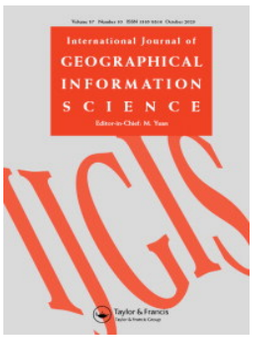

International Journal of Geographical Information Science

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tgis20

Gradient-based optimization for multi-scale geographically weighted regression

Xiaodan Zhou, Renato Assunção, Hu Shao, Cheng-Chia Huang, Mark Janikas & Hanna Asefaw

To cite this article: Xiaodan Zhou, Renato Assunção, Hu Shao, Cheng-Chia Huang, Mark Janikas & Hanna Asefaw (2023) Gradient-based optimization for multi-scale geographically weighted regression, International Journal of Geographical Information Science, 37:10, 2101-2128, DOI: 10.1080/13658816.2023.2246154

To link to this article: https://doi.org/10.1080/13658816.2023.2246154

INTERNATIONAL JOURNAL OF GEOGRAPHICAL INFORMATION SCIENCE
2023, VOL. 37, NO. 10, 2101–2128
https://doi.org/10.1080/13658816.2023.2246154

Taylor & Francis
Taylor & Francis Group

RESEARCH ARTICLE

Gradient-based optimization for multi-scale geographically weighted regression

Xiaodan Zhou[a]*, Renato Assunção[a,b], Hu Shao[a], Cheng-Chia Huang[a], Mark Janikas[a] and Hanna Asefaw[a]

[a]ESRI Inc, Redlands, CA, USA; [b]Departamento de Ciência da Computação, Universidade Federal de Minas Gerais, Belo Horizonte, Brazil

ABSTRACT

Multi-scale geographically weighted regression (MGWR) is among the most popular methods to analyze non-stationary spatial relationships. However, the current model calibration algorithm is computationally intensive: its runtime has a cubic growth with the sample size, while its memory use grows quadratically. We propose calibrating MGWR with gradient-based optimization. This is obtained by analytically deriving the gradient vector and the Hessian matrix of the corrected Akaike information criterion (AICc) and wrapping them with a trust-region optimization algorithm. We evaluate the model quality empirically. Our method converges to the same coefficients and produces the same inference as the current method but it has a substantial computational gain when the sample size is large. It reduces the runtime to quadratic convergence and makes the memory use linear with respect to sample size. Our new algorithm outperforms the existing alternatives and makes MGWR feasible for large spatial datasets.

ARTICLE HISTORY
Received 28 February 2023
Accepted 5 August 2023

KEYWORDS
Spatial analysis; spatially varying coefficients; large spatial dataset; scalability

Introduction

A common problem in spatial analysis is to understand non-stationary spatial processes. In these processes, the relationship between covariates and the response variable changes spatially, implying that the effect of a covariate varies depending on the area location (Assunção 2003, Brunsdon *et al*. 2010). The coefficient of a covariate is not uniquely defined but varies from area to area. Multiple schemes have been proposed to analyze a non-stationary spatial process, such as Bayesian spatially varying coefficient models (Assunção 2003, Gelfand *et al*. 2003), eigenvector spatial filtering (Murakami and Griffith 2015), geographically weighted regression (GWR) (Brunsdon *et al*. 1998, 2010, Fotheringham *et al*. 2003), and GWR's recent enhancement, the multi-scale geographically weighted regression (MGWR) (Fotheringham *et al*. 2017, Yu *et al*. 2020).

CONTACT Renato Assunção rassuncao@esri.com
*Xiaodan Zhou is now affiliated with the Department of Statistics, North Carolina State University, Raleigh, NC, USA.

GWR and MGWR are very popular due to their intuitive assumptions, interpretability, and accessibility, with several software implementations available. They both assume that the relationship between the multiple covariates and the response variable varies smoothly in space. A weighted local regression model is fitted locally with weights determined by a geographic distance decay function. This decay function is based on a kernel function with a specific bandwidth representing the spatial range of the spatial process. While GWR assumes a single bandwidth for all covariates, MGWR allows each covariate to have its own bandwidth, representing the range of its spatial influence. This is a reasonable assumption, as the influence of a covariate could be highly localized or manifest globally. Since its inception in 2017, MGWR has been used to analyze a diverse set of problems such as the incidence of COVID-19 (Iyanda *et al*. 2020, Mollalo *et al*. 2020, Maiti *et al*. 2021), the spatial determinants of housing price (Shen *et al*. 2020), obesity rate (Oshan *et al*. 2020), mortality rate (Cupido *et al*. 2021), opioid overdose death risk (Forati *et al*. 2021), presence of urban heat islands (Niu *et al*. 2021), ecosystem services (Tran *et al*. 2022), and alcohol control (De Ridder *et al*. 2022).

Although MGWR has shown promise, one limitation is the intensive computational resources, specifically runtime and memory use, required for bandwidth selection and the calculation of statistical inference. The existing software implementation of MGWR uses a back-fitting algorithm, that does not scale well for large datasets (Li and Fotheringham 2020). The runtime grows quadratically with sample size during the bandwidth selection phase, while memory use grows linearly. In addition, the runtime grows cubically during the statistical inference phase, while memory use grows quadratically with sample size. As a result, even with the best available implementation, it requires four days of execution time and 11 GB of memory to fit a dataset of 40,000 points using ten covariates (Li and Fotheringham 2020).

The computational burden of MGWR has received considerable attention in the literature recently. The back-fitting procedure adopted by MGWR requires solving several iterative univariate GWR problems. Currently, Li *et al*. (2019) provide the fastest and most memory-efficient implementation of the GWR algorithm. Using a single desktop computer, it can process 200,000 samples in approximately one hour. Murakami *et al*. (2021) proposed a linear-time algorithm for GWR through a pre-compression of the matrices and vectors whose size depends on the sample size. This method was implemented in the R package 'GWmodel'. Despite the speed enhancement over the method proposed by Li *et al*. (2019), it trades off increased memory use for faster computing. As a result, the memory use scales up so quickly that the method will not complete for 10,000 samples and ten covariates using a standard personal computer (Li and Fotheringham 2020).

Li and Fotheringham (2020) have implemented the fastest current back-fitting solution for MGWR. It can run MGWR with 40,000 samples in four days with a standard desktop computer. Wu *et al*. (2022) proposed an efficient algorithm for MGWR that speeds up the bandwidth selection phase. However, even though it is faster than the Li and Fotheringham (2020) procedure, its time complexity with respect to the sample size remains the same, with a $n^2 \log(n)$ complexity. Shahneh *et al*. (2021) proposed the idea of Stateless MGWR, which leverages a broader class of optimization methods

for bandwidth selection, and hence shares a core idea with our proposal. However, it uses layered optimization algorithms: the bandwidths are optimized with a simulated annealing algorithm, then further optimized with a hill-climbing algorithm. As a result, this method has a complex structure and does not have theoretical guarantees for runtime enhancement. Most importantly, all previous research has focused on the bandwidth selection phase, without addressing the performance of the statistical inference phase.

Gradient-based methods are among the most efficient optimization procedures, typically having a quadratic convergence rate (Nocedal and Wright 2006), but they have not been used in the MGWR optimization algorithms. One of the reasons for this absence is the difficulty in deriving the required gradient and Hessian matrix. Notwithstanding this difficulty, there are two main reasons to consider the possibility of using a gradient-based algorithm to fit an MGWR model. Firstly, the golden section search used in the back-fitting algorithm is not as efficient as a gradient-based algorithm, the former having linear convergence while the latter has quadratic convergence (Nocedal and Wright 2006). Secondly, statistical inference using the back-fitting method is so computationally and memory-use intensive that it is impractical with very large datasets. We propose a new gradient-based solution for MGWR calibration that addresses both bandwidth selection and statistical inference. We show how to obtain the analytic expressions for the gradient and Hessian matrix associated with the MGWR corrected Akaike information criterion (AICc). We propose a gradient-based method to obtain the MGWR bandwidths and the regression coefficient estimates based on these gradient and Hessian expressions.

To present our proposal, we start in Section "MGWR formulation and calibration" by reviewing the MGWR formulation and notation and by discussing the performance of the best available solutions. Next, we introduce our gradient-based MGWR deriving the analytic expressions to optimize the AICc criterion in Section "Gradient-based optimization for MGWR calibration". Finally, We conduct an extensive simulation study in the Section "Theoretical performance of gradient-based MGWR" to compare our solution to existing implementations. We show that our solution retrieves the same parameters as the alternative algorithms but with considerably less computational cost in large datasets. In Section "Real data analysis" we illustrate the use of our method with two real-world datasets.

MGWR formulation and calibration

GWR formulation

Suppose a map is partitioned into n areas indexed by $i = 1, \ldots, n$. We have a response variable y_i and a set of potential explanatory variables collected into the vector $\boldsymbol{x}_i$ for each area. GWR aims to find the spatially varying regression coefficients between the response and explanatory variables. We assume a local regression model $y_i = \boldsymbol{x}_i\beta_i + \varepsilon_i$, where ε_i denotes the error term. The area-specific regression coefficient β_i is estimated by the usual weighted least squares method:

$$\hat{\boldsymbol{\beta}}_i = (\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{Y} \tag{1}$$

where $\boldsymbol{Y}$ is the $n \times 1$ response variable vector and $\boldsymbol{X}$ is the design matrix of size $n \times k$ with the first column composed of the constant 1 indicating the intercept. The area-specific $\boldsymbol{W}_i$ is an $n \times n$ diagonal matrix, with the diagonal elements denoting the weight assigned to each area in the map. Given a scalar bandwidth r value, $\boldsymbol{W}_i = \text{diag}[K_r(d_{i1}), \ldots, K_r(d_{in})]$, with d_{ij} denoting the distance between the areas i and j and $K_r()$ being a kernel function. The most commonly used kernel functions are the exponential ($K_r(d) = \exp(-d/r)$), the Gaussian ($K_r(d) = \exp(-d^2/(2r^2))$), and the bi-square ($K_r(d) = (1 - (d/r)^2)^2 \mathbb{I}(d < r)$) kernels. The fitted value in area i is given by $\hat{y}_i = \boldsymbol{x}_i' \hat{\beta}_i$.

Typically, the GWR bandwidth r is selected by minimizing the corrected Akaike information criterion (AICc) given by:

$$\text{AICc}(r) = n \log(\hat{\sigma}^2(r)) + n \log(2\pi) + n\left\{\frac{n + \text{tr}(\boldsymbol{S})}{n - 2 - \text{tr}(\boldsymbol{S})}\right\}, \tag{2}$$

where $\hat{\sigma}^2(r)$ is the normalized residual sum of squares defined by $\hat{\sigma}^2(r) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2/(n - \boldsymbol{tr}(\boldsymbol{S}))$, $\boldsymbol{tr}(\boldsymbol{S})$ is the trace of $\boldsymbol{S}$, and $\boldsymbol{S}$ is the $n \times n$ hat matrix satisfying $\hat{\boldsymbol{Y}} = \boldsymbol{SY}$ and given by

$$\boldsymbol{S} = \begin{bmatrix} \boldsymbol{h}_1 \\ \boldsymbol{h}_2 \\ \vdots \\ \boldsymbol{h}_n \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}_1'(\boldsymbol{X}'\boldsymbol{W}_1\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_1 \\ \boldsymbol{x}_2'(\boldsymbol{X}'\boldsymbol{W}_2\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_2 \\ \vdots \\ \boldsymbol{x}_n'(\boldsymbol{X}'\boldsymbol{W}_n\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_n \end{bmatrix} \tag{3}$$

GWR calibration

Current implementations of MGWR use a golden section search to find the optimal bandwidth r (Li *et al*. 2019). In contrast with the quadratic rate of the gradient-based methods, golden section search has a linear convergence rate (Sauer 2018, p. 596). The computational cost of the model calibration is $O(k^2 n^2 \log(n))$, including the following nested procedures: (1) $O(\log(n))$ is required for the golden section search iterations to find the optimal r bandwidth; (2) to calculate $\hat{\boldsymbol{\beta}}_i$ in 1, we need $(\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_i$ in Equation (3), which requires $O(k^2 n)$ operations; (3) finally, the algorithm loops over the n areas to calculate $\hat{\boldsymbol{\beta}}_i$. As for memory use, storing the hat matrix for inference in Equation (3) takes $O(n^2)$ space memory. Li *et al*. (2019) have reduced the memory use to $O(nk)$ by simplifying the inference calculation.

MGWR formulation

MGWR assumes that the scale of the underlying spatial process being modeled varies according to the covariates. Rather than assuming a single value, MGWR has a vector of bandwidths $\boldsymbol{r} = (r_1, r_2, \ldots, r_k)$ with r_i associated with $\boldsymbol{x}_i$, the i-th covariate. Fotheringham *et al*. (2017) and Yu *et al*. (2020) reframe MGWR as a generalized additive model (GAM) and use the back-fitting algorithm to fit the model. GAM assumes an additive structure between smoothed predictor variables,

$$\boldsymbol{y} = \sum_{j=1}^{k} \boldsymbol{f}_j(\boldsymbol{x}_j) + \epsilon \tag{4}$$

where $\boldsymbol{f}_j$ is a smooth function of $\boldsymbol{x}_j$ approximated by splines (Hastie and Tibshirani 1990). In MGWR, $\boldsymbol{f}_j$ is a spatial smoothing surface function of $\boldsymbol{x}_j$. The bandwidth r_j indicates the level of geographical smoothing in the $\boldsymbol{f}_j(\boldsymbol{x}_j)$ surface, with a larger bandwidth producing a smoother surface.

The optimal bandwidth vector **r** minimizes the corrected Akaike information criterion (2). Following Yu *et al.* (2020), the fitted response vector is expressed as $\hat{\boldsymbol{Y}} = \boldsymbol{S}(\boldsymbol{r})\boldsymbol{Y} = (\boldsymbol{R}_1(r_1) + \boldsymbol{R}_2(r_2) + \ldots + \boldsymbol{R}_k(r_k))\boldsymbol{Y}$, where $\boldsymbol{R}_j(r_j)$ is associated with the weights determined by the r_j-th bandwidth:

$$\boldsymbol{R}_j(r_j) = \begin{bmatrix} x_{1j}\boldsymbol{e}_j(\boldsymbol{X}'\boldsymbol{W}_1(r_j)\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_1(r_j) \\ x_{2j}\boldsymbol{e}_j(\boldsymbol{X}'\boldsymbol{W}_2(r_j)\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_2(r_j) \\ \vdots \\ x_{nj}\boldsymbol{e}_j(\boldsymbol{X}'\boldsymbol{W}_n(r_j)\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_n(r_j) \end{bmatrix} \tag{5}$$

$\boldsymbol{e}_j$ is a $1 \times k$ one-hot encoding row-vector with $\boldsymbol{e}_j[l] = 0$, if $l \neq j$, and $\boldsymbol{e}_j[l] = 1$, if $l = j$.

MGWR calibration and performance issues

Table 1 provides a short description of the back-fitting algorithm applied to MGWR. After initiating the additive model by GWR (step 0), this algorithm refines the surfaces $\boldsymbol{f}_1(\boldsymbol{x}_1), \boldsymbol{f}_2(\boldsymbol{x}_2), \ \ldots, \boldsymbol{f}_k(\boldsymbol{x}_k)$ iteratively (steps 1 and 2). The refinement is done by finding the optimal bandwidth r_j to minimize the AICc while controlling for the other $r_1, \ldots, r_{j-1}, r_{j+1}, \ldots, r_k$ bandwidths (step 3). We obtain the partial residuals $\boldsymbol{y} - \sum_{l \neq j} \hat{\boldsymbol{f}}_l$ and regress it using GWR against the j-th covariate $\boldsymbol{X}_j$ obtaining an updated surface $\hat{\boldsymbol{f}}_j = \boldsymbol{A}_j\boldsymbol{X}_j$ where $\boldsymbol{A}_j$ is the hat matrix of the partial regression model. Finally, the hat matrix **S** is updated.

The bandwidth selection (step 3) requires a computational complexity of $O(kdn^2 \log(n))$ with the following nested procedures: (1) $O(d)$ iterations until we reach convergence with the back-fitting algorithm; (2) in each iteration, $O(k)$ operations looping over covariates to update a single bandwidth; (3) to update a single bandwidth, univariate GWR is computed using golden section search with $O(n^2 \log(n))$ complexity. Statistical inference (steps 4 and 5) is even more expensive than bandwidth selection, requiring $O(kdn^3)$ operations due to the expensive matrix multiplication of two $n \times n$ matrices. Memory use is also intensive for inference. For example, storing the hat matrix **S** and the additive components of the hat matrix $\boldsymbol{R}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_k$ for statistical inference requires $O(kn^2)$ of memory space.

Gradient-based optimization for MGWR calibration

We estimate the bandwidth vector **r** using a gradient-based optimization algorithm. After obtaining the optimal **r** bandwidth vector, we follow the idea of the generalized additive model for statistical inference. The hat matrix is derived from **r** using Equation (5), rather than by the expensive iterations in Table 1. This means that

Table 1. Key steps in back-fitting MGWR and its required number of operations.

Back-fitting MGWR	
0: initiate $\hat{\boldsymbol{f}}_{1\ldots k}, \hat{\epsilon}, \boldsymbol{R}_{1\ldots k}, \boldsymbol{S}$ from GWR $\{\boldsymbol{Y} \sim \boldsymbol{X}\}$	
1: do until $\boldsymbol{f}_{1\ldots k}$ converge:	$O(d)$
2: for each term j from 1 to k:	$\times O(k)$
3: $\hat{\boldsymbol{f}}_j, \hat{\epsilon}, \boldsymbol{A}_j \leftarrow$ GWR $\{\hat{\boldsymbol{f}}_j + \hat{\epsilon} \sim \boldsymbol{X}_j\}$ using optimal r_j	$\times [O(n^2 \log(n))$
4: $\boldsymbol{R}_j \leftarrow \boldsymbol{A}_j \boldsymbol{R}_j + \boldsymbol{A}_j - \boldsymbol{A}_j \boldsymbol{S}$	$+O(n^3)$
5: $\boldsymbol{S} \leftarrow \sum_1^k \boldsymbol{R}_j$	$+O(n^2)]$

storing the $n \times n$ hat matrix is unnecessary, thus substantially decreasing the memory requirements, as we explain in the Section "Statistical inference".

The current section discusses our choice of an optimization framework. First, we propose our final choice of the trust-region optimization algorithm by discussing the rules used to evaluate candidate algorithms (Section "Bandwidth selection"). Next, we show how statistical inference is conducted in a fast and memory-efficient way (Section "Statistical inference"). Finally, we derive the gradient vector and Hessian matrix of MGWR (Section "Gradient-based method for MGWR").

We derive the gradient vector and the Hessian matrix of AICc with respect to **r** in two scenarios. In the first scenario, we have bandwidth **r** as a vector in Euclidean space, with each coordinate representing a straight distance in the geographical units used in the map. This is the fixed bandwidth scenario, and it means that we can take the derivative of AICc with respect to **r**. The second one, called the adaptive bandwidth scenario, uses an integer number of nearest neighbors to specify the bandwidth. This means that the distance scale for each covariate is different across areas. Handling the adaptive bandwidth is challenging because it is not feasible to take the derivative with respect to discrete variables (number of nearest neighbors). We solve this by building a conversion function between the number of nearest neighbors' distances. With the conversion function, the problem becomes similar to the fixed bandwidth scenario. More details are in Section "The adaptive bandwidth case".

Bandwidth selection

We chose the gradient-based optimization framework by following these rules:

1. The gradient vector must be obtained by an analytical approach. There are two ways to obtain the gradient vector: analytical derivation or numerical approximation. We chose the analytical approach as it usually has a faster convergence, especially in high dimensions.
2. Avoid using the Hessian matrix. Some optimization algorithms allow a Hessian matrix input. As we will show, computing the Hessian matrix is very expensive for MGWR and could negate the speed improvement from reducing iterations. After preliminary experiments, we decide not to use the Hessian matrix.
3. We chose optimization frameworks that allow for box constraints for the bandwidth vector **r**. For example, a model with two covariates can have constraints like $100 \leq r_1 \leq 800, 200 \leq r_2 \leq 1000, 300 \leq r_3 \leq 1200$.

Three candidate algorithms met our requirements for an optimization approach: trust-region, L-BFGS, and Newton conjugate-gradient. We choose the trust-region framework because, in our preliminary experiments, the terminating condition of the trust-region approach performed the best. L-BFGS and Newton conjugate-gradient either stop too early or too late.

Statistical inference

In back-fitting MGWR, the standard errors of the β_{ij} coefficients are obtained using $\boldsymbol{R}_j(r_j)$, the large $n \times n$ matrices in (5):

$$\text{std.err.}(\hat{\beta}_{ij}) = \sqrt{\frac{\left[\boldsymbol{e}_j \boldsymbol{R}_j(r_j)\right]\left[\boldsymbol{e}_j \boldsymbol{R}_j(r_j)\right]'}{x_{ij}^2}}.$$

This requires storing multiple $n \times n$ matrices: $\boldsymbol{R}_1, \boldsymbol{R}_2, \ldots, \boldsymbol{R}_k, \boldsymbol{S}$.

In gradient-based MGWR, after obtaining the best bandwidth, we follow the generalized additive model for statistical inference. However, obtaining the standard errors of coefficients is more straightforward based on equation 5,

$$\text{std.err.}(\hat{\beta}_{ij}) = \sqrt{\left[\boldsymbol{e}_j (\boldsymbol{X}'\boldsymbol{W}_i(r_j)\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_i(r_j)\right]\left[\boldsymbol{e}_j (\boldsymbol{X}'\boldsymbol{W}_i(r_j)\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_i(r_j)\right]'}.$$

By avoiding the $n \times n$ large matrices, the MGWR memory use only grows linearly with sample size.

Gradient-based method for MGWR

We initially discuss the fixed bandwidth case, where r_j is a scalar constant across all the areas. In the Section "The adaptive bandwidth case", we discuss the case where this bandwidth is defined adaptively, meaning that the distance scale for a single covariate is different across areas. The Appendix provides a detailed derivation of the results presented in the main paper.

To minimize $AICc(\boldsymbol{r})$ with a derivative-based method, we need to calculate the gradient vector

$$\frac{\partial AICc(\boldsymbol{r})}{\partial \boldsymbol{r}} = \frac{1}{\hat{\sigma}^2(\boldsymbol{r})}\frac{\partial \text{RSS}(\boldsymbol{r})}{\partial \boldsymbol{r}} + \frac{2n(n-1)}{(n-2-\text{tr}(\boldsymbol{S}(\boldsymbol{r})))^2}\frac{\partial \text{tr}(\boldsymbol{S}(\boldsymbol{r}))}{\partial \boldsymbol{r}} \tag{6}$$

Considering the derivative of RSS $(\boldsymbol{r})$, we have

$$\frac{\partial RSS(\boldsymbol{r})}{\partial \boldsymbol{r}} = -2\sum_i (y_i - \hat{y}_i)\underbrace{\frac{\partial \hat{y}_i}{\partial \boldsymbol{r}}}_{k\times 1} = -2\underbrace{\begin{bmatrix} \left(\frac{\partial \hat{\boldsymbol{Y}}}{\partial r_1}\right)' \\ \vdots \\ \left(\frac{\partial \hat{\boldsymbol{Y}}}{\partial r_k}\right)' \end{bmatrix}}_{k\times n}\underbrace{(\boldsymbol{Y} - \hat{\boldsymbol{Y}})}_{n\times 1} \tag{7}$$

In the Appendix, we show that this gradient can be expressed analytically. The first derivative on the right-hand side is given by

$$\frac{\partial \hat{\boldsymbol{Y}}}{\partial r_j} = \begin{bmatrix} x_{1j}\boldsymbol{e}_j(\boldsymbol{X}'\boldsymbol{W}_1(r_j)\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_1(r_j)\boldsymbol{D}_1(r_j)\left[\boldsymbol{Y}-\hat{\boldsymbol{Y}}_1(r_j)\right] \\ x_{2j}\boldsymbol{e}_j(\boldsymbol{X}'\boldsymbol{W}_2(r_j)\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_2(r_j)\boldsymbol{D}_2(r_j)\left[\boldsymbol{Y}-\hat{\boldsymbol{Y}}_2(r_j)\right] \\ \vdots \\ x_{nj}\boldsymbol{e}_j(\boldsymbol{X}'\boldsymbol{W}_n(r_j)\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_n(r_j)\boldsymbol{D}_n(r_j)\left[\boldsymbol{Y}-\hat{\boldsymbol{Y}}_n(r_j)\right] \end{bmatrix} \tag{8}$$

where $\boldsymbol{D}_n(r_j)$ is a diagonal matrix that depends on the distance kernel function adopted. The derivative of $\mathrm{tr}(\boldsymbol{S}(\boldsymbol{r}))$ is given by

$$\frac{\partial \mathrm{tr}(\boldsymbol{S}(\boldsymbol{r}))}{\partial r_j} = -\sum_{i=1}^{n} x_{ij}\left[(\boldsymbol{X}'\boldsymbol{W}_i(r_j)\boldsymbol{X})^{-1}(\boldsymbol{X}'\boldsymbol{W}_i(r_j)\boldsymbol{D}_i(r_j)\boldsymbol{X})(\boldsymbol{X}'\boldsymbol{W}_i(r_j)\boldsymbol{X})^{-1}\boldsymbol{x}_i\right][j] \tag{9}$$

The Hessian matrix for MGWR

Newton's method requires the evaluation of the $k \times k$ Hessian matrix

$$\boldsymbol{H}(\boldsymbol{r}) = \frac{\partial^2 AICc(\boldsymbol{r})}{\partial \boldsymbol{r}^2} = \frac{\partial}{\partial \boldsymbol{r}}\frac{\partial AICc(\boldsymbol{r})}{\partial \boldsymbol{r}}$$

composed by the second-order partial derivatives of $AICc(\boldsymbol{r})$. The Appendix shows the technical details of how this matrix can be obtained explicitly allowing for its calculation in practice.

The adaptive bandwidth case

This section discusses the use of gradient-based optimization methods when the bandwidth has an adaptive value. To explain our proposal, consider the simpler GWR model when the bandwidth is the same for all covariates. In the adaptive case, rather than a fixed parameter r for all areas (and all covariates), we have an adaptive parameter r_i that varies from area to area. This r_i parameter depends on the spatial density around the i-th area. We select a number v of nearest neighbors and find the area-specific minimum distance $r_i(v)$ needed to reach the v-th nearest neighbor of area i. For areas in dense regions, $r_i(v)$ tends to be a short distance. In regions with sparsely spread areas, $r_i(v)$ tends to be relatively large, as one needs to travel a long distance to find the v nearest neighbors.

The estimated coefficient vector $\hat{\beta}_i$ is found by weighted least squares as in (1) using a kernel function $K_r(d)$. However, unlike the non-adaptive situation, the kernel $K_r(d)$ uses an area-specific radius $r_i(v)$. We can see the weight matrix $\boldsymbol{W}_i$ in (1) as a function of the selected v nearest neighbors. As before, the fitted vector is given by $\hat{\boldsymbol{Y}} = \boldsymbol{S}\boldsymbol{Y}$ where $\boldsymbol{S}$ is the $n \times n$ matrix given by $\boldsymbol{S}$ as in (3). In this adaptive case, we aim to find the number of neighbors v that minimizes the AICc, given previously in (2). $AICc(v)$ is a function of the number of neighbors v instead of fixed bandwidth r. By selecting different values for v, we induce different values for the adaptive bandwidths $r_i(v)$ and the weight matrices $\boldsymbol{W}_i(r_i(v))$. As a consequence, we have different values for $AICc(v)$.

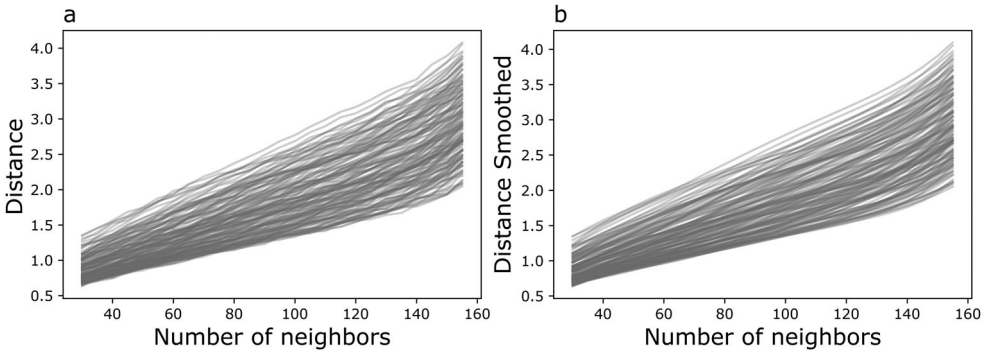

Figure 1. (a) true relationship between distance and the number of nearest neighbors, with one curve per location. (b) smoothed curves using fifth-degree polynomials.

The number of neighbors v is an integer, and hence there is no derivative of $AICc(v)$. However, we can imagine a mathematical interpolation between the discrete points $(v, AICc(v))$, which allows us to use calculus techniques. The Newton method starts with some initial value $v(0)$ and follows the iterative steps given by:

$$v^{(q+1)} = v^{(q)} - \left[\frac{\partial^2 AICc(v)}{\partial v^2}\right]^{-1} \frac{\partial AICc(v)}{\partial v}.$$

The algorithm is run until convergence if we can obtain the first and second derivatives of $AICc(v)$. After finding the best estimate $\hat{v}$, we round it off to obtain an integer value. The gradient is

$$\frac{\partial AICc(v)}{\partial v} = \frac{1}{\hat{\sigma}^2} \underbrace{\frac{\partial \mathrm{RSS}}{\partial v}}_{M} + n \underbrace{\frac{\partial}{\partial v}\left\{\frac{n + \mathrm{tr}(\boldsymbol{S})}{n - 2 - \mathrm{tr}(\boldsymbol{S})}\right\}}_{N} \tag{10}$$

We will consider each of the two derivatives M and N, on the right-hand side of (10). We have

$$M = \frac{\partial \mathrm{RSS}}{\partial v} = -2\sum_i (y_i - \hat{y}_i)\frac{\partial \hat{y}_i}{\partial v} = -2\sum_i (y_i - \hat{y}_i)\frac{\partial \hat{y}_i}{\partial r_i}\frac{\partial r_i}{\partial v} \tag{11}$$

Note that we can only use the chain rule inside the summation when the index i is held fixed at some value, and we then consider the composite function $r_i(v)$. Repeating the steps taken to derive (A4), we show in the Appendix that

$$\frac{\partial \hat{y}_i}{\partial r_i} = \boldsymbol{h}_i \boldsymbol{D}_i(r_i(v))\left(\boldsymbol{Y} - \hat{\boldsymbol{Y}}_i\right) \tag{12}$$

where $\boldsymbol{h}_i$ is defined in (3) and $\hat{\boldsymbol{Y}}_i = \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{W}_i(r_i(v))\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_i(r_i(v))\boldsymbol{Y}$. Note that $\hat{\boldsymbol{Y}}_i$ is not the same as the fitted vector $\hat{\boldsymbol{Y}} = \boldsymbol{S}\boldsymbol{Y}$. It is useful to remark that the derivative is taken with respect to $\hat{y}_i$ and, therefore, with a fixed and arbitrary i-th index.

To complete the calculation of (11), we need the derivative $\partial r_i/\partial v$. Our approach is to build a continuously twice differentiable function $p_i(v)$ interpolating the points $(v, r_i(v))$, where $r_i(v)$ is the distance between the i-th area and its v-th nearest neighbor in the map. To explain the idea using an illustrative example, consider the Georgia

state map partitioned into counties. Figure 1(a) shows a line for each county. Each line connects with straight line segments the points $(v, r_i(v))$ of its associated county. Figure 1(b) shows a smoothly differentiable function approximation $p_i(v)$ based on fifth-degree polynomials for these empirical curves.

We have the area-specific polynomial $p_i(v) = a_{0i} + a_{i1}v + \ldots + a_{i5}v^5$ with simple expressions for its first derivative: $\partial r_i/\partial v \approx p_i'(v) = a_{i1} + \ldots + 5a_{i5}v^4$. The derivative $p_i'(v)$ is a scalar value and hence the final form of M, the derivative of RSS in Equation (10) is given by:

$$M = \frac{\partial \mathrm{RSS}}{\partial v} = -2\sum_i p_i'(v)(y_i - \hat{y}_i)\boldsymbol{h}_i\boldsymbol{D}_i\left(\boldsymbol{Y} - \hat{\boldsymbol{Y}}_i\right). \tag{13}$$

It remains to look at N, the derivative involving the trace $\mathrm{tr}(\boldsymbol{S})$ in Equation (10). We invoke the chain rule once again after passing the derivative operator inside the summation:

$$\begin{aligned}\frac{\partial \mathrm{tr}(\boldsymbol{S})}{\partial v} &= \sum_i \boldsymbol{S}[i,i] = \sum_i \frac{\partial \boldsymbol{x}_i'(\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{X})^{-1}\boldsymbol{x}_i}{\partial r_i}\frac{\partial r_i}{\partial v} \\ &= \textstyle\sum_i \left[\boldsymbol{x}_i'(\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{D}_i\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{X})^{-1}\boldsymbol{x}_i\right]p_i'(v).\end{aligned} \tag{14}$$

Therefore, the derivative of $(n + \mathrm{tr}(\boldsymbol{S}))/(n - 2 - \mathrm{tr}(\boldsymbol{S}))$ with respect to v is given by

$$-\frac{2n(n-1)}{(n-2-\mathrm{tr}(\boldsymbol{S}))^2}\sum_i \left[\boldsymbol{x}_i'(\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{D}_i\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{X})^{-1}\boldsymbol{x}_i\right]p_i'(v).$$

These expressions are enough to obtain the gradient. In the Appendix, we provide more complex expressions for the Hessian for this adaptive bandwidth situation. After finding a real value v that minimizes $AICc(v)$, we round it off to provide an integer number of neighbors as the solution. The expressions for the more general MGWR follow the same arguments as those used to obtain the GWR expressions and are omitted.

Theoretical performance of gradient-based MGWR

Computing cost

The gradient-based algorithm estimates coefficients iteratively. Let us denote the number of iterations as d', which is determined by the efficiency of the trust-region optimization framework. Table 2 highlights the key steps to derive the gradient vector and Hessian matrix. We lay out the most computationally-intensive steps of matrix calculation with big-O denotation. In each iteration (step 1), we iterate over the locations (step 2) and covariates (step 3) to calculate the AICc, gradient vector, and Hessian matrix. The computational cost of this nested procedure is $O(kd'n)$. The procedure to compute AICc is $O(k^2n)$ (step 4). The additional cost is $O(kn)$ (step 5) if we calculate the gradient vector. If we calculate the Hessian matrix, the additional cost is $O(k^2n)$ (step 6), which is much higher than the cost for the gradient vector. Calculating the Hessian matrix is so time-consuming that we did not attempt it in our empirical study. Together, the computational cost of the proposed gradient-based method is $O(k^3d'n^2)$, and improves on the $O(kdn^2\log(n)) + O(kdn^3)$

Table 2. Key procedure to derive gradient vector and Hessian matrix in gradient-based MGWR.

Gradient-based MGWR	
0: initiate bandwidths $\mathbf{r}$ from GWR $\{\mathbf{Y} \sim \mathbf{X}\}$	
1: do until $\mathbf{r}$ convergence:	$O(d)$
2: for each term i from 1 to n:	$\times O(n)$
3:	
$\mathbf{W}_i$	$\times[\mathrm{O(n)}$
$\mathbf{X}'\mathbf{W}_i$ ($\mathbf{W}_i$ is diagonal matrix)	$+O(kn)$
$\mathbf{X}'\mathbf{W}_i\mathbf{X}$	$+O(k^2n)$
$(\mathbf{X}'\mathbf{W}_i\mathbf{X})^{-1}$	$+O(k^3)$
$\mathbf{A}_i := (\mathbf{X}'\mathbf{W}_i\mathbf{X})^{-1}$	
$\mathbf{A}_i\mathbf{X}'$	$+O(k^2n)$
$\mathbf{A}_i\mathbf{X}'\mathbf{W}_i$	$+O(k^2n)\ ]$
4: for each term j from 1 to k:	$\times[\mathrm{O(k)}$
5:	
$\mathbf{D}_i$ (j-th row only)	$+O(n)$
$\mathbf{A}_i\mathbf{X}'\mathbf{W}_i\mathbf{D}_i$ (j-th row only)	$+O(kn)$
6:	
$\mathbf{X}'\mathbf{W}_i\mathbf{D}_i$	$+O(k^2n)$
$\mathbf{B}_i := \mathbf{X}'\mathbf{W}_i\mathbf{D}_i$	
$\mathbf{A}_i\mathbf{B}_i$	$+O(k^2n)$
$\mathbf{A}_i\mathbf{B}_i\mathbf{E}_i$	$+O(k^2n)$
$\mathbf{A}_i\mathbf{B}_i\mathbf{X}$	$+O(k^2n)$
$\mathbf{A}_i\mathbf{B}_i\mathbf{X}\mathbf{A}_i$	$+O(k^3)$
$\mathbf{A}_i\mathbf{B}_i\mathbf{X}\mathbf{A}_i\mathbf{B}_i\mathbf{X}\mathbf{A}_i$	$+O(k^2n)]$
7: pull up together and get $AICc$, $\frac{\partial AICc(\mathbf{r})}{\partial \mathbf{r}}$, $\frac{\partial^2 AICc(\mathbf{r})}{\partial^2 \mathbf{r}}$	
update bandwidths $\mathbf{r}$	

The detailed explanation that justifies this algorithm is given in the Appendix.

cost of the back-fitting algorithm (Section "MGWR calibration and performance issues"). If we treat the number of iterations d and d' as constant numbers, the computational cost ratio between gradient-based and back-fitting algorithms is approximately $k^2 : (\log(n) + n)$.

Memory use

The gradient-based algorithm does not require storing the whole hat matrix. Instead, only a portion of the hat matrix is stored, and all statistics are computed on the fly, such as the standard error of coefficients. The maximum size of the matrix stored is $k \times n$, reducing the memory use of the proposed gradient-based method to $O(kn)$, much less than the $O(kn^2)$ in the back-fitting algorithm (Section "MGWR calibration and performance issues"), with a ratio of $1 : n$.

Empirical study

In this section, we present an empirical study to evaluate the performance of our method. Following Li and Fotheringham (2020), we adopted a data-generating process for deriving the test dataset. In total we have seven datasets, using spatially regular grids of $\{1, 2, 5, 8, 10, 15, 20, 30, 40\}$ thousand points. The i-th point location is denoted $\boldsymbol{s}_i = (s_{1i}, s_{2i})$. In each dataset, we generated ten explanatory variables $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_{10}$ from a ten-dimensional multivariate Gaussian distribution with mean zero, variance 1, and pair correlation of 0.3.

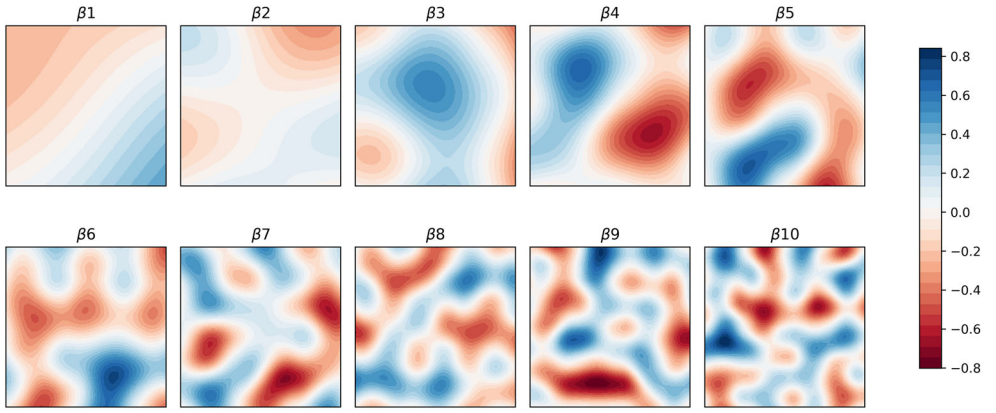

Figure 2. Synthetic coefficient surfaces $\beta_j(\mathbf{s})$ with varying spatial scales from large to small. The *j*-th covariate at the *i*-th location with coordinates $\mathbf{s}_i$ has a spatially varying coefficient β_{ij} given by $\beta_j(\mathbf{s}_i)$, the $\beta_j(\mathbf{s})$ surface height at the position $\mathbf{s}_i$.

We generated ten spatial coefficient surfaces $\beta_1(\boldsymbol{s}), \ldots, \beta_{10}(\boldsymbol{s})$ using Perlin noise (Gustavson 2005), corresponding to the ten explanatory variables. Perlin noise can generate a naturally smoothed surface of pseudo-random numbers at large or small spatial scales. With this technique, we can easily control the scale of the spatial processes.

Figure 2 shows the coefficient surfaces $\beta_1(\boldsymbol{s}), \ldots, \beta_{10}(\boldsymbol{s})$ in a map with 40,000 locations (200×200 grid). Although their range is the same (from –0.8 to 0.8), there is a considerable variation in their spatial scale, with $\beta_1(\boldsymbol{s})$ having a large spatial range variation while $\beta_{10}(\boldsymbol{s})$ has the smallest one. This means that the effect of the x_1 covariate tends to be very different for two relatively close locations, while the x_{10} covariate tends to have approximately the same effect even for pairs of locations that are far apart. The response variable at location $\boldsymbol{s}_i$ is given by $y_i = \sum_{j=1}^{10} x_{ij}\beta_{ij} + \varepsilon_i$, where $\varepsilon_i \sim N(0, 1)$ and x_{ij} is the *i*-th point value for the *j*-th covariate.

We used the bi-square kernel with adaptive bandwidth to fit the MGWR model with an intercept term. We ran the gradient-based and back-fitting algorithms for all datasets. We compared our proposed method to the back-fitting algorithm implemented in MGWR 2.2 (Li and Fotheringham 2020). We compare the two methods with respect to the inferred bandwidths, quality of coefficient estimation, execution time, and memory use.

Back-fitting algorithm implementation

Back-fitting MGWR is available in the Python PySAL package (Oshan *et al*. 2019) and in MGWR 2.2 with a graphical interface (Li and Fotheringham 2020). MGWR 2.2 provides the best computational performance available. It uses parallel computing to reduce execution time, and a chunk-wise partition method to avoid memory overflow. In this paper, we benchmark our method with this software.

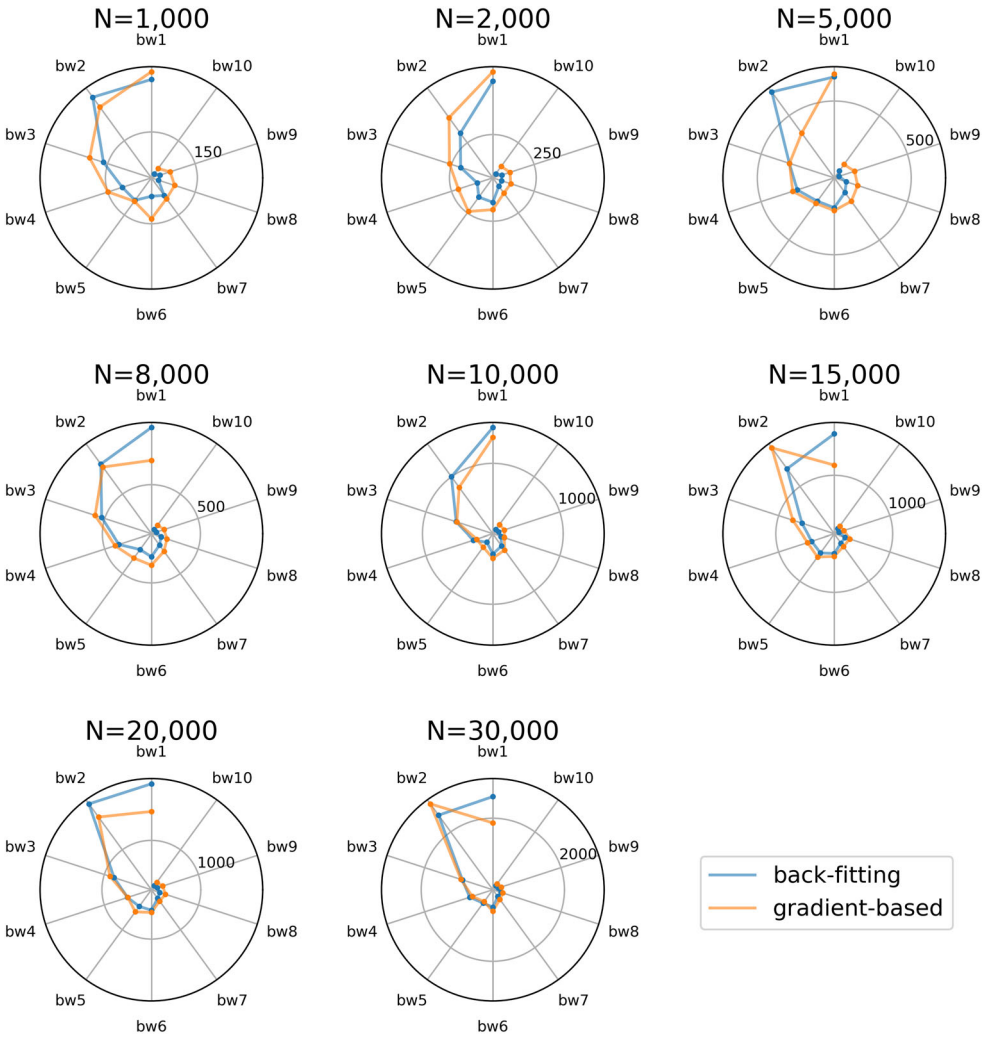

Figure 3. Comparison between the bandwidths selected by the gradient-based and the back-fitting algorithms for different sample sizes. The bandwidths for the covariates are shown in a clockwise way.

Gradient-based algorithm implementation

We implemented our method using the trust-region optimization framework by Byrd *et al.* (1999) and Conn *et al.* (2000) to solve gradient-based MGWR. We rely on the Python Scipy optimization library (https://scipy.org/). We also rely on Numba, a Python module that translates Python into fast machine code and allows for automatically parallelizing the code (https://numba.pydata.org/). Our study activates the automatic parallel option when we have 10,000 samples or more. For datasets smaller than 10,000 units, parallelization is not worthwhile because of its overhead cost. The code to execute the gradient method and reproduce our results is available at the link provided in the end of this paper.

Table 3. Bandwidth $[r_0, r_1, ..., r_{10}]$ selected for each sample size.

Sample size	Method	Bandwidth estimate
1000	Gradient-based	[999, 299, 253, 197, 149, 108, 138, 100, 97, 85, 65]
1000	Back-fitting	[987, 280, 283, 161, 112, 105, 82, 90, 55, 59, 49]
2000	Gradient-based	[510, 565, 406, 261, 214, 241, 192, 127, 127, 123, 104]
2000	Back-fitting	[347, 518, 310, 202, 115, 152, 155, 84, 78, 80, 58]
5000	Gradient-based	[1453, 650, 381, 335, 316, 249, 256, 235, 211, 193, 167]
5000	Back-fitting	[1663, 634, 662, 336, 290, 234, 240, 175, 145, 103, 122]
8000	Gradient-based	[1868, 687, 756, 578, 417, 353, 364, 289, 249, 226, 205]
8000	Back-fitting	[7998, 939, 784, 525, 386, 272, 299, 230, 202, 164, 163]
10,000	Gradient-based	[1504, 1327, 836, 590, 330, 320, 419, 366, 267, 267, 256]
10,000	Back-fitting	[4697, 1452, 1001, 599, 373, 242, 366, 302, 221, 190, 178]
15,000	Gradient-based	[4876, 1146, 1686, 787, 565, 571, 485, 385, 396, 307, 296]
15,000	Back-fitting	[14,999, 1595, 1314, 650, 504, 499, 442, 325, 325, 235, 251]
20,000	Gradient-based	[1992, 1469, 1657, 905, 601, 642, 567, 427, 437, 393, 348]
20,000	Back-fitting	[13,277, 1918, 1921, 834, 605, 533, 531, 372, 339, 298, 281]
30,000	Gradient-based	[3144, 1886, 2846, 1070, 778, 612, 776, 542, 506, 465, 432]
30,000	Back-fitting	[29,366, 2529, 2505, 1032, 844, 654, 676, 458, 458, 388, 375]

The first number represents the bandwidth associated with the intercept. The other 10 values are ordered starting from x_{10} and ending at x_1.

Metrics for coefficient estimation

We evaluate the quality of the coefficient estimation using common metrics. The mean absolute error (MAE) is the mean of $|\hat{\beta}_{ij} - \beta_{ij}|$ for $i = 1, ..., n$ and $j = 1, ..., k$. The root-mean-square error (RMSE) is the square root of the mean of $(\hat{\beta}_{ij} - \beta_{ij})^2$ over all i and j indexes, while the mean absolute percentage error (MAPE) takes the average of $|\hat{\beta}_{ij} - \beta_{ij}|/|\beta_{ij}|$. Since $|\hat{\beta}_{ij} - \beta_{ij}|/|\beta_{ij}|$ is highly sensitive when the denominator is close to zero, we replace it with $|\hat{\beta}_{ij} - \beta_{ij}|/0.01$ when $|\beta_{ij}| < 0.01$.

Testing environment

Our results were obtained using a single desktop machine with Intel(R) Xeon(R) W-2133 at 3.60 GHz with a 6-core CPU and 16.0 GB RAM. We used Python version 3.6. Execution times are measured based on the Python built-in module 'time'. The measure of memory usage is roughly estimated from Windows Task Manager.

Results

Figure 3 shows the bandwidth selected for each covariate by the gradient-based and back-fitting algorithms. Each plot shows the results for a given sample size. The covariates are arranged circularly in each plot, with the bandwidths increasing in the clockwise direction. The estimated bandwidth is given by the segment length along the axis radiating from the circle's center. The exact values used in this figure are in Table 3.

Figure 3 shows that both algorithms capture the spatial pattern of scales. Both methods obtain very similar bandwidth values for all covariates for all sample sizes. The most noticeable but occasional differences occur with the two largest bandwidths, and covariates 1 and 2. If this difference was observed on the small bandwidths, there would be a reason for concern. However, as a large bandwidth means that the associated coefficient surface is flat (see Figure 2) and the kernel weights decrease slowly

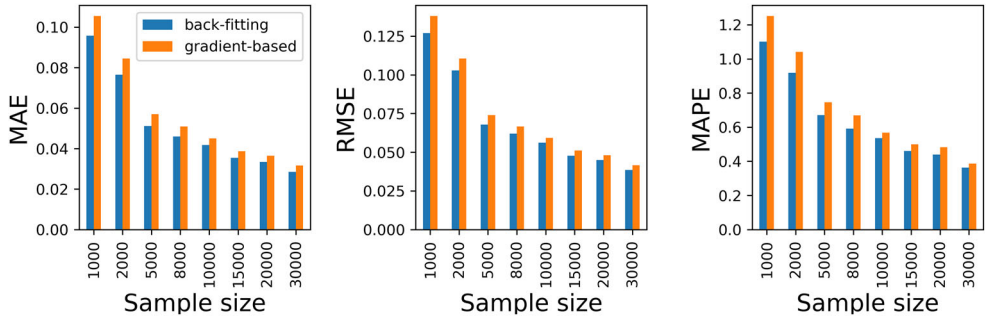

Figure 4. Quality of coefficient estimation comparison between the gradient-based and the back-fitting algorithms.

with the distance, these differences have little impact on the estimation of the spatially varying coefficients, as we will see shortly. In summary, this means that the two algorithms provide essentially the same understanding of the spatial process.

Figure 4 shows the quality of the coefficient estimation using the MAE, RMSE, and MAPE metrics. Table 4 shows the exact values used in this figure and also shows other statistics (AICc, R-squared (R2), and the residual sum of squares (RSS)).

In each plot of Figure 4, we compare the gradient-based to the back-fitting algorithms. In the plots, the error measurement decreases as the sample size increases. The back-fitting algorithm consistently outperforms the gradient-based algorithm in all metrics, but the difference is small. Indeed, considering MAE, the ratio between the gradient-based and back-fitting methods is at least 90%. For RMSE, this ratio is always above 92%, while for MAPE it is 88%. Similarly, looking at the AICc, R^2, and RSS in Table 4, we see that the back-fitting has a slightly better performance than the gradient-based method. However, although the back-fitting algorithm produces better results than the gradient-based algorithm, the relative difference between the methods is quite small and likely not relevant in most situations.

We now analyze the computational performance of the algorithms, the main reason one should consider our gradient-based algorithm. Figure 5 (left) shows their runtime. The back-fitting algorithm is executed in parallel in all cases, while the gradient-based algorithm is executed in parallel only when the sample size is equal to or greater than 10,000 spatial units. The two algorithms require similar runtime when the sample size is less than 10,000. With larger sample sizes, the gradient-based algorithm begins to excel due to parallelization. The back-fitting algorithm takes around 26 h to run on 30,000 samples, whereas the gradient-based method takes 6 hours. When we use 40,000 samples with the back-fitting algorithm, our machine fails due to memory overflow. However, with the gradient-based algorithm, our machine successfully runs in approximately 13 h. The theoretical analysis in Section "Theoretical performance of gradient-based MGWR" shows that the computational complexity of the gradient method grows quadratically with the sample size, while that of the back-fitting algorithm grows cubically. Our empirical results support that theoretical analysis.

Figure 5 (right) shows the memory use of the two algorithms. Our proposed method has a significant advantage in this respect. It uses less than 250 MB, with a

Table 4. Measures of goodness-of-fit for the gradient-based and the back-fitting algorithms.

Sample size	Method	AICc	R2	RSS	MAE	RMSE	MAPE
1000	Gradient-based	2068.2	0.726	274.177	0.106	0.138	1.251
1000	Back-fitting	2060.4	0.773	227.123	0.096	0.127	1.100
2000	Gradient-based	3835.8	0.721	558.315	0.085	0.111	1.041
2000	Back-fitting	3782.3	0.777	445.786	0.077	0.103	0.919
5000	Gradient-based	8858.6	0.732	1341.542	0.057	0.074	0.745
5000	Back-fitting	8784.7	0.746	1268.717	0.051	0.068	0.670
8000	Gradient-based	13,562.2	0.736	2111.387	0.051	0.067	0.670
8000	Back-fitting	13,470.2	0.743	2055.635	0.046	0.062	0.591
10,000	Gradient-based	16,955.8	0.732	2684.886	0.045	0.059	0.568
10,000	Back-fitting	16,817.9	0.738	2620.274	0.042	0.056	0.536
15,000	Gradient-based	25,064.87	0.726	4103.572	0.039	0.051	0.499
15,000	Back-fitting	24,908.753	0.731	4042.444	0.035	0.048	0.461
20,000	Gradient-based	32,759.7	0.731	5372.747	0.036	0.048	0.483
20,000	Back-fitting	32,565.8	0.734	5319.019	0.033	0.045	0.438
30,000	Gradient-based	48,890.56	0.728	8174.561	0.032	0.042	0.387
30,000	Back-fitting	48,682.35	0.728	8154.707	0.029	0.039	0.363
40,000	Gradient-based	64,602.41	0.727	10937.745	0.029	0.038	0.363

We omit the results of the back-fitting algorithm with 40,000 spatial units because we ran into memory overflow on our machine.

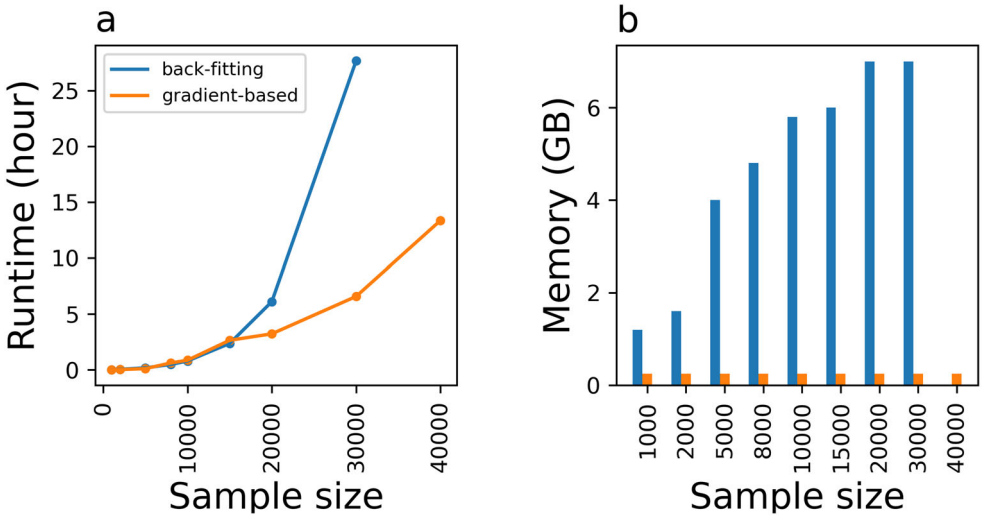

Figure 5. (a) Runtime and (b) memory use comparison between gradient-based and back-fitting algorithm, as the sample size grows.

negligible increment as the sample size increases. In comparison, the memory required by the back-fitting algorithm grows quickly with the sample size, reaching 7 GB if the dataset has 30,000 units.

This sample size was the maximum our test environment supported. We could not run the back-fitting algorithm with 40,000 units. From Section "Theoretical performance of gradient-based MGWR", we know that the memory use of the back-fitting algorithm grows quadratically with the sample size, but the gradient-based method grows linearly. This matches our results here.

The results shown in the previous tables and figures are global because they aggregate the estimates over the entire map and present an omnibus value. Figure 6

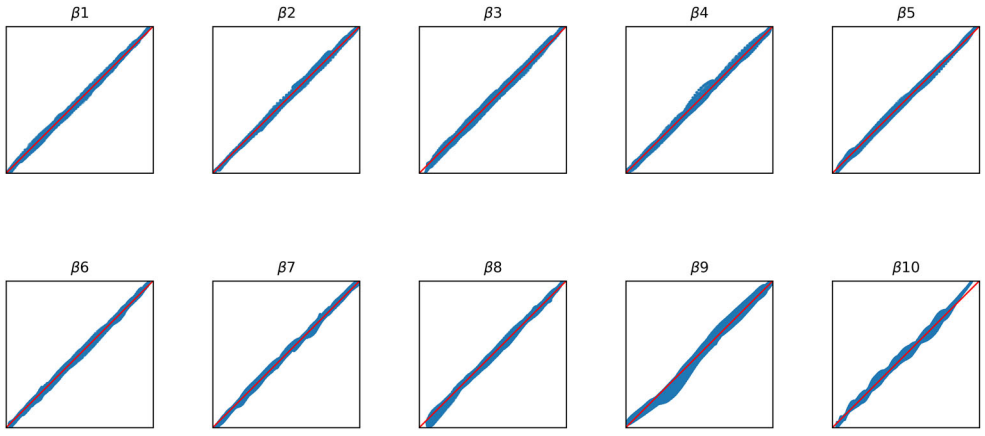

Figure 6. The estimated coefficients $\hat{\beta}_{ij}$ for each covariate (the different plots) and for each area (the dots in each plot). The horizontal axis shows the back-fitting values and the vertical axis shows the gradient-based values. The diagonal red line is where the back-fitting and gradient-based values are equal.

provides information about the individual estimates $\hat{\beta}_{ij}$ obtained by the two algorithms using a data set with 30,000 spatial units. Each plot represents one covariate, and each dot is a spatial unit. The horizontal axis shows the values of the back-fitting algorithm, while the vertical axis shows the values of the gradient-based algorithm. We have the $y = x$ line in red in each plot and we used the same scale in the plots and on both axes to ease their comparison.

The two estimates are very similar for all covariates. The coefficients β_{i10} and β_{i9} associated with the last two covariates have more discrepancy between the two estimates. These coefficients have the smallest bandwidth (see Figure 2) and hence the largest standard deviation. We still do not have a good explanation for these discrepancies, and we cannot rule out the possibility of the presence of some subtle and underlying spatial pattern.

Figure 7 shows the map of the true and estimated coefficients using a data set of 30,000 samples. The estimated coefficients by the two algorithms match among themselves as well as with their true value. In conclusion, we can say that, even for highly volatile surface coefficients, the algorithms have small global errors and estimate approximately the same value in every single area for all covariates.

Real data analysis

We present two applications of our algorithm using real-world data. In each case, we run the gradient-based and back-fitting algorithms. Since these datasets are based on real-world information, we do not possess true values for the coefficients to evaluate the performance of the algorithms. However, we can examine the typical quality-of-fit summary statistics for each algorithm and compare the results.

The first example involves the Georgia dataset, which is included in the MGWR2.2 software manual. The dataset consists of 156 Georgia counties, with the response variable being the percentage of residents holding at least a bachelor's degree. Following

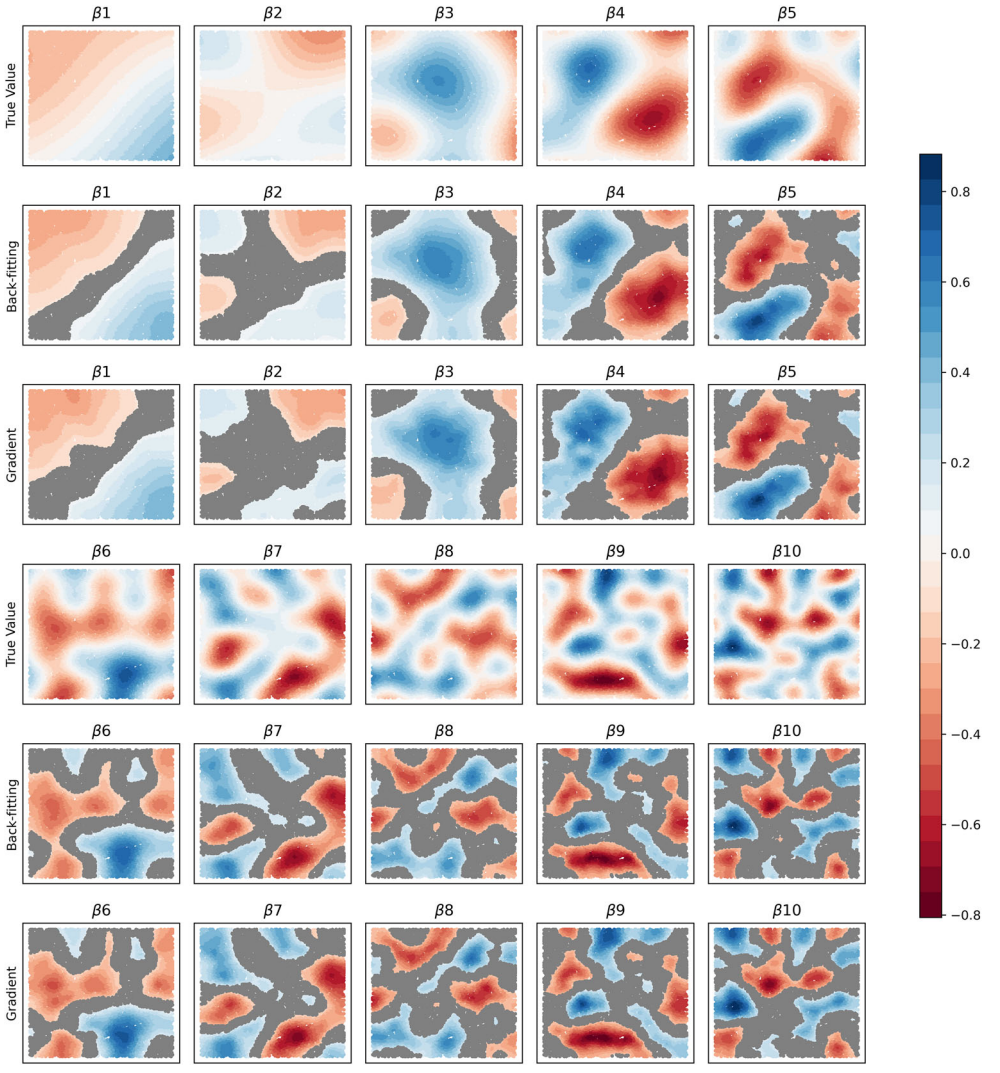

Figure 7. The true coefficient values and the estimated values $\hat{\beta}_{ij}$ from both the back-fitting and gradient-based methods mapped spatially. Non-significant coefficients are marked in grey.

the example in the software manual, we selected four covariates along with the intercept:

- PctBlack: the percentage of African-Americans
- PctFB: the percentage of foreign-born inhabitants
- TotPop90: total population in 1990
- PctEld: the percentage of elderly

When the covariates are entered in the order listed, applying the MGWR2.2 with adaptive bisquare kernel and Golden Section algorithm yields an $AICc = 289.432$, $R^2 = 0.715$, and adjusted $R^2 = 0.682$. Our gradient method produced slightly better results

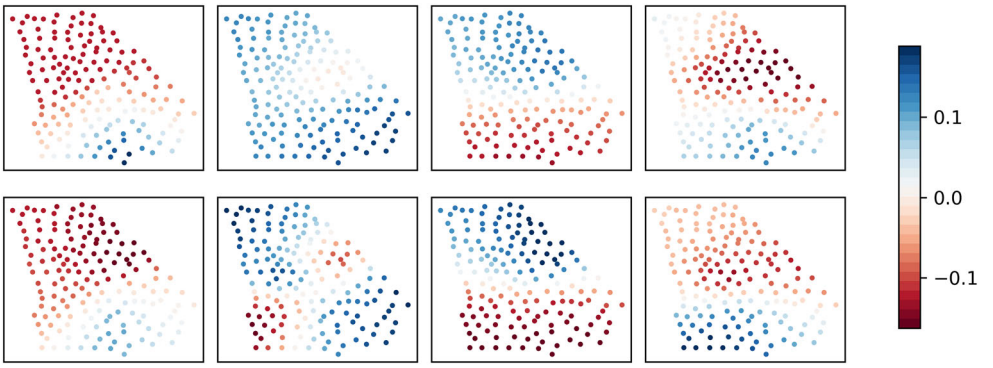

Figure 8. The estimated coefficients $\hat{\beta}_{ij}$ for each covariate in the Georgia dataset. The top row shows the back-fitting results while the bottom row has the gradient-based results. The covariates' order in each row is PctBlack, PctFB, TotPop90, and PctEld.

with an $AICc = 287.840$, $R^2 = 0.712$, and adjusted $R^2 = 0.683$. However, in practical terms, these differences can be considered negligible.

Figure 8 shows the coefficients estimated by each algorithm. It is evident that the results are highly similar, with the exception of the second coefficient (PctFB). For the back-fitting algorithm, the bandwidths for PctBlack, PctFB, TotPop90, and PctEld were 96, 116, 67, and 142, respectively. On the other hand, the gradient algorithm yielded bandwidths of 101, 110, 114, and 122 for the same covariates. Although the TotPop90 covariate displayed the largest difference, Figure 8 indicates that the resulting estimated coefficients were similar. The similarity in *AICc* and R^2 values for both algorithms suggests that the objective function is relatively flat around its minimum, and different approximating solutions lead to similar final estimates.

One challenge in this empirical comparison is that the estimated bandwidths in the back-fitting algorithm are influenced by the order in which we pass the covariate list, due to how the algorithm cycles through the variables. To illustrate this, we re-ran MGWR2.2 with the covariates entered in a different order: TotPop90, PctEld, PctBlack, and PctFB. This resulted in bandwidth estimates of 67, 117, 117, and 116, respectively. These values differ from the estimates obtained using the previous order of covariates. Furthermore, the algorithm converged in 11 iterations compared to the previous 15 iterations. These challenges encountered in empirical comparisons of algorithms highlight why theoretical complexity analysis, as discussed in Section "Theoretical performance of gradient-based MGWR" is the preferred method for comparing performance.

In another application, we employed the well-known Boston Housing dataset originally introduced in Harrison and Rubinfeld (1978). This dataset comprises information collected by the U.S Census Service, focusing on housing in 506 areas of Boston, Massachusetts. The dependent variable of interest is the median value of houses in each location. Our selected covariates are the per capita crime rate (CRIM), the average number of rooms per dwelling (RM), the proportion of owner-occupied units built prior to 1940 (AGE), the weighted distances to five Boston employment centers (DIS), and the nitric oxide concentration (NOX).

In the simulations in Section "Results", our gradient-based algorithm exhibited larger RMSE and MAE values compared to the back-fitting algorithm, as indicated in

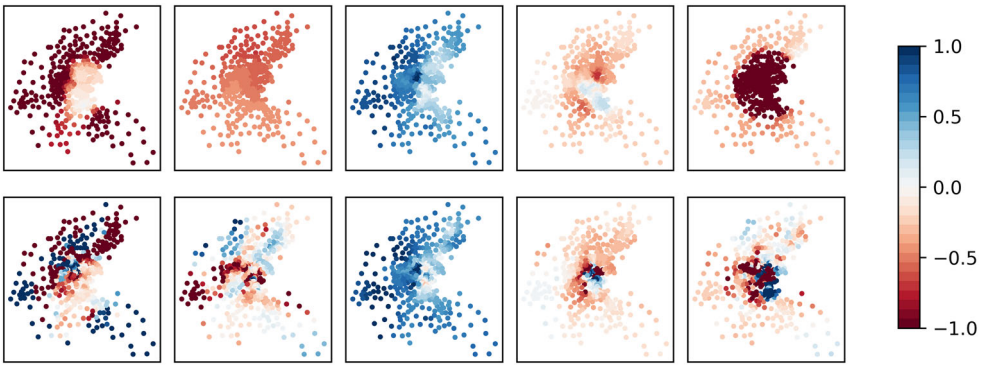

Figure 9. The estimated coefficients $\hat{\beta}_{ij}$ for each covariate in the Boston map. The top row shows the back-fitting results while the bottom row has the gradient-based results. The covariates' order in each row is CRIM, NOX, RM, AGE, DIS.

Table 4. However, in this specific application, we obtained an R^2 value of 0.953, an adjusted R^2 value of 0.926, and an *AICc* of 465.766 for the gradient-based algorithm. On the other hand, the back-fitting algorithm yielded an R^2 value of 0.883, an adjusted R^2 value of 0.857, and an *AICc* of 567.783. Although there is a difference between these metrics, it is relatively small.

Figure 9 displays the coefficients estimated by each model. Both algorithms capture the same spatial pattern, but the back-fitting algorithm generates somewhat smoother surfaces compared to the gradient-based algorithm. It is important to note that both algorithms started with the same initial common bandwidth of $r = 44$, as determined by fitting the GWR model. The final bandwidth estimates for CRIM, RM, AGE, and NOX were $63, 164, 44, 44, 44$ for the back-fitting algorithm, and $35, 35, 34, 34, 34$ for the gradient-based algorithm, respectively.

Discussion

We have explored and demonstrated the possibility of using a gradient-based optimization algorithm for MGWR. We built up the fundamental work for gradient-based MGWR solutions by deriving the gradient vector and Hessian matrix and we measured their computational complexity. Our work demonstrated the power of the gradient-based algorithm to retrieve the true MGWR coefficients and, most importantly, it scales up well with sample size.

We made a comprehensive comparison between gradient-based and back-fitting algorithms. The gradient-based MGWR runs faster and takes less memory than back-fitting for large datasets. In addition, it captures the spatial pattern of covariates, represented by bandwidth, as the back-fitting algorithm does, indicating that they reach the same understanding of the spatial processes involved. Though it has a slightly worse coefficient estimation performance, this does not cancel out the value of the proposed method. Ideally, when computing resources are unlimited, we would choose back-fitting over gradient-based methods for more accurate estimates. However, we may prefer the gradient-based algorithm to analyze large datasets with limited computing power. This is easier to understand if we compare the output from gradient-

based with 40,000 samples and the back-fitting algorithm with 30,000 samples: the former takes 13 h to reach an MAE, RMSE, and MAPE of 0.029, 0.038, 0.362 respectively, while the later takes 28 h to reach extremely close values (0.029, 0.039, 0.362 respectively).

As we showed in Section "Theoretical performance of gradient-based MGWR", computing the Hessian matrix is very expensive for MGWR. We ran some experiments with large datasets, and we saw that its calculation might cancel out the speed enhancement provided by the gradient method. After these preliminary experiments, we do not recommend using the Hessian matrix in the gradient method unless a better implementation is available. Indeed, we expect that a better implementation of our method would be valuable to enhance its performance. We used Python in our implementation coupled with Numba compilation. An implementation entirely in C may speed up the optimization.

The gradient-based MGWR has some advantages. It provides intuitive solutions to the bandwidth selection problem and statistical inference. It also provides computing enhancement. We reduce the runtime cost and memory use in both phases, bandwidth selection, and statistical inference. We showed that the gradient-based algorithm reduces the computational cost from $O(n^3)$ to $O(n^2)$ and reduces memory requirement from $O(n^2)$ to $O(n)$. Another advantage of the gradient-based algorithm is that it applies to all variants of the MGWR model family: GWR and MGWR; fixed and adaptive bandwidths; Gaussian, exponential, and the bi-square kernel, and it can be easily extended to other kernel types. It can be parallelized and it opens up opportunities for big data analysis with MGWR. Furthermore, it allows extending the inference procedure for MGWR applied to GLM models such as logistic or Poisson regression or other objective functions different from the AICc criterion.

Author contributions

The idea of this research was mainly proposed by Mark Janikas and Renato Assunção. The latter worked on the mathematical derivation while Xiaodan Zhou crafted the algorithm and proofread the mathematical derivation. Hu Shao implemented the final code. Cheng-Chia Huang and Xiaodan Zhou designed and ran the simulations and empirical examples. The first version of the manuscript was written by Renato Assunção and Xiaodan Xhou, and it was revised by Hanna Asefaw. All authors commented on the final manuscript and improved it.

Disclosure statement

No potential conflict of interest was reported by the author(s).

Notes on contributors

Xiaodan Zhou is currently a Ph.D. student majoring in Statistics at North Carolina State University. Her expertise lies in spatial statistics, causal inference, and their diverse applications in a variety of fields. Twitter @xiaodanzhou11. E-mail: xzhou34@ncsu.edu

Renato Assunção received his Ph.D. in Statistics from the University of Washington in 1994. He has been an academic until 2021 when he joined ESRI as a researcher. He develops algorithms and probabilistic methods for the statistical analysis of spatial data, especially areal and point

processes data. He has developed Bayesian spatially varying parameter models, a regional partitioning method based on minimum spanning trees, methods to detect arbitrarily shaped clusters, and surveillance methods for the detection of emergent space-time clusters. Twitter: @assuncaoest. E-mail: rassuncao@esri.com

Hu Shao is a software developer in ESRI since 2018. He received his Ph.D. in Geography from Arizona State University in 2018. He focuses on developing spatial statistics algorithms and tools. Email: hshao@esri.com

Cheng-Chia Huang is currently a Sr. Product Engineer in Spatial Statistics Team at ESRI. With GIS and Geography background, she enjoys solving geographical problems using spatial data science techniques. Twitter: @karie_huang. E-Mail: Cheng-Chia_Huang@esri.com

Mark Janikas is a Lead Product Developer focusing on spatial statistics and has been working at esri since earning his Ph.D. in Quantitative Geography from UC Santa Barbara in 2006. Email: mjanikas@esri.com

Hanna Asefaw is a Ph.D. candidate at Scripps Institution of Oceanography at the University of California, San Diego and a Product Engineer at Esri. E-mail: hasefaw@esri.com

Data and codes availability statement

The data and codes that support the findings of this study are available at https://github.com/shaohu/MGWR, and here is its related DOI: 10.5281/zenodo.8200787.

References

Assunção, R.M., 2003. Space varying coefficient models for small area data. *Environmetrics*, 14 (5), 453–473.

Brunsdon, C., Fotheringham, A.S., and Charlton, M.E., 2010. Geographically weighted regression: a method for exploring spatial nonstationarity. *Geographical Analysis*, 28 (4), 281–298.

Brunsdon, C., Fotheringham, S., and Charlton, M., 1998. Geographically weighted regression. *Journal of the Royal Statistical Society*, 47 (3), 431–443.

Byrd, R.H., Hribar, M.E., and Nocedal, J., 1999. An interior point algorithm for large-scale nonlinear programming. *SIAM Journal on Optimization*, 9 (4), 877–900.

Conn, A.R., Gould, N.I., and Toint, P.L., 2000. *Trust region methods*. Philadelphia, PA: SIAM.

Cupido, K., Fotheringham, A.S., and Jevtic, P., 2021. Local modelling of us mortality rates: A multiscale geographically weighted regression approach. *Population, Space and Place*, 27 (1), e2379.

De Ridder, D., *et al.*, 2022. Evolution of the spatial distribution of alcohol consumption following alcohol control policies: a 25-year cross-sectional study in a swiss urban population. *medRxiv*.

Forati, A.M., Ghose, R., and Mantsch, J.R., 2021. Examining opioid overdose deaths across communities defined by racial composition: a multiscale geographically weighted regression approach. *Journal of Urban Health*, 98 (4), 551–562.

Fotheringham, A.S., Brunsdon, C., and Charlton, M., 2003. *Geographically weighted regression: the analysis of spatially varying relationships*. Chichester, UK: John Wiley & Sons.

Fotheringham, A.S., Yang, W., and Kang, W., 2017. Multiscale geographically weighted regression (MGWR). *Annals of the American Association of Geographers*, 107 (6), 1247–1265.

Gelfand, A.E., *et al.*, 2003. Spatial modeling with spatially varying coefficient processes. *Journal of the American Statistical Association*, 98 (462), 387–396.

Gustavson, S., 2005. *Simplex noise demystified*. Linköping, Sweden: Linköping University.

Harrison, D. Jr. and Rubinfeld, D.L., 1978. Hedonic housing prices and the demand for clean air. *Journal of Environmental Economics and Management*, 5 (1), 81–102.

Hastie, T., and Tibshirani, R., 1990. Exploring the nature of covariate effects in the proportional hazards model. *Biometrics*, 46 (4), 1005–1016.
Iyanda, A.E., *et al.*, 2020. A retrospective cross-national examination of covid-19 outbreak in 175 countries: a multiscale geographically weighted regression analysis (January 11-June 28, 2020). *Journal of Infection and Public Health*, 13 (10), 1438–1445.
Li, Z., and Fotheringham, A.S., 2020. Computational improvements to multi-scale geographically weighted regression. *International Journal of Geographical Information Science*, 34 (7), 1378–1397.
Li, Z., *et al.*, 2019. Fast geographically weighted regression (fastgwr): a scalable algorithm to investigate spatial process heterogeneity in millions of observations. *International Journal of Geographical Information Science*, 33 (1), 155–175.
Maiti, A., *et al.*, 2021. Exploring spatiotemporal effects of the driving factors on covid-19 incidences in the contiguous united states. *Sustainable Cities and Society*, 68, 102784.
Mollalo, A., Vahedi, B., and Rivera, K.M., 2020. Gis-based spatial modeling of covid-19 incidence rate in the continental united states. *The Science of the Total Environment*, 728, 138884.
Murakami, D., and Griffith, D.A., 2015. Random effects specifications in eigenvector spatial filtering: a simulation study. *Journal of Geographical Systems*, 17 (4), 311–331.
Murakami, D., *et al.*, 2021. Scalable gwr: a linear-time algorithm for large-scale geographically weighted regression with polynomial kernels. *Annals of the American Association of Geographers*, 111 (2), 459–480.
Niu, L., *et al.*, 2021. Identifying surface urban heat island drivers and their spatial heterogeneity in China's 281 cities: an empirical study based on multiscale geographically weighted regression. *Remote Sensing*, 13 (21), 4428.
Nocedal, J., and Wright, S., 2006. *Numerical optimization*. Berlin: Springer Science & Business Media.
Oshan, T.M., *et al.*, 2019. MGWR: a python implementation of multiscale geographically weighted regression for investigating process spatial heterogeneity and scale. *ISPRS International Journal of Geo-Information*, 8 (6), 269.
Oshan, T.M., Smith, J.P., and Fotheringham, A.S., 2020. Targeting the spatial context of obesity determinants via multiscale geographically weighted regression. *International Journal of Health Geographics*, 19 (1), 11.
Sauer, T., 2018. *Numerical analysis*. 3rd ed. New York, NY: Pearson Education.
Shahneh, M.R., Oymak, S., and Magdy, A., 2021. A-gwr: Fast and accurate geospatial inference via augmented geographically weighted regression, in Proceedings of the 29th International Conference on Advances in Geographic Information Systems, pp. 564–575.
Shen, T., *et al.*, 2020. On hedonic price of second-hand houses in Beijing based on multi-scale geographically weighted regression: scale law of spatial heterogeneity. *Economic Geography*, 40, 75–83.
Tran, D.X., *et al.*, 2022. Quantifying spatial non-stationarity in the relationship between landscape structure and the provision of ecosystem services: An example in the New Zealand hill country. *The Science of the Total Environment*, 808, 152126.
Wu, B., Yan, J., and Lin, H., 2022. A cost-effective algorithm for calibrating multiscale geographically weighted regression models. *International Journal of Geographical Information Science*, 36 (5), 898–917.
Yu, H., *et al.*, 2020. Inference in multiscale geographically weighted regression. *Geographical Analysis*, 52 (1), 87–106.

Appendix

Appendix A. Further technical details

We present in this appendix the detailed derivation of the gradient-based optimization method introduced in Section "Gradient-based optimization for MGWR calibration". Initially, we deal with

the fixed bandwidth case, where r_j is a scalar constant across all the areas. Next, we discuss the case where this bandwidth is defined adaptively, meaning that the distance scale for a single covariate is different across areas. To minimize $AICc(\boldsymbol{r})$ with a derivative-based method, we need to calculate the gradient vector

$$\frac{\partial AICc(\boldsymbol{r})}{\partial \boldsymbol{r}} = \frac{1}{\hat{\sigma}^2(\boldsymbol{r})} \frac{\partial \mathrm{RSS}(\boldsymbol{r})}{\partial \boldsymbol{r}} + \frac{2n(n-1)}{(n-2-\mathrm{tr}(\boldsymbol{S}(\boldsymbol{r})))^2} \frac{\partial \mathrm{tr}(\boldsymbol{S}(\boldsymbol{r}))}{\partial \boldsymbol{r}} \tag{A1}$$

Considering the derivative of $\mathrm{RSS}(\boldsymbol{r})$, we have

$$\frac{\partial RSS(\boldsymbol{r})}{\partial \boldsymbol{r}} = -2\sum_i (y_i - \hat{y}_i) \underbrace{\frac{\partial \hat{y}_i}{\partial \boldsymbol{r}}}_{k\times 1} = -2 \underbrace{\begin{bmatrix} \left(\frac{\partial \hat{\boldsymbol{Y}}}{\partial r_1}\right)' \\ \vdots \\ \left(\frac{\partial \hat{\boldsymbol{Y}}}{\partial r_k}\right)' \end{bmatrix}}_{k\times n} \underbrace{(\boldsymbol{Y} - \hat{\boldsymbol{Y}})}_{n\times 1} \tag{A2}$$

with

$$\underbrace{\frac{\partial \hat{\boldsymbol{Y}}}{\partial r_j}}_{n\times 1} = \frac{\partial}{\partial r_j} \boldsymbol{S}(\boldsymbol{r})\boldsymbol{Y} = \frac{\partial}{\partial r_j} (\boldsymbol{R}_1(r_1) + \boldsymbol{R}_2(r_2) + \ldots + \boldsymbol{R}_k(r_k))\boldsymbol{Y} = \frac{\partial \boldsymbol{R}_j(r_j)}{\partial r_j} \boldsymbol{Y} \tag{A3}$$

because the coordinate r_j appears only in $\boldsymbol{R}_j(r_j)$. Using

$$\frac{\partial \boldsymbol{A}^{-1}}{\partial r} = -\boldsymbol{A}^{-1} \frac{\partial \boldsymbol{A}}{\partial r} \boldsymbol{A}^{-1},$$

we can obtain the derivative of $\boldsymbol{R}_j(r_j)$ with respect to r_j and hence:

$$\frac{\partial \hat{\boldsymbol{Y}}}{\partial r_j} = \begin{bmatrix} x_{1j}\boldsymbol{e}_j(\boldsymbol{X}'\boldsymbol{W}_1(r_j)\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_1(r_j)\boldsymbol{D}_1(r_j)\left[\boldsymbol{I}_n - \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{W}_1(r_j)\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_1(r_j)\right] \\ x_{2j}\boldsymbol{e}_j(\boldsymbol{X}'\boldsymbol{W}_2(r_j)\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_2(r_j)\boldsymbol{D}_2(r_j)\left[\boldsymbol{I}_n - \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{W}_2(r_j)\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_2(r_j)\right] \\ \vdots \\ x_{nj}\boldsymbol{e}_j(\boldsymbol{X}'\boldsymbol{W}_n(r_j)\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_n(r_j)\boldsymbol{D}_n(r_j)\left[\boldsymbol{I}_n - \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{W}_n(r_j)\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_n(r_j)\right] \end{bmatrix} \boldsymbol{Y}$$

where $\boldsymbol{D}_n(r_j)$ is a diagonal matrix that depends on the distance kernel function adopted. The Gaussian kernel leads to $\boldsymbol{D}_i(r) = \mathrm{diag}(d_{i1}^2/r^3, \ldots, d_{in}^2/r^3)$. The exponential kernel implies on $\boldsymbol{D}_i(r) = \mathrm{diag}(d_{i1}/r^2, \ldots, d_{in}/r^2)$ and the bi-square kernel implies on $\boldsymbol{D}_i(r) = (4/r)\mathrm{diag}(a_{i1}, \ldots, a_{in})$ where $a_{ij} = d_{ij}^2/(r^2 - d_{ij}^2)$.

Define $\hat{\boldsymbol{Y}}_i(r_j) = \boldsymbol{X}(\boldsymbol{X}'\boldsymbol{W}_i(r_j)\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_i(r_j)\boldsymbol{Y}$. Note that $\hat{\boldsymbol{Y}}_i(r_j)\ \hat{\boldsymbol{Y}}_i \neq \hat{\boldsymbol{Y}} = \boldsymbol{S}(\boldsymbol{r})\boldsymbol{Y}$. We can rewrite the derivative of $\hat{\boldsymbol{Y}}$ with respect to r_j as

$$\frac{\partial \hat{\boldsymbol{Y}}}{\partial r_j} = \frac{\partial \boldsymbol{R}_j(r_j)}{\partial r_j} \boldsymbol{Y} = \begin{bmatrix} x_{1j}\boldsymbol{e}_j(\boldsymbol{X}'\boldsymbol{W}_1(r_j)\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_1(r_j)\boldsymbol{D}_1(r_j)\left[\boldsymbol{Y} - \hat{\boldsymbol{Y}}_1(r_j)\right] \\ x_{2j}\boldsymbol{e}_j(\boldsymbol{X}'\boldsymbol{W}_2(r_j)\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_2(r_j)\boldsymbol{D}_2(r_j)\left[\boldsymbol{Y} - \hat{\boldsymbol{Y}}_2(r_j)\right] \\ \vdots \\ x_{nj}\boldsymbol{e}_j(\boldsymbol{X}'\boldsymbol{W}_n(r_j)\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_n(r_j)\boldsymbol{D}_n(r_j)\left[\boldsymbol{Y} - \hat{\boldsymbol{Y}}_n(r_j)\right] \end{bmatrix} \tag{A4}$$

We turn our attention now to the derivative of $\mathrm{tr}(\boldsymbol{S}(\boldsymbol{r}))$ in (A1). Consider its j-th coordinate:

$$\frac{\partial \mathrm{tr}(\boldsymbol{S}(\boldsymbol{r}))}{\partial r_j} = \frac{\partial}{\partial r_j} \mathrm{tr}(\boldsymbol{R}_1(r_1) + \boldsymbol{R}_2(r_2) + \ldots + \boldsymbol{R}_k(r_k)) = \frac{\partial}{\partial r_j} \mathrm{tr}(\boldsymbol{R}_j(r_j)).$$

To obtain the trace, we care only for the diagonal elements. We know that $W_i(r_j)[i,i] = K_{r_j}(d_{ii}) = K_{r_j}(0) = 1$ for the Gaussian, exponential, and bi-square kernels. Therefore,

$$\boldsymbol{R}_j(r_j)[i,i] = \boldsymbol{x}_{ij}\boldsymbol{e}_j(\boldsymbol{X}'\boldsymbol{W}_i(r_j)\boldsymbol{X})^{-1}\boldsymbol{x}_i = x_{ij}\underbrace{\left[(\boldsymbol{X}'\boldsymbol{W}_i(r_j)\boldsymbol{X})^{-1}\boldsymbol{x}_i\right][j]}_{j-\text{th coord of a } k\times 1 \text{ vector}}.$$

We can finally obtain the derivative of the trace in (A1):

$$\begin{aligned}\frac{\partial \mathrm{tr}(\boldsymbol{S}(\boldsymbol{r}))}{\partial r_j} &= \frac{\partial}{\partial r_j}\mathrm{tr}\left(\boldsymbol{R}_j(r_j)\right)\\ &= \frac{\partial}{\partial r_j}\sum_i x_{ij}\left[(\boldsymbol{X}'\boldsymbol{W}_i(r_j)\boldsymbol{X})^{-1}\boldsymbol{x}_i\right][j]\\ &= -\sum_{i=1}^{n} x_{ij}\left[(\boldsymbol{X}'\boldsymbol{W}_i(r_j)\boldsymbol{X})^{-1}(\boldsymbol{X}'\boldsymbol{W}_i(r_j)\boldsymbol{D}_i(r_j)\boldsymbol{X})(\boldsymbol{X}'\boldsymbol{W}_i(r_j)\boldsymbol{X})^{-1}\boldsymbol{x}_i\right][j]\end{aligned} \tag{A5}$$

With (A2), (A4), and (A5), we have all the analytical expressions to calculate the gradient vector of $AICc(\boldsymbol{r})$ in (A1).

A1. The Hessian matrix for MGWR

Newton's method requires the evaluation of the $k \times k$ Hessian matrix $\boldsymbol{H}(\boldsymbol{r}) = \partial^2 AICc(\boldsymbol{r})/\partial \boldsymbol{r}^2$ composed by the second-order partial derivatives of $AICc(\boldsymbol{r})$. We have

$$\begin{aligned}\frac{\partial^2 AICc(\boldsymbol{r})}{\partial \boldsymbol{r}^2} &= \frac{\partial}{\partial \boldsymbol{r}}\frac{\partial AICc(\boldsymbol{r})}{\partial \boldsymbol{r}}\\ &= \frac{-1}{n(\hat{\sigma}^2)^2}\underbrace{\frac{\partial \mathrm{RSS}}{\partial \boldsymbol{r}}}_{k\times 1}\underbrace{\frac{\partial \mathrm{RSS}'}{\partial \boldsymbol{r}}}_{1\times k} + \frac{1}{\hat{\sigma}^2}\underbrace{\frac{\partial^2 \mathrm{RSS}}{\partial \boldsymbol{r}^2}}_{k\times k} +\\ &+ \frac{4n(n-1)}{(n-2-\mathrm{tr}(\boldsymbol{S}))^3}\underbrace{\frac{\partial \mathrm{tr}(\boldsymbol{S})}{\partial \boldsymbol{r}}}_{k\times 1}\underbrace{\frac{\partial \mathrm{tr}(\boldsymbol{S})'}{\partial \boldsymbol{r}}}_{1\times k} + \frac{2n(n-1)}{(n-2-\mathrm{tr}(\boldsymbol{S}))^2}\frac{\partial^2 \mathrm{tr}(\boldsymbol{S})}{\partial \boldsymbol{r}^2}\end{aligned} \tag{A6}$$

We obtained all these terms previously except for $\partial^2 \mathrm{RSS}/\partial \boldsymbol{r}^2$ and $\partial^2 \mathrm{tr}(\boldsymbol{S})/\partial \boldsymbol{r}^2$. The latter is a $k \times k$ matrix with (m, j)-th element $\partial^2 \mathrm{tr}(\boldsymbol{S}(\boldsymbol{r}))/(\partial r_m \partial r_j)$ given by

$$-\frac{\partial}{\partial r_m}\sum_i x_{ij}\left[(\boldsymbol{X}'\boldsymbol{W}_i(r_j)\boldsymbol{X})^{-1}(\boldsymbol{X}'\boldsymbol{W}_i(r_j)\boldsymbol{D}_i(r_j)\boldsymbol{X})(\boldsymbol{X}'\boldsymbol{W}_i(r_j)\boldsymbol{X})^{-1}\boldsymbol{x}_i\right][j] = a_{jm}\delta_{jm}$$

where $\delta_{jm} = \mathbb{I}(j = m)$ is the binary delta function and $[\boldsymbol{v}][j]$ is the j-th coordinate of the $\mathbf{v}$ vector. To specify the value of a_{jj}, let $\boldsymbol{A} = (\boldsymbol{X}'\boldsymbol{W}_i(r_j)\boldsymbol{X})^{-1}$ and $\boldsymbol{B} = (\boldsymbol{X}'\boldsymbol{W}_i(r_j)\boldsymbol{D}_i(r_j)\boldsymbol{X}$ Then,

$$\begin{aligned}a_{jj} &= -\sum_{i=1}^{n} x_{ij}\left[\left[-\boldsymbol{ABABA} + \boldsymbol{A}\left[\boldsymbol{X}'\boldsymbol{WD}\left(\boldsymbol{D} - \frac{3}{r}\boldsymbol{I}\right)\boldsymbol{X}\right]\boldsymbol{A} - \boldsymbol{ABABA}\right]\boldsymbol{x}_i\right][j]\\ &= -\sum_{i=1}^{n} x_{ij}\left[\left[-2\boldsymbol{ABABA} + \boldsymbol{AX}'\boldsymbol{WD}^2\boldsymbol{XA} - \frac{3}{r}\boldsymbol{ABA}\right]\boldsymbol{x}_i\right][j]\end{aligned}$$

Therefore, the $k \times k$ matrix $\partial^2 \mathrm{tr}(\boldsymbol{S})/\partial \boldsymbol{r}^2$ is a diagonal matrix.

It remains to calculate $\partial^2 RSS/\partial \boldsymbol{r}^2$:

$$\begin{aligned}\frac{\partial^2 RSS}{\partial r_m \partial r_j} &= \frac{\partial^2 \sum_{i=1}^{n}(y_j^2 + \hat{y}_i^2 - 2y_i\hat{y}_i)}{\partial r_m \partial r_j}\\ &= \underbrace{2\sum_{i=1}^{n}\frac{\partial \hat{y}_i}{\partial r_m}\frac{\partial \hat{y}_i}{\partial r_j}}_{A}\underbrace{-2\sum_{i=1}^{n}(y_i - \hat{y}_i)\frac{\partial^2 \hat{y}_i}{\partial r_m r_j}}_{B}\end{aligned}$$

The term labeled as A is the (m, j)-th element of the $n \times n$ matrix

$$2 \underbrace{\left[\frac{\partial \hat{\boldsymbol{Y}}}{\partial r_1} \cdots \frac{\partial \hat{\boldsymbol{Y}}}{\partial r_k}\right]}_{n \times k} \underbrace{\begin{bmatrix} \left(\frac{\partial \hat{\boldsymbol{Y}}}{\partial r_1}\right)' \\ \vdots \\ \left(\frac{\partial \hat{\boldsymbol{Y}}}{\partial r_k}\right)' \end{bmatrix}}_{k \times n}$$

and the vector $\partial \hat{\boldsymbol{Y}}/\partial r_k$ has been previously calculated in (A4). The term labeled B is non-zero only when $j = m$, and it contributes only to the diagonal element of the matrix $\frac{\partial \mathrm{RSS}}{\partial \boldsymbol{r}} \frac{\partial \mathrm{RSS}'}{\partial \boldsymbol{r}}$. These diagonal elements are given by

$$\begin{aligned}
\frac{\partial^2 \hat{y}_i}{\partial r_j^2} &= \frac{\partial}{\partial r_j} \sum_{i=1}^{n} x_{ij} \boldsymbol{e}_j (\boldsymbol{X}' \boldsymbol{W}_i(r_j) \boldsymbol{X})^{-1} \boldsymbol{X}' \boldsymbol{W}_i(r_j) \boldsymbol{D}_i(r_j) \left[\boldsymbol{Y} - \hat{\boldsymbol{Y}}_i(r_j)\right] \\
&= -\sum_{i=1}^{n} x_{ij} \boldsymbol{e}_j (\boldsymbol{X}' \boldsymbol{W}_i(r_j) \boldsymbol{X})^{-1} (\boldsymbol{X}' \boldsymbol{W}_i(r_j) \boldsymbol{D}_i(r_j) \boldsymbol{X}) (\boldsymbol{X}' \boldsymbol{W}_i(r_j) \boldsymbol{X})^{-1} \boldsymbol{X}' \boldsymbol{W}_i(r_j) \\
&\qquad \boldsymbol{D}_i(r_j) \left[\boldsymbol{Y} - \hat{\boldsymbol{Y}}_i(r_j)\right] \\
&\quad + \sum_{i=1}^{n} x_{ij} \boldsymbol{e}_j (\boldsymbol{X}' \boldsymbol{W}_i(r_j) \boldsymbol{X})^{-1} \boldsymbol{X}' \boldsymbol{W}_i(r_j) \boldsymbol{D}_i(r_j) \left(\boldsymbol{D}_i(r_j) - \frac{3}{r_j} \boldsymbol{I}\right) \left[\boldsymbol{Y} - \hat{\boldsymbol{Y}}_i(r_j)\right] \\
&\quad + \sum_{i=1}^{n} x_{ij} \boldsymbol{e}_j (\boldsymbol{X}' \boldsymbol{W}_i(r_j) \boldsymbol{X})^{-1} \boldsymbol{X}' \boldsymbol{W}_i(r_j) \boldsymbol{D}_i(r_j) \left[\boldsymbol{Y} - \frac{\partial \hat{\boldsymbol{Y}}_i(r_j)}{\partial r_j}\right]
\end{aligned}$$

This concludes what is needed to calculate the Hessian of AICc ($\boldsymbol{r}$).

Appendix B. The gradient for the adaptive bandwidth case

We provide here the details to obtain (12). We have

$$\begin{aligned}
\frac{\partial \hat{y}_i}{\partial r_i} &= \boldsymbol{x}_i' \frac{\partial}{\partial r_i} \left[(\boldsymbol{X}' \boldsymbol{W}_i(r_i(v)) \boldsymbol{X})^{-1} \boldsymbol{X}' \boldsymbol{W}_i(r_i(v)) \boldsymbol{Y}\right] \\
&= \boldsymbol{x}_i' \frac{\partial (\boldsymbol{X}' \boldsymbol{W}_i(r_i(v)) \boldsymbol{X})^{-1}}{\partial r_i} \boldsymbol{X}' \boldsymbol{W}_i(r_i(v)) \boldsymbol{Y} + \boldsymbol{x}_i' (\boldsymbol{X}' \boldsymbol{W}_i(r_i(v)) \boldsymbol{X})^{-1} \boldsymbol{X}' \frac{\partial \boldsymbol{W}_i(r_i(v))}{\partial r_i} \boldsymbol{Y} \\
&= \boldsymbol{x}_i' (\boldsymbol{X}' \boldsymbol{W}_i(r_i(v)) \boldsymbol{X})^{-1} \boldsymbol{X}' \boldsymbol{W}_i(r_i(v)) \boldsymbol{D}_i(r_i(v)) \left(\boldsymbol{Y} - \hat{\boldsymbol{Y}}_i\right) \\
&= \boldsymbol{h}_i \boldsymbol{D}_i(r_i(v)) \left(\boldsymbol{Y} - \hat{\boldsymbol{Y}}_i\right)
\end{aligned} \tag{B1}$$

Appendix C. The hessian for the adaptive bandwidth case

To run the adaptive Newton, we only need to calculate the second derivative of $AICc(v)$:

$$\frac{\partial^2 AICc(v)}{\partial v^2} = \frac{\partial}{\partial v} \frac{\partial AICc(v)}{\partial v} = \underbrace{\frac{\partial}{\partial v} \left\{\frac{1}{\hat{\sigma}^2} \frac{\partial \mathrm{RSS}}{\partial v}\right\}}_{A} + \underbrace{\frac{\partial}{\partial v} \left\{\frac{2n(n-1)}{(n - 2 - \mathrm{tr}(\boldsymbol{S}))^2} \frac{\partial \mathrm{tr}(\boldsymbol{S})}{\partial v}\right\}}_{B} \tag{C1}$$

Repeating the same steps and arguments previously developed for the derivative with respect to r, the term labeled as A in (C1) becomes:

$$A = \frac{-1}{n(\hat{\sigma}^2)^2} \left(\frac{\partial RSS}{\partial v}\right)^2 + \frac{1}{\hat{\sigma}^2} \frac{\partial^2 RSS}{\partial v^2} = \frac{1}{\hat{\sigma}^2} \left[\underbrace{\frac{\partial^2 RSS}{\partial v^2}}_{C} - \frac{1}{n\hat{\sigma}^2} \left(\underbrace{\frac{\partial RSS}{\partial v}}_{\text{in (11)}}\right)^2\right]$$

with

$$
\begin{aligned}
C &= \frac{\partial^2 RSS}{\partial v^2} = \frac{\partial}{\partial v}\left[-2\sum_i (y_i - \hat{y}_i)\frac{\partial \hat{y}_i}{\partial v}\right] \\
&= 2\sum_i \left[\left(\tfrac{\partial \hat{y}_i}{\partial v}\right)^2 - (y_i - \hat{y}_i)\frac{\partial^2 \hat{y}_i}{\partial v^2}\right] \\
&= 2\sum_i \left[\left(p_i'(v)\boldsymbol{h}_i\boldsymbol{D}_i(\boldsymbol{Y} - \hat{\boldsymbol{Y}}_i)\right)^2 - (y_i - \hat{y}_i)\frac{\partial^2 \hat{y}_i}{\partial v^2}\right]
\end{aligned}
$$

To obtain the term $\partial^2 \hat{y}_i/\partial v^2$, note that

$$
\begin{aligned}
\frac{\partial^2 \hat{y}_i}{\partial v^2} &= \frac{\partial}{\partial v}\left[\frac{\partial \hat{y}_i}{\partial r_i}\frac{\partial r_i}{\partial v}\right] \\
&= \left[\frac{\partial}{\partial v}\frac{\partial \hat{y}_i}{\partial r_i}\right]\frac{\partial r_i}{\partial v} + \frac{\partial \hat{y}_i}{\partial r_i}\frac{\partial^2 r_i}{\partial v^2} \\
&= \frac{\partial^2 \hat{y}_i}{\partial r_i^2}\left(\frac{\partial r_i}{\partial v}\right)^2 + \frac{\partial \hat{y}_i}{\partial r_i}\frac{\partial^2 r_i}{\partial v^2} \\
&= \frac{\partial^2 \hat{y}_i}{\partial r_i^2}\left(p_i'(v)\right)^2 + \underbrace{\frac{\partial \hat{y}_i}{\partial r_i}}_{\text{in (12)}} p_i''(v)
\end{aligned}
\tag{C2}
$$

among which we have:

$$
\frac{\partial^2 \hat{y}_i}{\partial r_i^2} = \frac{\partial}{\partial r_i}\left(\frac{\partial \hat{y}_i}{\partial r_i}\right) = \frac{\partial \boldsymbol{h}_i\boldsymbol{D}_i(\boldsymbol{Y} - \hat{\boldsymbol{Y}}_i)}{\partial r_i} = \boldsymbol{x}_i'\boldsymbol{A}_i\boldsymbol{B}_i(-2\boldsymbol{X}\boldsymbol{A}_i\boldsymbol{B}_i + \boldsymbol{E}_i)(\boldsymbol{Y} - \hat{\boldsymbol{Y}}_i).
$$

Putting these expressions all together back in the A term in expression (C1), we have

$$
\begin{aligned}
A &= \frac{\partial}{\partial v}\left\{\frac{1}{\hat{\sigma}^2}\frac{\partial \mathrm{RSS}}{\partial v}\right\} \\
&= \frac{1}{\hat{\sigma}^2}\left[2\sum_i \left[\left(p_i'(v)\boldsymbol{h}_i\boldsymbol{D}_i(\boldsymbol{Y} - \hat{\boldsymbol{Y}}_i)\right)^2 - (y_i - \hat{y}_i)\underbrace{\frac{\partial^2 \hat{y}_i}{\partial v^2}}_{\text{in (23)}}\right] - \frac{1}{n\hat{\sigma}^2}\left(\underbrace{\frac{\partial RSS}{\partial v}}_{\text{in (11)}}\right)^2\right]
\end{aligned}
$$

It remains to find the term B in (C1).

$$
\begin{aligned}
B &= \frac{\partial}{\partial v}\left\{\frac{2n(n-1)}{(n-2-\mathrm{tr}(\boldsymbol{S}))^2}\frac{\partial \mathrm{tr}(\boldsymbol{S})}{\partial v}\right\} \\
&= \frac{4n(n-1)}{(n-2-\mathrm{tr}(\boldsymbol{S}))^3}\left(\underbrace{\frac{\partial \mathrm{tr}(\boldsymbol{S})}{\partial v}}_{\text{in (14)}}\right)^2 + \frac{2n(n-1)}{(n-2-\mathrm{tr}(\boldsymbol{S}))^2}\underbrace{\frac{\partial^2 \mathrm{tr}(\boldsymbol{S})}{\partial v^2}}_{J}
\end{aligned}
\tag{C3}
$$

The last step is to obtain J in (C3):

$$
\begin{aligned}
\frac{\partial^2 \mathrm{tr}(\boldsymbol{S})}{\partial v^2} &= \frac{\partial}{\partial v}\sum_i \frac{\partial \boldsymbol{x}_i'(\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{X})^{-1}\boldsymbol{x}_i}{\partial r_i}\frac{\partial r_i}{\partial v} \\
&= \sum_i \frac{\partial}{\partial v}\left[\frac{\partial \boldsymbol{x}_i'(\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{X})^{-1}\boldsymbol{x}_i}{\partial r_i}\frac{\partial r_i}{\partial v}\right] \\
&= \sum_i \left[\frac{\partial}{\partial v}\frac{\partial \boldsymbol{x}_i'(\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{X})^{-1}\boldsymbol{x}_i}{\partial r_i}\right]\frac{\partial r_i}{\partial v} + \frac{\partial \boldsymbol{x}_i'(\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{X})^{-1}\boldsymbol{x}_i}{\partial r_i}\frac{\partial^2 r_i}{\partial v^2} \\
&= \sum_i \frac{\partial^2 \boldsymbol{x}_i'(\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{X})^{-1}\boldsymbol{x}_i}{\partial r_i^2}\left(\frac{\partial r_i}{\partial v}\right)^2 + \frac{\partial \boldsymbol{x}_i'(\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{X})^{-1}\boldsymbol{x}_i}{\partial r_i}\frac{\partial^2 r_i}{\partial v^2}
\end{aligned}
$$

Plugging in notations $\boldsymbol{A_i} = (\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{X})^{-1}$, and $\boldsymbol{B_i} = \boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{D}_i\boldsymbol{X}$ into (14) we have:

$$\begin{aligned}\frac{\partial \boldsymbol{x}_i'(\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{X})^{-1}\boldsymbol{x}_i}{\partial r_i} &= \boldsymbol{x}_i'(\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{D}_i\boldsymbol{X}(\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{X})^{-1}\boldsymbol{x}_i \\ &= \boldsymbol{x}_i'\boldsymbol{A}_i\boldsymbol{B}_i\boldsymbol{X}\boldsymbol{A}_i\boldsymbol{x}_i\end{aligned} \tag{C4}$$

and

$$\frac{\partial^2 \boldsymbol{x}_i'(\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{X})^{-1}\boldsymbol{x}_i}{\partial r_i^2} \tag{C5}$$

$$= \frac{\partial(\boldsymbol{x}_i'\boldsymbol{A}_i\boldsymbol{B}_i\boldsymbol{X}\boldsymbol{A}_i\boldsymbol{x}_i)}{\partial r_i} \tag{C6}$$

$$= -\sum_i \boldsymbol{x}_i'\Big[\boldsymbol{A}_i\boldsymbol{B}_i\boldsymbol{E}_i\boldsymbol{X}\boldsymbol{A}_i - 2[\boldsymbol{A}_i\boldsymbol{B}_i\boldsymbol{X}]^2\boldsymbol{A}_i\Big]\boldsymbol{x}_i \tag{C7}$$

where the diagonal matrix $\boldsymbol{E}_i$ varies with the kernel type:

- Gaussian kernel: $\boldsymbol{E}_i = \boldsymbol{D}_i - \frac{3}{r}\boldsymbol{I}$;
- Bi-square kernel: $\boldsymbol{E}_i = \frac{1}{2}\boldsymbol{D}_i - \frac{3}{r}\boldsymbol{I}$.

Therefore, we can finally produce the complete expression for the J term in (C3):

$$\frac{\partial^2 \mathrm{tr}(\boldsymbol{S})}{\partial v^2} = \sum_i \underbrace{\frac{\partial^2 \boldsymbol{x}_i'(\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{X})^{-1}\boldsymbol{x}_i}{\partial r_i^2}}_{\text{in (C7)}} \left(p_i'(v)\right)^2 + \underbrace{\frac{\partial \boldsymbol{x}_i'(\boldsymbol{X}'\boldsymbol{W}_i\boldsymbol{X})^{-1}\boldsymbol{x}_i}{\partial r_i}}_{\text{in (C4)}} p_i''(v) \tag{C8}$$