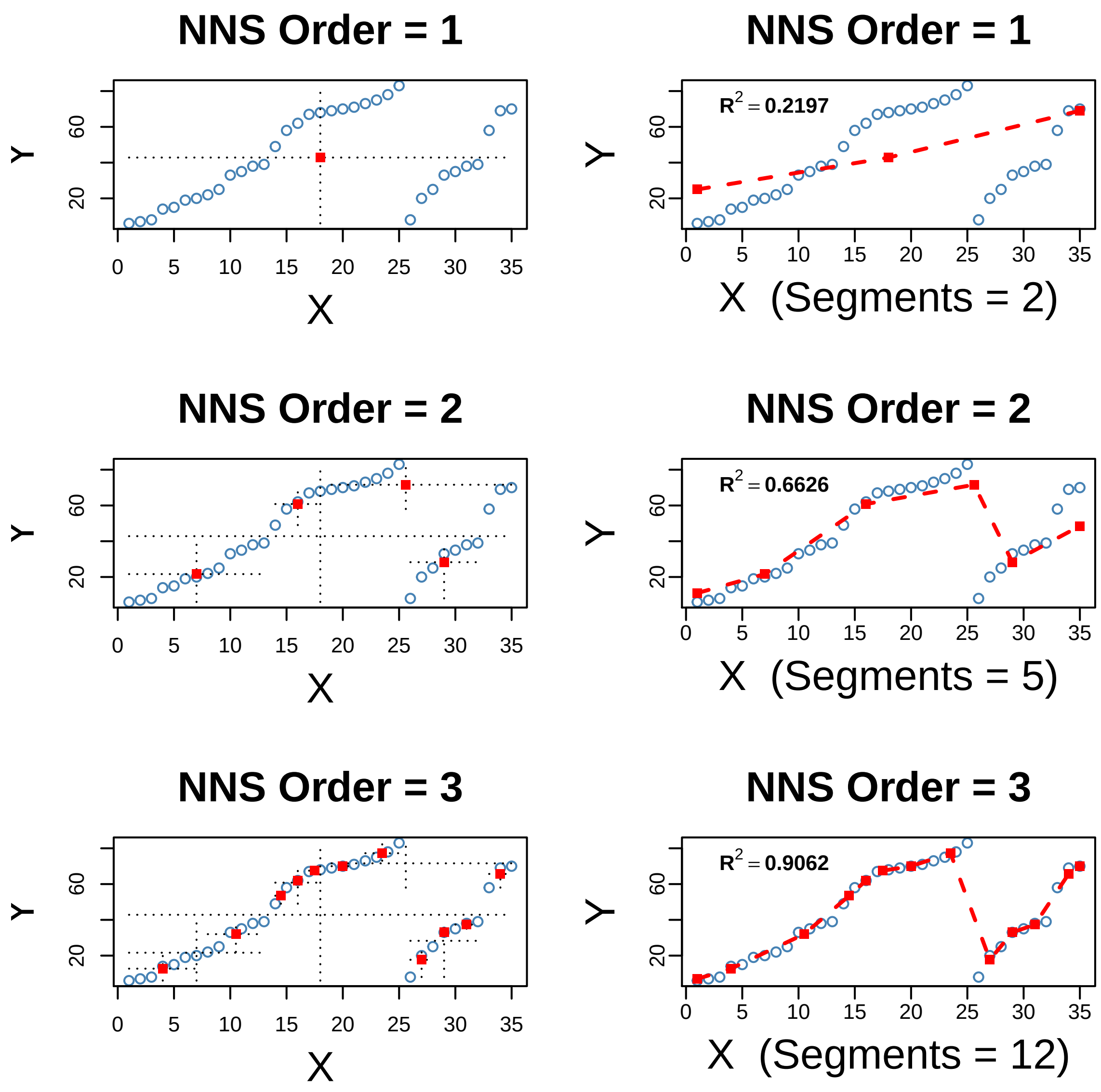

Figure 4. Nonlinear data, NNS Clusters and fitted regression lines along the right hand column for *O*=1:3.

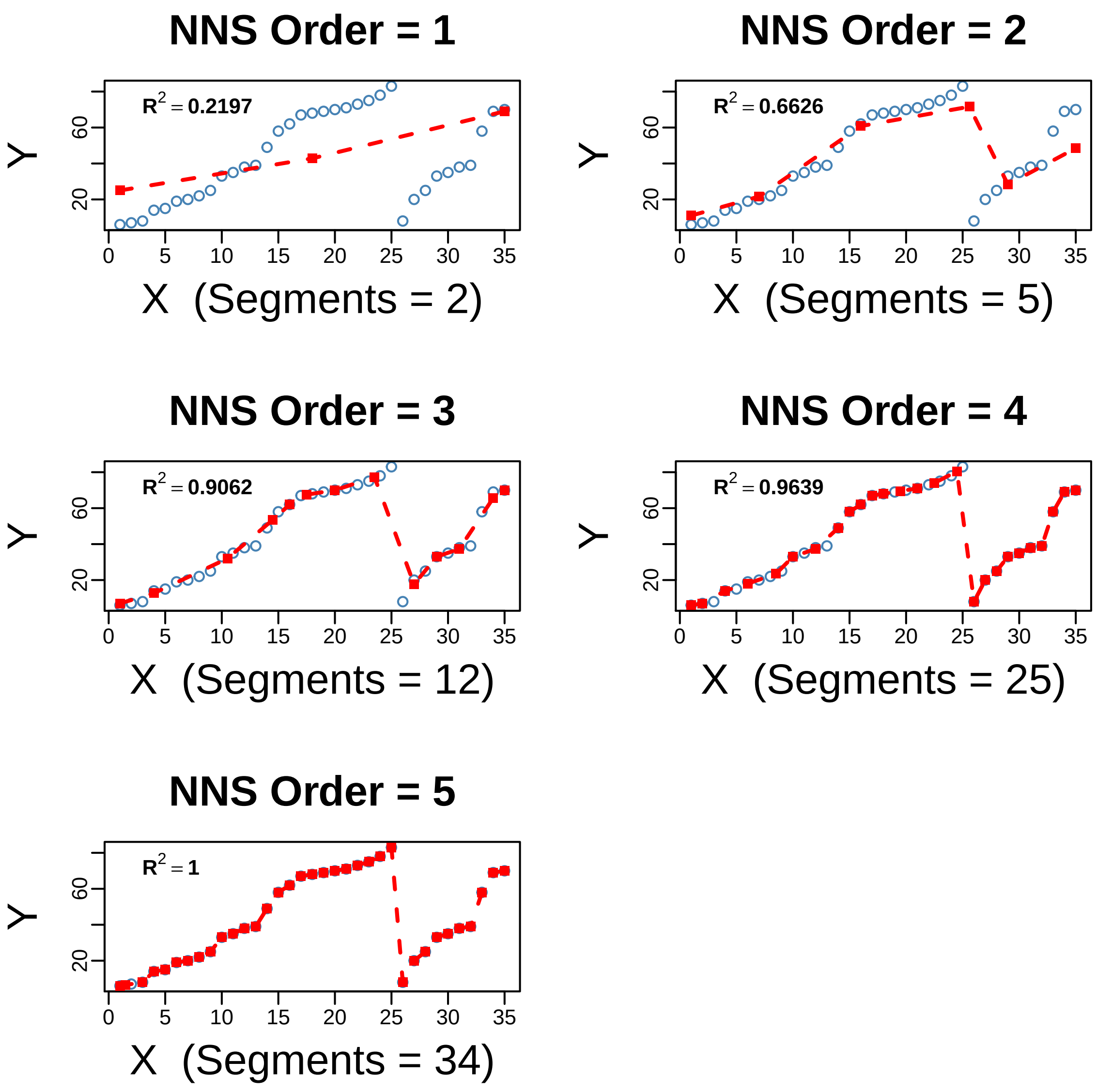

Figure 5. Progression of O in univariate NNS fitting. $f(x)$ in blue, NNS fit in red. The k quadrant segment means are presented as red squares with red dashed lines for connecting segments.

3.1. Quadrant (Cluster) Identifications

The NNS algorithm assigns each observation a special identification number reflecting the quadrant where it ultimately falls. In vector quantization, this step is analogous to the “codebook" which represents the “codevectors". The codevectors are the segment means from a vector quantization partitioning, and are very similar to NNS cluster points as defined above. Again, the primary difference is the lack of an initial specification of a number of desired points for NNS. The sequence of partitioning and assignment of quadrant identifications for each observation is included here for completeness.

Table 1 presents the special identification numbers for our univariate example in the limit condition, that is for $O = 5$. Column 1 is the observation number, column entitled “y” has values of the dependent variable, “quadrant” is the sequence of clusters assigned to each observation and the “fitted value” column contains the NNS predictions $\hat{y}$. Columns 5:9 are entitled “order.O” for the relevant order and lists the special identification numbers mentioned above.

In the third column entitled “quadrant”, the format “q...” describes the sequential partitions. For the first observation, “q44444” was first assigned a partition “4”, as noted in column “order.1”. The final digit 4 is the latest quadrant assignment for the last “order”, which is $O = 5$ here. The bottom row of Table 1 reports $k \leq 4^{O-1}$, the number of nonempty quadrants.

The internal NNS numbering of quadrants is not of theoretical significance. NNS uses 1=CUPM (North East quadrant); 2=DUPM (North West quadrant); 3=DLPM (South East quadrant); 4=CLPM (South West quadrant). The rationale behind bookending the CUPM and CLPM with the lowest and highest assignments respectively is computational ease via “min” and “max” commands if those quadrants should be called. For example, our first observation is consistently in the lower left CLPM or fourth quadrant after 5 iterative partitions. These are also used in multivariate dependence defined in [7].

Table 1. Quadrant (cluster) identification for univariate example where $O = 5$. Sequence of quadrant identifications as O increases and k as the number of non-empty quadrants including endpoints for each O.

i	y_i	NNS.ID	fitted value	order.1	order.2	order.3	order.4	order.5
1:	6	q44444	6	q4	q44	q444	q4444	q44444
2:	7	q44441	7	q4	q44	q444	q4444	q44441
3:	8	q44414	8	q4	q44	q444	q4441	q44414
4:	14	q44244	14	q4	q44	q442	q4424	q44244
5:	15	q44144	15	q4	q44	q441	q4414	q44144
6:	19	q44124	19	q4	q44	q441	q4412	q44124
7:	20	q44114	20	q4	q44	q441	q4411	q44114
8:	22	q41444	22	q4	q41	q414	q4144	q41444
9:	25	q41414	25	q4	q41	q414	q4141	q41414
10:	33	q41244	33	q4	q41	q412	q4124	q41244
11:	35	q41144	35	q4	q41	q411	q4114	q41144
12:	38	q41124	38	q4	q41	q411	q4112	q41124
13:	39	q41114	39	q4	q41	q411	q4111	q41114
14:	49	q24444	49	q2	q24	q244	q2444	q24444
15:	58	q24144	58	q2	q24	q241	q2414	q24144
16:	62	q22444	62	q2	q22	q224	q2244	q22444
17:	67	q21444	67	q2	q21	q214	q2144	q21444
18:	68	q21144	68	q2	q21	q211	q2114	q21144
19:	69	q14444	69	q1	q14	q144	q1444	q14444
20:	70	q14414	70	q1	q14	q144	q1441	q14414
21:	71	q14144	71	q1	q14	q141	q1414	q14144
22:	73	q12444	73	q1	q12	q124	q1244	q12444
23:	75	q12414	75	q1	q12	q124	q1241	q12414
24:	78	q12144	78	q1	q12	q121	q1214	q12144
25:	83	q12114	83	q1	q12	q121	q1211	q12114
26:	8	q34444	8	q3	q34	q344	q3444	q34444
27:	20	q34244	20	q3	q34	q342	q3424	q34244
28:	25	q34144	25	q3	q34	q341	q3414	q34144
29:	33	q32444	33	q3	q32	q324	q3244	q32444
30:	35	q31444	35	q3	q31	q314	q3144	q31444
31:	38	q31244	38	q3	q31	q312	q3124	q31244
32:	39	q31144	39	q3	q31	q311	q3114	q31144
33:	58	q13444	58	q1	q13	q134	q1344	q13444
34:	69	q13244	69	q1	q13	q132	q1324	q13244
35:	70	q13144	70	q1	q13	q131	q1314	q13144
				$k = 1$	$k = 4$	$k = 11$	$k = 24$	$k = 35$

4. Multivariate Iris Quadrants (Clusters)

We use Fisher's celebrated 'iris' dataset using the categorical variable "Species" as the dependent variable to illustrate the multivariate case.

Figure 6 demonstrates the perfect fit and classification achieved by NNS along the bottom right. Since multivariate clusters for four or more regressors cannot be visualized, we present Y and $\hat{Y}$ for a given NNS order. Since 'iris' has a categorical dependent variable, the plots have a staircase appearance. When calculating goodness of fit and prediction accuracy, NNS rounds its values for categorical predictions but plots actual values.

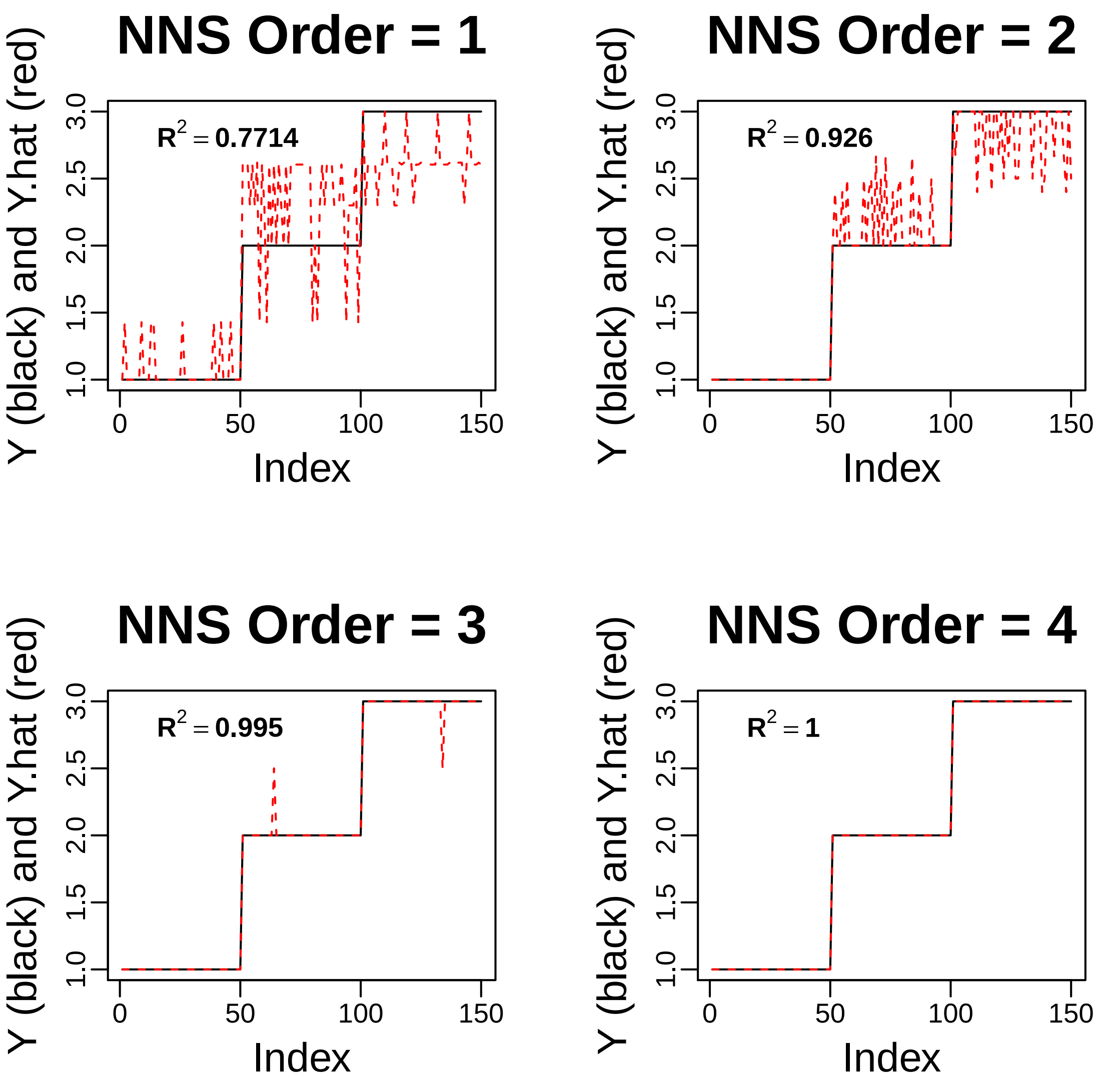

Figure 6. Progression of O in multivariate NNS fitting. Y and $\hat{Y}$ are displayed (in black and red respectively) for 'iris' dataset containing 4 regressors.

The quadrant identifications assigned by NNS for the corresponding regressor for the iris data are tabulated next. Table 2 shows the truncated output of the multivariate NNS classification using $O = 4$. Column 1 is the observation number, "y" is the dependent variable representing the 3 species types of iris, "NNS identifier" is the sequence of identifications assigned to each observation and "fitted value" is the NNS prediction of the 'iris' dependent variable.

The multivariate identifications are not simply quadrant identifications as in the univariate case. These identifications are derived from the "regression point matrix" (RPM) internal to NNS, for ease of reference as suggested by [8]. "Regression points" are the quadrant mean values interpreted as $\hat{X}$ values derived by NNS for a given O for all regressors in the X matrix. The reader may benefit from the vector quantization codebook analogy, given the RPM and codebooks serve identical purposes.

Since 'iris' has 4 regressors and 150 observations, the maximum dimensions (in the limit condition whereby every observation is a quadrant mean) of the RPM will be a 150×4 (row-column) matrix. The reason for these alternative identifications is straightforward, efficiency. For 4 regressors, a quadrant identification of $O = 4$ would require 16 entries of a number 1:4 per the univariate quadrant identification example immediately preceding. Whereby using the RPM, we can assign the unique X_i identification with much fewer entries even when some or all regressors have double digit entries. If each regressor is aligned with its first regression point *not partial moment quadrant*, each regressor will be assigned a "1". Thus, we have only 4, not 16 digits in our NNS identifier for multiple regressors.

The phrase "unique observation set" refers to the dependent variable y and a matrix of regressors X for the i-th observation, $(y, X)_i$ and the identification simply assigns the interval number $(1, 2, 3, \ldots)$ along each regressor column in the RPM that corresponding X_i is found. X_1 and the first few rows of RPM are given by:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
X_1	5.1	3.5	1.4	0.2

RPM:

	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
1:	4.300000	2.000000	1.0000	0.1000000
2:	4.612500	2.272727	1.3250	0.1852941
3:	5.012000	2.604545	1.5625	0.3000000
4:	5.388889	2.841667	1.9000	0.4000000
5:	5.704762	3.000000	3.2000	0.5500000
6:	6.057895	3.154167	3.5750	1.0000000

Now focus on the first line of Table 2 having the entry: "3.7.2.2" in the "NNS identifier" column. It means the first 'iris' regressor "Sepal.length" was assigned a "3", or $x_1 = 1$ because its value (5.1) lies within the third interval of $[5.012000, 5.388889]$ of the first column in the RPM; the second regressor "Sepal.Width" a "7", or $x_2 = 1$ because its value (3.5) corresponded with the 7th interval of the second column in the RPM; the third regressor "Petal.Length" a "2", or $x_3 = 2$ because its value (1.4) corresponded with the second interval of the third column; and the final regressor "Petal.Width" a "2", or $x_4 = 2$ because its value (0.2) corresponded with the second interval of the fourth column. This set of independent variables corresponded with a "setosa" species. If a new set of measurements matches this partition configuration, NNS will classify it as "setosa". The reader may immediately recognize the measurement matching technique, as it will generally return the same output as the k-Nearest Neighbors algorithm when $k = 1$. However, the goal of this paper is to demonstrate NNS as a form of 'unsupervised learning,' or clustering. The supervised learning, or classification / multivariate regression features of NNS based on these clusters suggests a strong relationship between NNS and kNN, with notable differences in cluster distance weighting and the lack of restricted cluster sizes for NNS. We leave the full NNS-kNN exposition for another paper.

As a further illustration, let us focus on the 5th line of Table 2 with "NNS identifier" value "2.8.2.2" means the first 'iris' regressor "Sepal.length" was assigned a "2", or $x_1 = 2$ because its value corresponded with the 2nd unique interval of the first column in the RPM; the second regressor "Sepal.Width" was assigned a "8", or $x_2 = 8$ corresponding with the 8th interval of the second column in the RPM; the third regressor "Petal.Length" a "2", or $x_3 = 2$; and the final regressor "Petal.Width" a "2", or $x_4 = 2$. This set of independent variables corresponded with a "setosa" species. If a new set of measurements matches this partition configuration, NNS will classify it as "setosa".

Finally let us focus on the last 150-th line of Table 2 with the "NNS identifier" as "5.5.11.12". It simply means that the first 'iris' regressor "Sepal.length" was assigned a "5", or $x_1 = 5$ because its value corresponded with the 5th interval of the first column in the RPM; the second regressor

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 10 January 2018 doi:10.20944/preprints201801.0090.v1

"Sepal.Width" equals "5", or $x_2 = 5$ corresponding with the 5th interval of the second column in the RPM; the third regressor "Petal.Length" equals "11", or $x_3 = 11$; and the final regressor "Petal.Width" equals "12", or $x_4 = 12$. This set of independent variables corresponded with a "virginica" species. If a new set of measurements matches this partition configuration, NNS will classify it as "virginica". The full table is obtained by a one-line R command from the NNS package provided in the Supplementary Material A.

Table 2. Multivariate NNS partitioning and NNS.ID assignments for 'iris' dataset for $O = 4$

i	y_i	NNS identifier	fitted value
1:	1	3.7.2.2	1
2:	1	2.5.2.2	1
3:	1	2.6.1.2	1
4:	1	1.5.2.2	1
5:	1	2.8.2.2	1
—			
146:	3	7.5.11.14	3
147:	3	6.2.11.12	3
148:	3	7.5.11.13	3
149:	3	6.7.12.14	3
150:	3	5.5.11.12	3

Table 2 shows that the categorical variable y and the fitted value $\hat{y}$ are identical, as does the R^2 Figure 6, proving a perfect fit in a finite order by NNS for the iris data.

4.1. Additional Visualizations

We present a univariate and multivariate 3d visualization of the progression of NNS fitting.

4.1.1. Univariate

The following R code is used to generate an NNS fit of increasing orders on a periodic sine wave.

```
x=seq(0,4*pi,pi/1000);y=sin(x)
for(i in 1:6){NNS.reg(x,y,order=i,noise.reduction='off')}
```

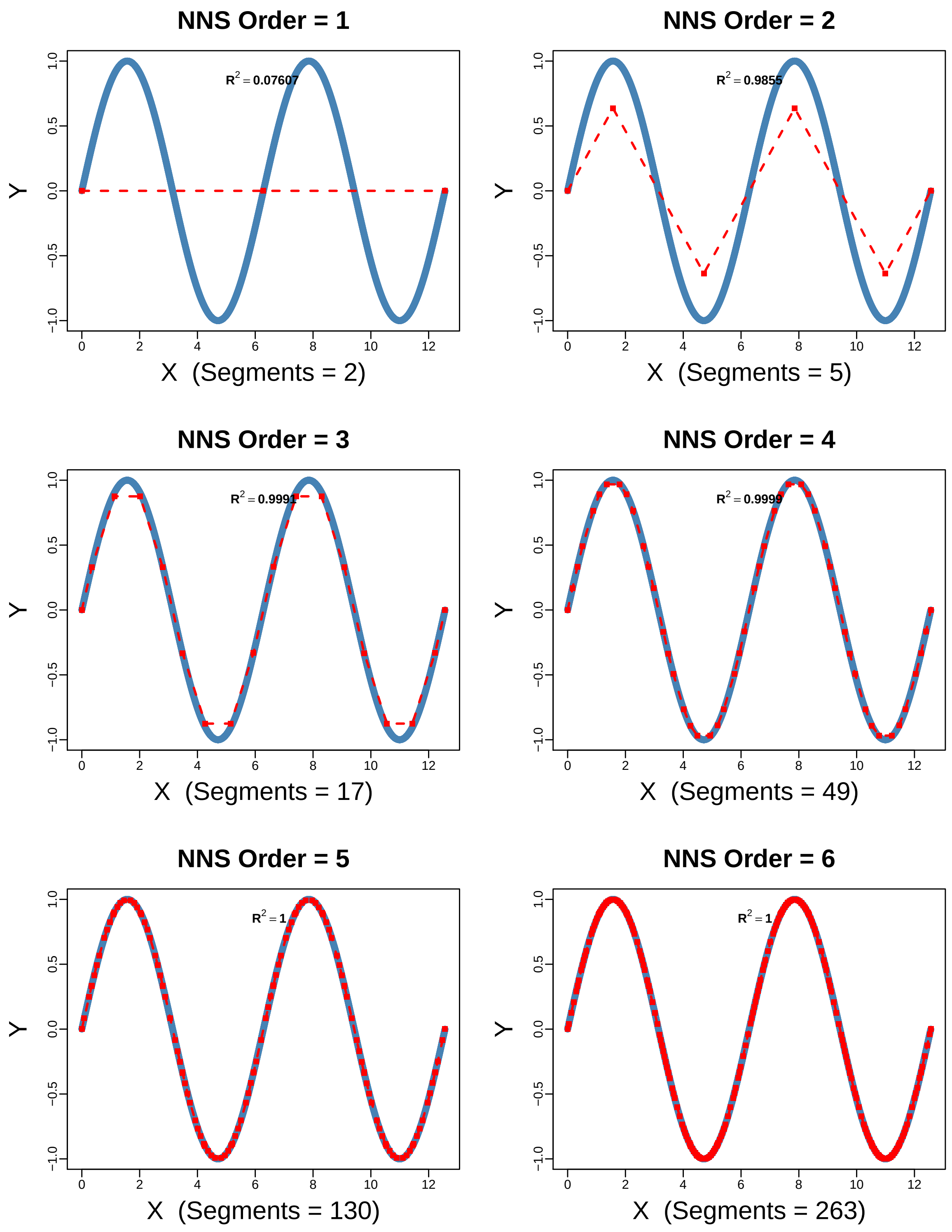

Figure 7. Progression of O in univariate NNS curve fitting of periodic sine wave.

4.1.2. Multivariate

The following R code is used to generate an NNS fit of increasing orders on a nonlinear multivariate function.

```
f <- function(x, y) x^3+3*y-y^3-3*x
x <- seq(-5, 5, 0.1);y <- seq(-5, 5, 0.1)
```

```
z <- expand.grid(x,y)
g <- f(z[,1],z[,2])
for(i in 1:6){NNS.reg(z,g,order=i)}
```

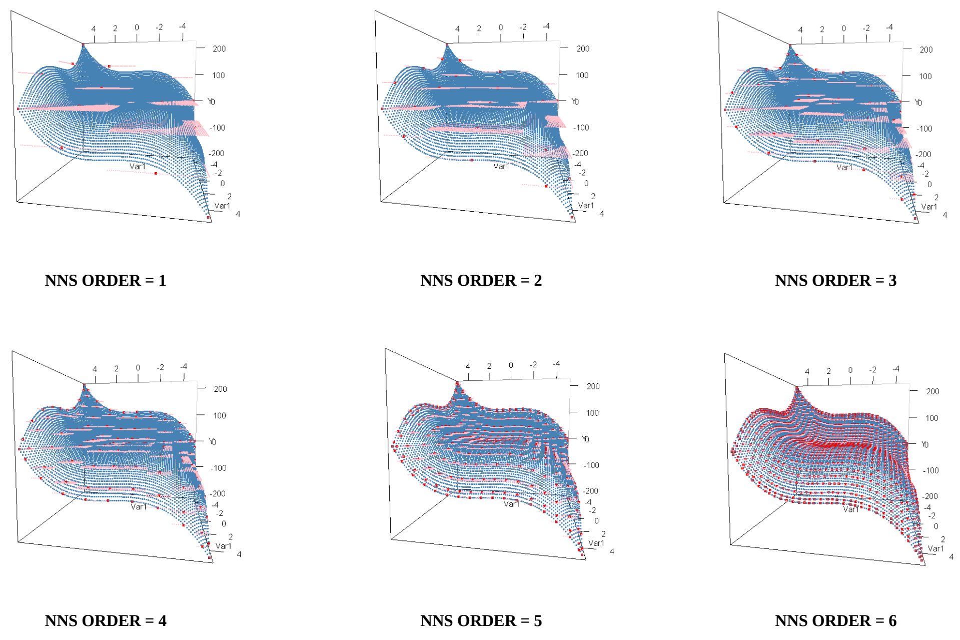

Figure 8. Progression of O in multivariate NNS curve fitting of nonlinear function. Red dots are NNS quadrant means and pink "plates" are regressor regions covered by specific NNS quadrant.

5. Conclusion

In conclusion, curve fitting is an old and important problem with a long history. High order polynomials and kernel regressions are intrinsically approximate. This paper uses artificial univariate $y = f(x)$ data and the well-known multivariate iris data to illustrate and display how a newer NNS algorithm reaches a perfect fit while using quadrants as clusters along the way. We offer analogies to existing techniques such as k-means and codevectors when defining NNS clusters and explain the differences in objective functions / initial parameter specifications. We also illustrate quadrant identifiers internal to NNS for univariate and multivariate cases which help speed the algorithm.

The perfect fit in finite steps is not claimed to be directly usable in practice since some noise is always present. In sampling theory sample size is sometimes determined after specifying the error rate, which is possible because the perfect result (exact value of the population parameter) is potentially available. We are currently working on extending NNS to nonparametric regressions, providing a similar capability to first choose an error rate and then obtain a nonparametric fit. All this is now feasible due to the perfect fit in relatively simple finite steps described here. In any case, NNS fills a gap in the nonparametric regression literature by offering flexibility which in turn yields high quality interpolation and extrapolation from a nonparametric fit illustrated in [1]. The NNS quadrant identifiers serve the same purpose as vector quantization codewords, and the resulting NNS identifier matrix serves the same purpose as vector quantization codebooks. Thus NNS shares a lot of features with some very robust long-standing techniques while offering a slew of additional features in a convenient R package.

Appendix. Supplemental R-Code

We provide all of the R-code used to produce the above examples and plots. Since *k*-means is not deterministic, the clustering examples may not replicate exactly to what is presented above but should still sufficiently highlight the differences in objective functions. Note that the typical code involves very few lines.
Clustered Dataset:

```
n = 100;g=6;set.seed(g)
d <- data.frame(x = unlist(lapply(1:g, function(i) rnorm(n/g, runif(1)*i^2))),
y = unlist(lapply(1:g, function(i) rnorm(n/g, runif(1)*i^2))))
```

Figure 1 R-commands:

```
require(NNS); require(clue)
par(mfrow=c(2,1))
plot(d, col = kmeans(d,1)$cluster, main=paste("k-means (k=",1,")",sep = ""),
cex.main=2)
points(kmeans(d,1)$centers,pch=15,col=1)
NNS.part(d$x,d$y,order=1,Voronoi = T)
```

Figure 2 R-commands:

```
par(mfrow=c(2,1))
plot(d, col = kmeans(d,3)$cluster, main=paste("k-means (k=",3,")",sep = ""),
cex.main=2)
points(kmeans(d,3)$centers,pch=15,col=1:3)
NNS.part(d$x,d$y,order=2,Voronoi = T,noise.reduction = 'off')
```

Figure 3 R-commands:

```
par(mfrow=c(2,1))
plot(d, col = kmeans(d,8)$cluster, main=paste("k-means (k=",8,")",sep = ""),
cex.main=2)
points(kmeans(d,8)$centers,pch=15,col=1:8)
NNS.part(d$x,d$y,order=3,Voronoi = T,noise.reduction = 'off')
```

Univariate Data Example R-command:

```
set.seed(345);x=sample(1:90);y1=sort(x[1:25]);y2=sort(x[1:10]);y=c(y1,y2);xx=1:35
```

Figure 4 R-commands:

```
par(mfrow=c(3,2))
for(i in 1:3){NNS.part(xx,y,order=i,Voronoi=T,noise.reduction = 'off');
NNS.reg(xx,y,order=i,noise.reduction = 'off')}
```

Figure 5 R-commands:

```
par(mfrow=c(3,2))
for(i in 1:5){NNS.reg(xx,y,order=i,noise.reduction = 'off')}
```

Figure 6 R-commands:

```
par(mfrow=c(2,2))
for(i in 1:4){NNS.reg(iris[,1:4],iris[,5],order=i)}
```

RPM R-commands:

```
head(NNS.reg(iris[,1:4],iris[,5],order=4)$rhs.partition)
```

Table 1 R-commands:

```
order=matrix(nrow=35,ncol=5)
colnames(order)=c("order.1","order.2","order.3","order.4","order.5")
for(i in 1:5){order[,i]=
NNS.reg(xx,y,order=i,noise.reduction = 'off')$Fitted.xy[,NNS.ID]}
cbind(NNS.reg(xx,y,order=5,noise.reduction = 'off')$Fitted.xy[,.(y,NNS.ID)],order)
```

Table 2 R-commands:

```
NNS.reg(iris[,1:4],iris[,5],order=4)$Fitted.xy[,.(y,NNS.ID,y.hat)]
```

Even though the paper does not discuss extrapolation and interpolation of fitted nonparametric regression functions in detail, the following code illustrates the ease with which these tasks are implemented. (i) Interpolation and extrapolation for artificial data are illustrated by following commands:

```
NNS.reg(xx,y,noise.reduction = NULL,point.est = 25.5)
NNS.reg(xx,y,noise.reduction = NULL,point.est = 36)
```

(ii) Illustrative Iris data interpolation and extrapolation commands are:

```
NNS.reg(iris[,1:4],iris[,5],point.est = (iris[1,1:4]+.01))$Point.est
NNS.reg(iris[,1:4],iris[,5],point.est = (iris[150,1:4]+1))$Point.est
```

1. Vinod, H.D.; Viole, F. Nonparametric Regression Using Clusters. *Computational Economics, June 19, 2017* **2017**.
2. Viole, F.; Nawrocki, D. Deriving Nonlinear Correlation Coefficients from Partial Moments. *SSRN eLibrary* **2012**.
3. Hayfield, T.; Racine, J.S. Nonparametric Econometrics: The np Package. *Journal of Statistical Software* **2008**, *27*, 1–32.
4. Kohonen, T. The Self-organizing Map. *Proceedings of the IEEE* **1990**, *78*, 1464–1480.
5. Grbovic, M.; Vucetic, S. Regression Learning Vector Quantization. *IEEE International Conference on Data Mining (ICDM)* **2009**, pp. 788–793.
6. Viole, F. *NNS: Nonlinear Nonparametric Statistics*, 2016. R package version 0.3.6.
7. Viole, F. Beyond Correlation: Using the Elements of Variance for Conditional Means and Probabilities. *SSRN eLibrary* **2016**.
8. Bellman, R. On the approximation of curves by line segments using dynamic programming. *Communications of the ACM* **1961**, *4*, 284.