

# Scalable Local Regression for Spatial Analytics

Alexei Pozdnoukhov  
National Centre for Geocomputation  
National University of Ireland Maynooth  
Maynooth, Co. Kildare, Ireland  
Alexei.Pozdnoukhov@nuim.ie

Christian Kaiser  
National Centre for Geocomputation  
National University of Ireland Maynooth  
Maynooth, Co. Kildare, Ireland  
Christian.Kaiser@nuim.ie

## ABSTRACT

Local regression models are one of the backbones of spatial analytics. With growing amount of attribute-rich spatial data the issue of computational scalability of such models has to be resolved. A distributed implementation provides a possible solution, however the requirements implied by this choice dictate the need for truly local modelling as communication between components in a distributed implementation is limited or even absent at some stages. The use of such models in a streaming context provide further restrictions. A calibration procedure has to be truly incremental, with constant memory and processing time for any sample in a stream. This paper explores a spatially distributed incremental local regression model satisfying these requirements and providing similar functionality in terms of interpretability and modelling accuracy as the widely-used geographically weighted regression. Our experiments were run on a conventional mid-range 8-core server. In the largest scale experiment we processed a stream of 157 million spatially referenced samples simulating power consumption readings taken every 2 hours in a period of 5 months from about 87000 households in a European country. The software implementation we developed for the evaluation and performance analysis is made available as an open source project.

## Categories and Subject Descriptors

H.2.8 [Database Management]: Database Applications—*data mining, mining methods and algorithms, interactive data exploration and discovery*

## General Terms

ALGORITHMS

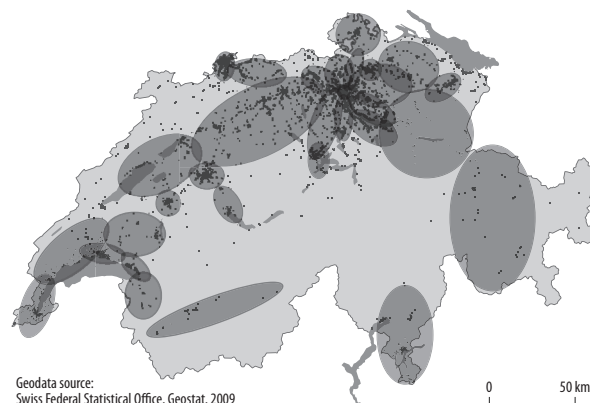
## Keywords

machine learning, spatial statistics, distributed computing, streaming data, mapreduce

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

ACM GIS '11 Chicago, IL, USA

Copyright 2011 ACM 978-1-60558-649-6/09/11 ...\$10.00.



**Figure 1: Distributed local regression: each model is trained incrementally and independently of its peers.**

## 1. INTRODUCTION

The applications when real-time analysis for millions of temporally varying spatially referenced samples is required over wide areas are becoming an everyday necessity. It is hardly likely that an analyst, by looking at data or by imagination can propose an acceptable data model accounting for complex mechanisms behind data generators [5] in order to make full use of the range of tools of spatial statistics. Locally linear regression models [8, 9], on the other hand, are flexible nonparametric tools that are well developed both from statistical and computational aspects. They have been in extensive use for decades, providing interpretable results for modelling non-stationary and nonlinear processes in data-rich situations. Local models have been significantly advanced in machine learning [1]. While being criticised for not performing well in high-dimensional spaces, local learning methods have yet served as a backbone for a wide range of applications. The work presented in this paper is based on a local learning method called receptive field weighted regression [25].

Localisation by spatial distance-decay weighting is a common approach to apply statistical models to deal with non-stationarity and understand heterogeneity in spatially varying relationships [12]. Through application at different spatial scales, such approaches open ways for exploratory analysis and identification of dependencies and thus aid spatial decision makers and researchers in a variety of domains. We are going to preserve these capabilities in our developments.

## 1.1 A motivating example

We now sketch a motivating example challenging the attractive properties of the state-of-the-art spatial statistics. Assume a “smart metering” infrastructure with thousands of sensors providing real-time high resolution readings of water or electricity consumption from nearly every household in a country. Another example is a data stream of domestic product consumption available to a large retail network that keeps track of individual purchases of millions of their customers. Despite evident spatial autocorrelation properties due to the prevailing development type in an area, such as a residential, commercial, or industrial, and availability of socio-economic attributes for individual properties, real-time modelling of these data streams for decision support and planning is computationally challenging. Recently the need for large scale implementation of the spatial statistics models for distributed infrastructures has been acknowledged and first attempts building on the existing software undertaken [14].

## 1.2 Contributions of this paper

The described setting and a general idea of this paper are illustrated in Figure 1. We develop a method that distributed locally linear regression models over an area in a way that independent parameter updates, including automatic bandwidth selection, are possible and computationally efficient when processing streaming data. We also show that this approach retains most of the attractive properties of a conventional geographically weighted regression such as interpretable regression parameter maps, and provide an open source implementation of the method.

There are several aspects in which this work advances the state-of-the-art.

- A locally weighted regression approach is extended for the case of streaming data.
- Temporal and spatial locality adaptation mechanisms are investigated.
- Distributed implementation combining several independently localised regression models is developed with a map-reduce framework, and an experimental validation is carried out.
- An open source implementation of the software is delivered for shared memory multicore servers. This flexibility makes it relatively simple for GIS researchers to benefit from computational resources already available within an academic department.

We conclude an introduction with a brief reminder on the basics of geographically weighted regression. The rest of the paper guides through a number of steps. Section 2 introduces the incremental parameter estimation strategies necessary for adopting data streams. Section 3 describes a strategy for distributing and managing an ensemble of localised regression models over a region. Adaptive approaches to define the shapes of localities and temporal scale for individual models are described in Section 4. Implementation details in a general framework of map-reduce paradigm are presented in Section 6, followed by a comprehensive experimental validation (Section 7) and a discussion (Section 8).

In the following, bold capitals refer to matrices, bold lowercase to column vectors, and regular letters to scalars. Particularly, we consider the incoming data stream of vector attributes  $\{\mathbf{x}_1, \mathbf{x}_2, \dots\}$  of scalar observations  $\{y_1, y_2, \dots\}$  at coordinates  $\{\mathbf{u}_1, \mathbf{u}_2, \dots\}$  at time moments  $\{t_1, t_2, \dots\}$ . Referring to a fixed-size subset of data, a design matrix and the response vector are denoted as  $\mathbf{X}$  and  $\mathbf{Y}$ , and a diagonal matrix of weights  $w_i$  is  $\mathbf{W}$ . Other matrices encountered in the paper are either of the dimensionality of the extended attribute space  $m + 1$  or of the dimensionality of geographical space.

## 1.3 Geographically weighted regression

Geographically weighted regression (GWR) is a form of local linear regression model [8, 9] where locality is assessed by closeness of regression point to calibration data samples in geographical space. With a well-developed statistical inference framework available for a variety of data models this technique gained popularity in quantitative geographical data analysis [12]. Based on a linear regression model  $\mathbf{Y} = \mathbf{X}\beta + \varepsilon, \varepsilon \sim N(0, \sigma\mathbf{I})$  for a set of samples located at  $\mathbf{u}_i$ , GWR provides local parameter estimates  $\beta$  for a location  $\mathbf{c}$  as:

$$\beta = (\mathbf{X}^T \mathbf{W}(\mathbf{u}_i) \mathbf{X})^{-1} \mathbf{X}^T \mathbf{W}(\mathbf{u}_i) \mathbf{Y}, \quad (1)$$

where  $\mathbf{W}(\mathbf{u}_i)$  is a diagonal spatial weight matrix with elements  $w_i$  usually computed by a distance-decay kernel depending on the geographical proximity of regression point  $\mathbf{c}$  to every  $i$ -th calibration sample  $(\mathbf{x}, y)$ <sup>1</sup> located at  $\mathbf{u}_i$ . The inverse of the weighted covariance matrix  $(\mathbf{X}^T \mathbf{W}(\mathbf{u}_i) \mathbf{X})^{-1}$  will be denoted as  $\mathbf{P}$  below.

## 2. STREAMING DATA

We start by considering a single local regression model for some particular locality centred at  $\mathbf{c}$  at time moment  $\tau$ . The weights  $w$  are computed relative to this centre. We assume a linear regression model  $y = \mathbf{x}\beta + \varepsilon$  describing the response  $y$  to a set of attributes  $\mathbf{x}$  for the current moment in time and will derive an update procedure for parameter vector  $\beta$  in response to the stream of geo-referenced samples  $\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \dots\}$  observed in this locality at some time instances  $\{t_1, t_2, \dots\}$ . A specific modification to the model to include temporal component is to define a parametric form of the time-decay weighting of the samples.

Algorithmically, there are several general requirements that need to be satisfied to apply the model to streaming data. One has to guarantee constant time for processing every incoming sample, with no possibility to re-visit past samples and under limited memory constraints. This framework is well developed in recursive system identification [18].

### 2.1 Space-time kernels

It is relatively straightforward to compute spatio-temporal weights with additional temporal decay to apply the model to time-varying data. The weight associated to a sample from the stream observed at  $t$  time units ago at location  $\mathbf{u}$  is:

$$w_c(\mathbf{u}, t) = e^{-(\mathbf{u}-\mathbf{c})^T \mathbf{D}(\mathbf{u}-\mathbf{c}) - \gamma(t-\tau)^q}. \quad (2)$$

<sup>1</sup>We extend the attribute vector  $\mathbf{x}$  with a constant,  $[1 \ \mathbf{x}]$ , to account for an offset  $\beta_0$  in the vector of regression coefficients.

Here,  $\mathbf{D}$  defines the size and shape of spatial locality and  $\gamma$  defines temporal range with the decay exponent  $q = \{1, 2\}$ . For the moment we consider them fixed for a given regression model and revisit this issue in Section 4. Specific issues related to using the method in a streaming context are further investigated in Section 5.

## 2.2 Incremental parameter learning

A sequential incremental update of the parameter vector  $\beta$  is well-known as recursive least squares technique [21] that minimises  $\sum_i^n w_i (y_i - \beta^T \mathbf{x}_i)^2$  and is widely used in recursive identification [18]. With every newly incoming sample  $(\mathbf{x}, y)$  with a corresponding weight  $w$ , the update is:

$$\beta_{n+1} = \beta_n + w \mathbf{P}_{n+1} \mathbf{x} (y - \beta_n^T \mathbf{x}) \quad (3)$$

where an updated  $\mathbf{P}_{n+1}$  is computed as:

$$\mathbf{P}_{n+1} = \frac{1}{\eta} \left( \mathbf{P}_n - \frac{\mathbf{P}_n \mathbf{x} \mathbf{x}^T \mathbf{P}_n}{\frac{\eta}{w} + \mathbf{x}^T \mathbf{P}_n \mathbf{x}} \right). \quad (4)$$

Here a forgetting factor  $\eta$  is introduced for the reasons we explain in Section 5. One can set  $\eta = 1$  without loss of generality until then. Under this choice, a single pass of Eqs. (3)-(4) over a fixed size dataset provides *exactly* the same solution as (1).

## 3. DISTRIBUTING LOCAL MODELS

The basic GWR model as presented in Section 1.3 is inherently localised when used for predictions. It takes a significant computational effort to run it on a large dataset. If predictions are required at  $K$  locations with  $m$ -dimensional attributes  $\{\mathbf{x}_1^T, \mathbf{x}_2^T, \dots, \mathbf{x}_K^T\}$ , in a brute force implementation of pair-wise distances computations in Eqs. (1)-(2) for a calibration set of  $N$  samples one needs  $O(mKN + Km^3)$  computations. Consider now that the same accuracy in predictions can be achieved by some  $\mathcal{K} \ll K$  number of independent local models, then it brings this down to  $O(m\mathcal{K}N + \mathcal{K}m^3)$ . We are going to achieve this by distributing  $\mathcal{K}$  localised linear models over a geographical region of the study.

### 3.1 Managing the ensemble

A simple constructive algorithm manages an ensemble of  $\mathcal{K}$  local models. It is derived to maintain a representative set of local models and to enhance the interpretability and exploratory power for spatial analytics. As a new sample  $\{(\mathbf{x}, y), (\mathbf{u}, t)\}$  arrives, a new model centred at  $\mathbf{c} = \mathbf{u}$ ,  $\tau = t$  is created if  $\max_k w_{\mathbf{c}_k, \tau_k}(\mathbf{u}, t) < w_{\text{add}}$ , where  $w_{\text{add}}$  is a user-defined threshold for activating a new local model. A local model is pruned if a new sample at  $\mathbf{u}_j$  activates at least two models at more than  $w_{\text{off}}$ , meaning there is a redundant model in the current ensemble. We will prune the model spanning a larger area, as estimated by  $\det(\mathbf{D})$ .

After initialisation by locating the first model at the first incoming sample of the stream, the algorithm maintains a growing number of locally linear models. To simplify the notation when talking about a single local model centred at  $\mathbf{c}_k$ , we will denote  $w_{\mathbf{c}_k}(\mathbf{u}_i, t_i)$  as  $w_i$  when referring to the  $i^{\text{th}}$  sample in a locality and  $w^k$  when referring to the weight given to a sample by the  $k^{\text{th}}$  model in the ensemble.

The final prediction of the ensemble of local models for some  $\mathbf{x}$  is computed as

$$\hat{y} = \frac{\sum_{k=1}^{\mathcal{K}} w^k \hat{y}_k}{\sum_{k=1}^{\mathcal{K}} w^k}, \quad (5)$$

where  $\hat{y}_k$  is an output of a local model  $\hat{y}_k = \mathbf{x} \beta_k$ .

## 4. LEARNING THE LOCALITY

Spatial data usually exhibit positive spatial autocorrelation and hence has a variety of impacts on inference in spatial modelling. Most of the approaches depend on a definition of spatial weighting while attempting to quantify a sometimes subjective concept of proximity [2]. The choice relies to a certain degree on one's prior beliefs which depend on domain knowledge and experience of a modeller. It is usually specific to the method and is typically based on distance-decay functions such as a kernel function (2) or some form of quantitative neighbourhood relationships between discrete spatial units.

In GWR, a problem of bandwidth selection for computing weights  $w$  is approached from either computational (leave-one-out cross-validation), theoretical (BIC, AIC) or combined (generalised cross-validation) perspectives. It is a central issue in modelling and interpretational analysis of spatial heterogeneity. This is a very relevant issue for the presented approach; however, the finite number of fixed local models and a data-rich streaming situation provide some interesting opportunities for automated bandwidth tuning that we introduce below.

### 4.1 Local leave-one-out

We will derive the locality update procedure for a single local linear model. Assuming a subset of  $p$  samples in a given locality, a weighted leave-one-out cross-validation error can be computed as

$$\mathcal{L} = \frac{1}{\sum_i w_i} \sum_i^p w_i (y_i - \hat{y}_{-i})^2, \quad (6)$$

where the  $\hat{y}_{-i}$  notation denotes that the  $i^{\text{th}}$  sample was removed from parameter estimation. The key result following from the Sherman-Morrison-Woodbury theorem, and known as a weighted PRESS residual error [19] is that  $\mathcal{L}$  can be computed as

$$\mathcal{L} = \frac{1}{\sum_i w_i} \sum_i^p \frac{w_i (y_i - \hat{y}_i)^2}{(1 - w_i \mathbf{x}_i^T \mathbf{P} \mathbf{x}_i)^2}, \quad (7)$$

thus relieving one to compute the leave-one-out estimates  $\hat{y}_{-i}$  through costly brute force procedure. Most significantly is that this functional is a function of weights  $w$ , and hence  $D$ , and can be minimised to obtain a better estimate of locality shapes for each model. This is a non-convex problem with a global minimum of  $\mathcal{L} \rightarrow 0$  as  $w_i \rightarrow \delta(\mathbf{u}_i, \mathbf{c}_k)$ , which translates to producing very small localities as number of data grows, and consequently to a overly large number of local models. This can be avoided by adding a regularisation term to (7) as:

$$\mathcal{L}_{reg} = \mathcal{L} + \lambda \sum_{g,h} D_{gh}^2 + \lambda_t \gamma^2. \quad (8)$$

The two terms introduced here does not allow the localities to shrink either spatially or temporally. The parameters  $\lambda$  and  $\lambda_t$  are regularisation parameters that control smoothness of the final model [27] and reduce the variance associated with overly local function fitting methods. Eq. (8) can be minimised with respect to  $\mathbf{D}$  and  $\gamma$  to obtain optimal space-time locality estimates. For convenience, we consider

an LU decomposition of  $\mathbf{D} = \mathbf{M}^T \mathbf{M}$  and minimise (8) with respect to  $\mathbf{M}$ .

We rely on prior knowledge for an appropriate initial guess for  $\mathbf{M}_0$  and  $\gamma_0$  and apply a gradient descent with learning rates  $\alpha$  and  $\alpha_\gamma$

$$\mathbf{M}_{n+1} = \mathbf{M}_n - \alpha \frac{\partial \mathcal{L}_{reg}}{\partial \mathbf{M}}, \quad \gamma_{n+1} = \gamma_n - \alpha_\gamma \frac{\partial \mathcal{L}_{reg}}{\partial \gamma}. \quad (9)$$

Gradients' calculation yields:

$$\frac{\partial \mathcal{L}_{reg}}{\partial \mathbf{M}} = \frac{\partial \mathcal{L}}{\partial \mathbf{M}} + \lambda \sum_{g,h} \frac{\partial D_{gh}^2}{\partial \mathbf{M}}, \quad (10)$$

and

$$\frac{\partial \mathcal{L}_{reg}}{\partial \gamma} = \frac{\partial \mathcal{L}}{\partial \gamma} + 2\lambda_t \gamma. \quad (11)$$

Let us have a closer look at the first terms on the right of Eqs. (10)-(11). Substituting (7) and applying the chain rule yields

$$\frac{\partial \mathcal{L}}{\partial \mathbf{M}} = \sum_{j=1}^p \frac{\partial \mathcal{L}}{\partial w_j} \frac{\partial w_j}{\partial \mathbf{M}} = \sum_{j=1}^p \sum_{i=1}^p \frac{\partial J_i}{\partial w_j} \frac{\partial w_j}{\partial \mathbf{M}}, \quad (12)$$

$$\frac{\partial \mathcal{L}}{\partial \gamma} = \sum_{j=1}^p \frac{\partial \mathcal{L}}{\partial w_j} \frac{\partial w_j}{\partial \gamma} = \sum_{j=1}^p \sum_{i=1}^p \frac{\partial J_i}{\partial w_j} \frac{\partial w_j}{\partial \gamma}, \quad (13)$$

where

$$J_i = \frac{1}{W} \frac{w_i (y_i - \hat{y}_i)^2}{(1 - w_i \mathbf{x}_i^T \mathbf{P} \mathbf{x}_i)^2}, \quad W = \sum_i w_i. \quad (14)$$

This is a valid strategy for optimising the locality shapes for a fixed set of data.

## 4.2 Incremental updates

In the streaming context, there is no possibility to store all the  $p$  samples to compute double sums in (12)-(13). Hence, one has to derive a sequential update to the gradient when a new sample  $\{(\mathbf{x}, y), (\mathbf{u}, t)\}$  arrives. Dropping the sums and performing a stochastic gradient descent on a new sample is not a valid approximation as it leads to shrinking the locality and neglecting the data in a vicinity<sup>2</sup>. Instead, [25] proposed an approximation that keeps a memory trace of the gradient and the necessary statistics to perform updates respecting the previously encountered data samples.

Rearranging the derivatives in (10)-(11) to express a single summation sweep through the data set, and using the notation introduced in (14) yields:

$$\frac{\partial \mathcal{L}_{reg}}{\partial \mathbf{M}} = \sum_{j=1}^p \left[ \sum_{i=1}^p \frac{\partial J_i}{\partial w_j} \frac{\partial w_j}{\partial \mathbf{M}} + \lambda \frac{w_j}{W} \sum_{g,h} \frac{\partial D_{gh}^2}{\partial \mathbf{M}} \right], \quad (15)$$

$$\frac{\partial \mathcal{L}_{reg}}{\partial \gamma} = \sum_{j=1}^p \left[ \sum_{i=1}^p \frac{\partial J_i}{\partial w_j} \frac{\partial w_j}{\partial \gamma} + 2 \frac{w_j}{W} \lambda_t \gamma \right]. \quad (16)$$

For every  $n^{\text{th}}$  incoming sample  $\{(\mathbf{x}, y), (\mathbf{u}, t)\}$ , we are going to use an expression inside the square brackets in (15) and (16) to sequentially compute an approximation to the gradient. This sequential approach requires keeping the memory

<sup>2</sup>Note that the second terms in Eqs. (10)-(11), despite contributing to the growth of localities correspondingly in space and time and preventing shrinking, are not aware of the data and appeared here due to enforced regularization.

---

### Algorithm 1 The scalable local regression (SLR) algorithm

---

#### Inputs

- Initial values for spatial and temporal localities  $\mathbf{D}, \gamma$ ;
- Regularization parameters  $\lambda, \lambda_t$ ;
- Forgetting and learning rates  $\eta, \alpha, \alpha_t$ ;
- Add, prune and update thresholds  $w_{\text{add}}, w_{\text{off}}, w_{\text{upd}}$ .

- 1: **repeat**
  - 2:   Get next sample  $\{(\mathbf{x}, y), (\mathbf{u}, t)\}$  from the stream;
  - 3:   Compute weights  $w^k$  per Eq. (2);
  - 4:   **for** each local model with  $w^k > w_{\text{upd}}$  **do**
  - 5:     Update  $\beta_k$  per Eqs. (3)-(4);
  - 6:     Update  $\mathbf{D}_k, \gamma_k$  per Eq. (9) using Eqs. (15)-(20) and closing remarks in Section 4.2;
  - 7:   **end for**
  - 8:   **if**  $\forall k, w^k < w_{\text{add}}$  **then**
  - 9:     Create local model centered at  $\mathbf{c} = \mathbf{u}$ .
  - 10:   **end if**
  - 11:   **if**  $\exists k_1, k_2: \min\{w^{k_1}, w^{k_2}\} > w_{\text{off}}$  **then**
  - 12:     Prune model with  $\text{argmax}\{\det(\mathbf{D}_{k_1}), \det(\mathbf{D}_{k_2})\}$ .
  - 13:   **end if**
  - 14: **until** the stream is processed.
- 

traces for the involved values. For example for  $W$  one keeps  $W_{n+1} = \eta W_n + w$ . The term  $\frac{\partial J_i}{\partial w_j}$  is concerned with updating the locality following the incremental modifications of the parameter vector and inverse covariance matrix (3)-(4), and is derived in Eqs. (16)-(17) of [25]. The total storage requirements are of  $O(m^2)$ . The remaining derivatives are computed as:

$$\frac{\partial w}{\partial \gamma} = -w(t - \tau)^q, \quad (17)$$

$$\frac{\partial w}{\partial M_{fg}} = -w(\mathbf{u} - \mathbf{c})^T \frac{\partial \mathbf{D}}{\partial M_{fg}} (\mathbf{u} - \mathbf{c}), \quad (18)$$

$$\frac{\partial D_{gh}^2}{\partial M_{fs}} = 2D_{gh} \frac{\partial D_{gh}}{\partial M_{fs}}, \quad (19)$$

$$\frac{\partial D_{gh}}{\partial M_{fs}} = \delta_{sh} M_{fg} + \delta_{gs} M_{fh}. \quad (20)$$

where  $\delta$  is a conventional Kronecker operator, and the index  $j$  is dropped off  $w$  meaning that this weight corresponds to the sample from the stream being processed.

We finally note that Appendix 2 of [25] describes a more efficient second-order gradient descent for locality adaptation which we use in our implementation in experiments below but not present it here to keep the paper concise.

## 5. TIME-VARYING MODELS

There is one problem with the ensemble management approach of Section 3.1. It's direct use would lead to an unbounded growth of the number of local models which fill the space-time cube spanning the data extent. However, it is often the current moment in time that is of most interest to an analyst. There are several ways to focus the developed model on the present. One practical solution is to introduce a temporal memory horizon threshold  $\tau_h$  such that a local model is only initiated and updated if it is relevant within a predefined characteristic time period in the past (such as 24 hours or 1 month), otherwise the model is discarded. Another solution is to include time as an attribute if temporal

trends are believed to be linear in nature or to extend the model with an auto-regressive temporal component.

## 5.1 A focus on streaming

In this work we approached the problem by focusing on efficient real-time processing in a streaming context. The focus on the present moment in time provides an important advantage. Local models can only be distributed in space and updated incrementally to keep up with the variations most relevant for current time. To achieve this, we assume that a nominal order of samples in the stream is consistent with the temporal order, and use a forgetting factor  $\eta$  in Eqs. (3)-(4) to discount the influence of samples from the past on parameter estimation. As shown in [18] (p.57), the choice of  $\eta_i = 1 - \delta_i$ ,  $\delta_i \ll 1$  implies (3)-(4) to give minimum to a weighted least squares:

$$\sum_{i=1}^n \theta(n, i) (y_i - \beta^T \mathbf{x}_i)^2, \quad \text{with } \theta(n, i) = \eta_i \theta(n, i-1). \quad (21)$$

For a constant forgetting rate  $\eta$  one obtains  $\theta(n, i) = w_i \eta^{n-i}$ , where  $w_i$  is taken a spatial part of the weight (2). The difference one can observe from the implicit temporal weighting in (2) is that the samples are discarded with respect to the nominal order in which they appear in the stream, and the rate of the decay is exponential in the number of samples. In practice, however, a corresponding *temporal* length scale can be estimated with  $\tau_h \sim \frac{1}{1-\eta}$  and an appropriate time-dependent form for  $\eta_i$  can be derived. More details on this technique, as well as a related Bayesian interpretation, is available in [18], Sections 2.6 and 5.6.

## 5.2 Summary of the algorithm

Summarising the above, the processing sequence initiated for every sample in a stream is presented in Algorithm 1. For  $\mathcal{K}$  independent local models, sequential processing of every sample without locality updates takes  $O(\mathcal{K}m^2)$  operations. Introducing locality updates bring it back up to  $O(\mathcal{K}m^3)$ . With growing number of models distributed over large area, every sample only activates a low number  $k_{\text{eff}} \ll \mathcal{K}$  of relevant ones in its own locality. One can set up a threshold on the number of models that can be activated to trade off accuracy for keeping firm control over computational costs. Note also that a cumulative computational load with respect to the number of samples in a stream  $n$  grows linearly as  $O(n)$ .

Individual models in the ensemble only communicate at the initial stage of estimating the weights  $w_k$  which only depend on geographical dispersal of the model's centres  $\mathbf{c}$  and current distance metric  $\mathbf{D}$ . The dictionary of centres and distance metrics is low dimensional and can be stored centrally, while after thresholding and selecting the models to update all the processing is done independently at each node containing a local model. The memory capacity required at a node scales with attribute dimensionality as  $O(m^2)$  and is reasonably low in typical spatial analytics applications.

A final remark to make is that every  $k^{\text{th}}$  local model uses an (optional) pre-processing step to apply normalisation to the incoming attributes  $\mathbf{x}$  and response values  $y$  which can be computed with conventional incremental mean and variance formulae.

We can now pass over to presenting a distributed implementation of the described algorithm which we call the scalable local regression (SLR) for simplicity.

## 6. DISTRIBUTED IMPLEMENTATION

The design of the method was aimed at deriving completely independent locally linear models: the update Steps (5)-(6) in Algorithm 1 can be performed independently on distributed nodes. The only communication between the models is performed at Step (3) via the weights  $w^k$ . It is well suited to be implemented as a client-server or a parallel processing system with a centralised gaiter managing the ensemble of the local models, including requesting the weights  $w^k$ , sending the data samples to the models that need to be updated, creating new or removing obsolete ones. We will consider an implementation of this scheme in a MapReduce framework optimised for distributed models and stream processing. We use the MapReduce paradigm as a basis for implementing the local models due to the flexibility it provides. There is no pre-defined way in which spatial data collection infrastructures are going to be designed, will it be a hierarchical, centralised or fully distributed peer-to-peer system. The implementation we provide<sup>3</sup> can be fitted to either of these with minimal effort.

### 6.1 MapReduce

MapReduce is a programming paradigm developed for robust parallel processing of large data sets on clusters with potentially hundreds or thousands of nodes [11]. MapReduce borrows the concepts of *map* and *reduce* from functional programming. The user-defined *map* function processes key/value pairs to generate intermediate key/value pairs. The *reduce* function processes all intermediate values sharing the same key and generates the final result. The computation task is divided into small chunks and an intermediary result is computed inside the map function; the reduce function combines then all the results from the chunks into the final result. The map functions can easily be distributed over several computers. If these chunks are sufficiently small, they can be easily relaunched in case of a node failure. This makes MapReduce a programming model suitable for building robust distributed computing systems. MapReduce computations can be deployed both on multicore systems such as modern desktop computers with 2 to 12 cores and big distributed clusters with hundreds of nodes [23].

Several statistical and machine learning algorithms such as k-means clustering, locally weighted linear regression, linear support vector machines, back-propagation networks or hidden Markov models have known implementations for the MapReduce paradigm [7]. However, these algorithms operate on static data sets and not on a data stream. Indeed, MapReduce was designed primarily for processing huge static datasets in a distributed storage and processing environment. In its original version, MapReduce is not very well adapted for stream processing. A computation task is typically submitted to the distributed processing system where it runs until completion and terminates. In stream processing, we typically have data flowing into the system at a given externally determined rate, and the system has to keep up with this data stream. The data needs to be processed continuously, batch processing is not a suitable option.

Therefore general purpose distributed stream processing platforms have recently emerged [20]. Particularly, the MapReduce programming paradigm has been adapted for allow-

<sup>3</sup> <http://nccg.nuim.ie/i2maps/>

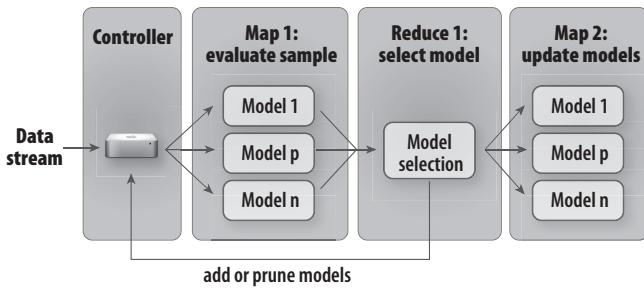


Figure 2: MapReduce for calibrating locally linear regression models in a streaming context.

ing *online aggregation* and *continuous queries* [10]. This can be achieved by establishing a pipeline between the individual mappers and the reducer. Each map output is sent to the reducers as soon as possible, without waiting the completion of all map tasks. Pipelining also allows for continuous MapReduce jobs, where each mapper contains an *agent* recording some statistics of interest. The reducer then aggregates these statistics and delivers current snapshots as results.

## 6.2 Distributing online local spatial regression

Figure 2 illustrates the way how Algorithm 1 is translated into the streaming MapReduce paradigm for model calibration. Each local model runs as a separate process on one of the computational nodes. In a first map step, an activation weight  $w^k$  is computed for each model  $k$  as soon as a new sample  $\{(\mathbf{x}, y), (\mathbf{u}, t)\}$  arrives. Based on the number of activated models, the reducer decides which models have to be updated, and if any model has to be added or removed. This is done through an additional communication channel between the job controller and the reducer.

The different MapReduce operations are done asynchronously. This means that when sample 1 is sent to the  $k$  models for computing the weights  $w^k$ , there is no need to wait for the termination of the whole operation for this particular sample before sending sample 2. The communication between all the processing steps is done using the queues buffering the intermediary results. If a queue is full, the producing process is halted until the consuming process pulls a pending result for processing. The queues also make sure that the processing order is respected.

Figure 3 illustrates the MapReduce implementation for predicting a value given a regression sample  $\{(\mathbf{x}), (\mathbf{u}, t)\}$ . The weights  $w^k$  and predictions  $\hat{y}_k$  of each  $k^{th}$  model are computed in a map step, and these  $k$  intermediary results are combined into the final prediction as specified in Equation 5. As the number of models can vary in time, the controller communicates the number of expected predictions directly to the reducer. This mechanism goes beyond the original MapReduce paradigm, but is a straightforward extension to the streaming context where each mapper and reducer runs in a separate process. The reducer buffers the output of each mapper until all intermediary results have been received. A reducer can emit an intermediary result before receiving all map outputs, leading to a temporary result that will be updated later on.

Model updates can be done at the same time as model predictions. Additionally, if needed, copies of the running

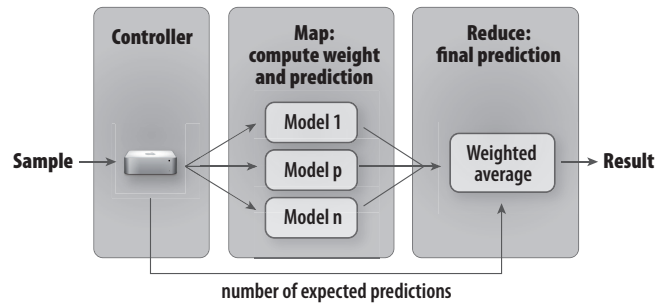


Figure 3: MapReduce for prediction with locally linear regression models in a streaming context.

models can be created on the fly and run in separate processes for prediction purposes.

## 6.3 Fault-tolerance

MapReduce has been designed specifically for large clusters of commodity PCs connected together by a standard networking infrastructure [11]. Machine failures are common in such environment: it is not unusual that a process crashes or hangs, or a hardware component fails. The MapReduce framework is robust in case of different failures mainly through materialisation of the intermediary states to local disks. If a particular map or reduce task fails, it can then simply be recomputed.

In our distributed stream processing scheme, this kind of recovery backup would work with a slight adaptation. A model runs on a node continuously as a separate process. If this node halts, or the process crashes, the state of the model is lost and there is no way to reconstruct it unless a backup model copy is kept. Several instances of the same model can be kept at different nodes and updated in parallel, or multiple copies of each model created after each successful update. All what is needed to achieve this is the possibility for a model to be cloned to another node which only involves  $O(m^2)$  communication costs. It is also possible to have more than one controller process sending the samples to the models, by connecting more than one process to the queues.

## 7. EXPERIMENTS

The volumes of geo-referenced streaming data will grow enormously in the nearest future. An example is the high-resolution electricity consumption data that will become available due to the installation of “smart metering” systems. As these developments are only being at their initial stage at the moment, there is no real data source available to us to investigate all the data modelling aspects in full detail. Instead, the analysis in this section mainly focuses on the computational properties of the presented SLR model.

### 7.1 Data

To face the future scenario, we have simulated the energy consumption data in 87382 households in Switzerland during 5 months at 2 hours interval (1800 time steps), which totals in a data stream of 157 million samples<sup>4</sup>. We have used the

<sup>4</sup>i.e. several orders of magnitude higher than datasets used in GWR modelling until now [14]

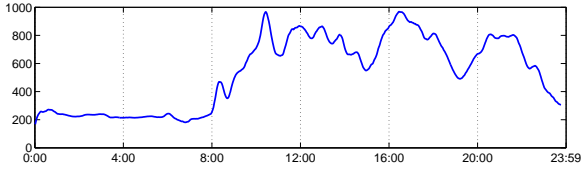


Figure 4: A simulated power consumption profile of a residential building for a typical weekday [24].

Table 1: Testing RMSE

Model	Range	# units	Time, s	Test RMSE
OLS	inf	1	72.3	0.768
GWR	0.5	n/a	$2.5 \times 10^4$	0.678
GWR	0.2	n/a	$2.5 \times 10^4$	0.205
GWR	0.1	n/a	$2.5 \times 10^4$	0.145
GWR	0.05	n/a	$2.5 \times 10^4$	0.061
GWR	0.025	n/a	$2.5 \times 10^4$	0.0345
SLR	0.5	1	75.2	0.772
SLR	0.2	4	121.2	0.239
SLR	0.1	13	192.0	0.174
SLR	0.05	35	291.0	0.063
SLR	0.025	100	577.4	0.035
SLR <sup>adapt</sup>	0.5	1	347.1	0.770
SLR <sup>adapt</sup>	0.25	10	424.2	0.183
SLR <sup>adapt</sup>	0.1	25	621.2	0.107

attribute and location data of real households to simulate the full dataset. Domestic electricity consumption profiles were produced using an advanced micro-simulation generator described in [24]. The presented experimental study therefore investigates computational and exploratory abilities of SLR and can not be used to draw conclusions on the real energy supply system of the region. A sample daily power consumption profile of a large residential building is shown in Figure 4.

The data stream used for regression modelling was generated according to the following model, which is an exact weighted locally linear combination (5):

$$y_{pw} = \frac{1}{\sum_k w_k} \sum_k w_k \sum_j \beta_j^k(\mathbf{u}, t) x^j + \beta_0(\mathbf{u}, t) + \varepsilon, \quad (22)$$

where  $x^j$  are independent variables imitating the living area of the property, the year of construction (which we assume to have a negative impact due to lower energy efficiency) and type of heating used. A number of  $j = 3$  centres at random locations  $\mathbf{u}_k$  was introduced for simulation with regression coefficients  $\beta_j^k$  selected as elements of an Hadamard matrix (taking values of  $\{+1, -1\}$ ) to avoid collinearity issues. A spatial modulation ( $\mathbf{u}_{\text{lon}} + \cos(10\mathbf{u}_{\text{lat}}^2)$ ) was applied to the power consumption profiles. This residual variability is expected to be accounted for by an offset coefficient  $\beta_0(\mathbf{u}, t)$  which varies both in time and space. The last term is the additive homoscedastic normally distributed noise  $\varepsilon \sim N(0, \sigma_n)$ .

## 7.2 Evaluation

A set of experiments presented in this section investigates

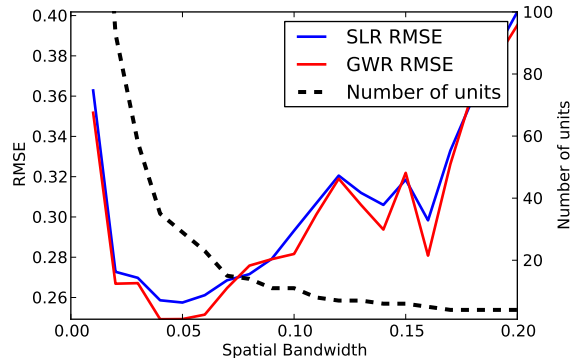


Figure 5: Testing RMSE of GWR and SLR on the same dataset for different spatial bandwidths. The dashed line and a corresponding axis on the right presents the number of local regression units initiated by SLR.

the accuracy of the SLR model, its variation with a number of local models and the issues of parameters’ choice. We provide a comprehensive comparison to a GWR model. The experiments were conducted on 87382 spatial samples for a particular moment in time, using 50% of data for calibration and the other 50% for testing. The results of several models run with various parameters and a noise level of  $\sigma_n = 0.01$  are presented in Table 1.

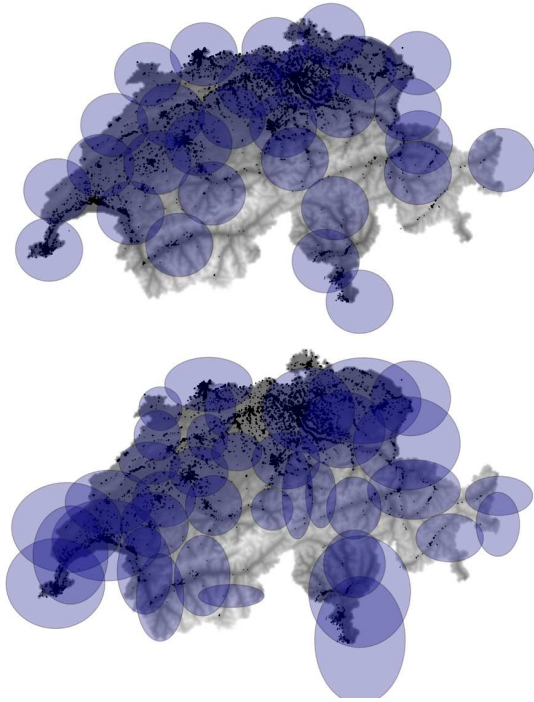
The main observation one can make from Table 1 is that SLR is capable to cope with GWR in terms of accuracy when introducing about 35 localised units, and has comparable performance with much lower computational time otherwise.

Our next experiment concerns model selection. The main issue to be aware of is the bias-variance trade-off. There is a risk of over-fitting by “overly local” models especially if data are noisy. We increased the noise level  $\sigma_n = 0.2$  in the data generator (22) to highlight this issue. Figure 5 presents the testing RMSE for a range of spatial bandwidths  $\sigma$ . An axis on the right indicates the number of local models initiated by SLR which is evidently decreasing with increasing  $\sigma$  and eventually converges to a global ordinary least squares (OLS) regression. One can notice a distinct minima of the testing RMSE at  $\sigma \approx 0.05$  for both GWR and SLR, and a similar overall performance in the whole range of parameters.

## 7.3 Distance learning

This section presents experimental results demonstrating the spatial arrangement of local regression units and the influence of the distance metric adaptation algorithm described in Section 4. Figure 6 (top) illustrates the distribution of 29 centres of local regression units assigned by SLR given fixed bandwidth of  $\sigma = 0.05$ .

The distance learning algorithm is expected to fit the distance metrics to minimise a local leave-one-out error produced by the model, and hence delineate spatial regions within which the modelled dependency is locally linear. As a function of positions of the centres  $\mathbf{c}_i$  and shapes  $\mathbf{D}_i$ , leave-one-out  $\mathcal{L}_{reg}$  is a non-convex functional with many local minima, hence, appropriate fine-tuning of the distance measures depends on appropriate initial values and experience of



**Figure 6:** Top: Local regression units assigned by adaptive distance SLR with initial values  $\sigma = 0.05$ . Bottom: Local regression units with adapted anisotropic distance metric. The locations of data samples (buildings) are shown with black dots.

the expert. Figure 6 (bottom) illustrates the solution found under initial localities of  $\sigma = 0.05$ .

#### 7.4 Interpretability

One of the most important properties of GWR is the possibility to explore the variation of the regression parameters in space. This is a major exploratory instrument to understand and draw conclusions on the influence of space on the observed processes given statistical significance of the observed spatial variation. This possibility is retained in the presented model, as local regression parameters can be mapped in geographical space using an interpolation

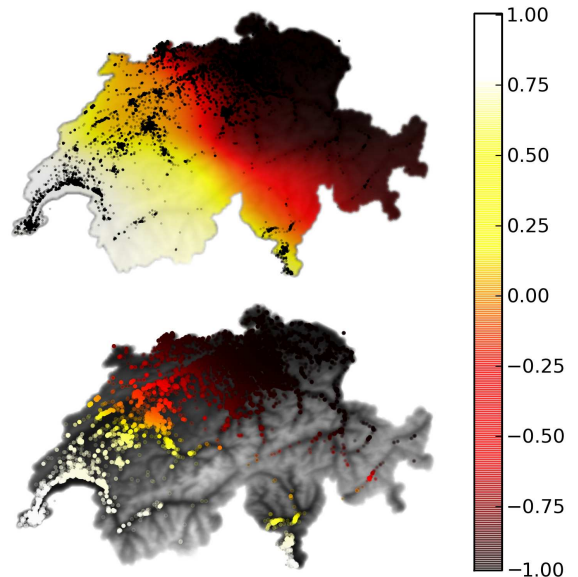
$$\beta^j(\mathbf{u}) = \frac{\sum_{k=1}^{\mathcal{K}} w^k(\mathbf{u}) \beta_k^j}{\sum_{k=1}^{\mathcal{K}} w^k(\mathbf{u})}. \quad (23)$$

Figure 7 present an example of spatial variation of  $\beta_1$  obtained with SLR and a GWR of the same bandwidths. Note that in GWR these values are computed at regression points and no additional interpolation is applied to avoid confusion. Finally, let us note that  $t$ -surfaces can be computed using similar interpolation approach from incrementally updated standard errors of parameters of local models, however we have not explored this possibility in the current paper and focused on performance and scalability tests instead.

#### 7.5 Performance and scalability

There are several important baseline scaling properties we would like to verify experimentally with computational requirements of the SLR model.

First and most important is to verify how computational



**Figure 7:** Spatial variation of  $\beta_1$  regression parameter as obtained with SLR (top) and GWR (bottom) of the same spatial bandwidths.

time grows with a number of local models  $\mathcal{K}$  processed by a single core. Figure 8 presents the results of this experiment. One can observe a linear increase in computation time for both model update and prediction. The implementation used in this test keeps the number of models constant at a predefined value in order to estimate the processing time per data sample. In practice the number of models can vary in time. It depends on the parameters  $w_{add}$  and  $w_{off}$  defined in Section 3.1, the spatial extent of the region and the spatial heterogeneity and non-linearity of the data generation process within this range.

The other property we would like to understand is scaling with respect to the dimensionality of the attribute space which is of  $O(m^2)$  without distance adaptation and  $O(m^3)$  with it according to the theory. The experiment has been run on a single core, the results are summarised in Figure 9. While the computation time for the model update scales super-linearly, the time for querying the models grows linearly with  $m$  and is negligibly small within the tested range of dimensions. Once again, the maximum number of models is kept fixed in this experiment in order to exclude effects related to a varying number of models in time. The update threshold was set to  $w_{upd} = 0$  in order to force the model update for each sample. In practice, the threshold is nonzero and computational costs would generally be significantly lower than those in Figure 9 of our experiment.

#### 7.6 A 157 million samples experiment

Finally, we check the total computational load when increasing the number of samples in a stream. Figure 10 presents a graph where the cumulative computational time is plotted against the number of samples  $n$  as an experimental validation of linear scaling  $O(n)$  and hence a constant time for processing every sample in a stream. The plot shows the scaling up to 1 million samples. This experiment has been run in a realistic production environment using our

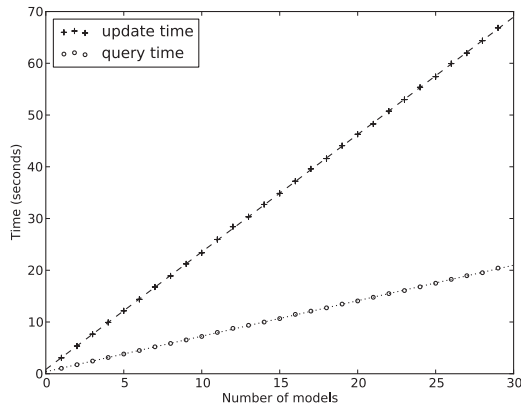


Figure 8: Scaling behaviour with respect to the number of local models.

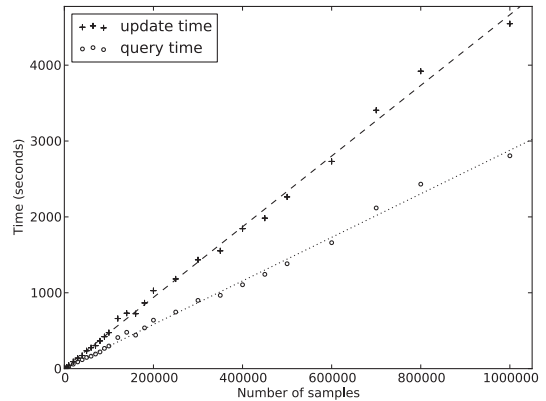


Figure 10: Scaling behaviour with respect to the number of samples.

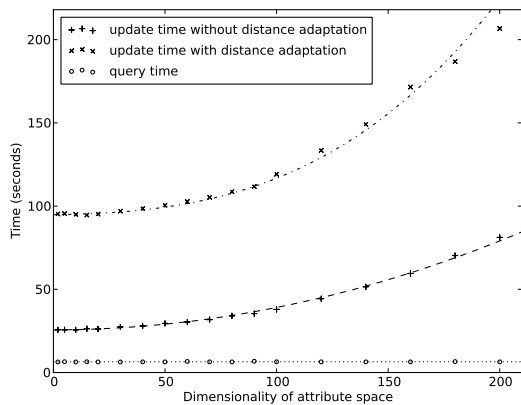


Figure 9: Scaling behaviour with respect to the dimensionality of the attribute space.

distributed implementation on a mid-range 2.13GHz 8-core server. The number of models varies in time which explains the slight deviations of the computation time from a pure linear increase. Due to the constructive nature of the model creation algorithm, the linear scaling behaviour is only true if  $w_{\text{upd}} > w_{\text{add}}$ . Otherwise, one may encounter that a new model must be created and all local models need to be updated as a new sample arrives, leading to  $O(m^2n^2)$  scaling.

In order to check the stability of our implementation and its longer-term scaling behaviour, we have run a 157 million samples experiment using the data outlined in Section 7.1. The model update took roughly 188 hours, while the prediction time for the same amount of samples was about 125 hours which accords well with the scaling behaviour in Figure 10.

## 8. DISCUSSION AND CONCLUSIONS

The presented work has been targeted at a scenario when an analyst is most interested in exploring linear, and thus easily interpretable, dependencies in the observed spatial process over an extended region at the current moment in time. GWR technique can be used to answer such queries, however, its direct application would require: (1) efficient dynamic storage for data samples within a space-time cube of the nearest past; (2) intensive computations of  $O(mn +$

$m^3$ ) for parameter calibration at each query. These drawbacks were resolved with our approach by using incremental parameter estimation technique for an ensemble of independent local models distributed in a region by a constructive algorithm. With this, we drastically improved query latency for real-time analytics without compromising prediction accuracy or model interpretability.

A major advantage for computational efficiency is due to the linearity of the baseline local regression model. For example, a non-linear recursive kernel-based least squares regression implemented in a similar way [16] scales quadratically both in terms of computation and storage requirements with the number of samples overseen by a local model (and not attributes) and is thus significantly slower.

We have used MapReduce, a promising distributed computing framework, to implement the model. Even if algorithms using the MapReduce paradigm are fairly recent, distributed computing has been around for a while already, and an extensive literature on distributed data mining techniques exists [3]. The two directions where advances in recent years were mainly achieved in this context is sensor network data processing [13], where communication between nodes has to be kept minimal and being aware of the overall network topology, and batch processing of huge datasets by various parallel computing architectures. For example, different solutions for distributed multivariate regression [15, 4] are known. In this context, both local linear regression and GWR have been previously implemented in a distributed computation environment [14, 4] for batch processing.

Further extensions to the presented model are possible. An idea to automatically find clusters within which to calibrate local regression models has a long history of developments since it was originated [26]. Given that the streaming nature of the method is kept, a simple constructive algorithm used in our work can possibly be improved using more sophisticated schemes [6] to enhance the usefulness of the model for exploratory spatial analysis.

Concerning the temporal domain it is interesting to further explore the links with recursive identification theory and develop an auto-regressive extension of the SLR. This requires dealing with attribute spaces of high dimensionality. It is tackled in a related model of incremental locally weighted projection regression [17] which is built as an ex-

tension of [25]. In spatial domain, more complex spatial proximity measures can be investigated to enrich models with prior knowledge [22].

Finally, a completely distributed implementation for SLR would need to be developed for the peer-to-peer environments. With this, one would overcome the drawbacks inherent to parallel processing architectures with a centralized controller which acts as a single node of failure both in terms of hardware, data security and user privacy.

## 9. ACKNOWLEDGMENTS

Research presented in this paper was funded by a Strategic Research Cluster grant (07/SRC/I1168) and Stokes Lectureship award by Science Foundation Ireland under the National Development Plan. The authors gratefully acknowledge this support. The authors are grateful to C. Brunson, M. Charlton and A.S. Fotheringham for fruitful discussions on GWR, and to S. Vijayakumar and S. Schaal for releasing and maintaining the MATLAB implementation of RFWR and LWPR models.

## 10. REFERENCES

- [1] C. G. Atkeson, A. W. Moore, and S. Schaal. Locally weighted learning. *Artificial Intelligence Review*, 11(1):11–73, Feb. 1997.
- [2] F. Bavaud. Models for spatial weights: a systematic look. *Geographical Analysis*, 30:153–171, 1998.
- [3] K. Bhaduri, K. Das, K. Liu, and H. Kargupta. Distributed data mining bibliography. <http://www.cs.umbc.edu/~hillol/DDMBIB/>.
- [4] K. Bhaduri and H. Kargupta. An efficient local algorithm for distributed multivariate regression in peer-to-peer networks. In *SIAM conference on data mining, April 24-26*, Atlanta, Georgia, 2008.
- [5] L. Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16(3):199–215, 2001.
- [6] O. Celepcikay and C. Eick. Reg<sup>2</sup>: a regional regression framework for geo-referenced datasets. In *Proceedings of the 17th SIGSPATIAL International Conference on Advances in GIS*, pages 326–335, Seattle, WA, USA, 2009. ACM.
- [7] C.-T. Chu, S. K. Kim, Y.-A. Lin, Y. Y. Yu, G. Bradski, A. Y. Ng, and K. Olukotun. Map-Reduce for machine learning on multicore. *NIPS*, 19, 2007.
- [8] W. S. Cleveland. Robust locally weighted regression and smoothing scatterplots. *Journal of the American Statistical Association*, 74(368):829–836, 1979.
- [9] W. S. Cleveland and S. J. Devlin. Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83(403):596–610, 1988.
- [10] T. Condie, N. Conway, P. Alvaro, J. M. Hellerstein, K. Elmeleegy, and R. Sears. MapReduce online. In *7th USENIX symposium on networked systems design and implementation NSDI'10, April 28-30*, San Jos, CA, 2010.
- [11] J. Dean and S. Ghemawat. MapReduce: simplified data processing on large clusters. In *6th Symposium on Operating Systems Design & Implementation*, San Francisco, CA, 2004. Usenix.
- [12] S. A. Fotheringham, C. Brunson, and M. Charlton. *Geographically Weighted Regression : The Analysis of Spatially Varying Relationships*. John Wiley & Sons, October 2002.
- [13] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden. Distributed regression: an efficient framework for modeling sensor network data. In *Proceedings of the 3rd International Symposium on Information Processing in Sensor Networks (IPSN'04), April 26-27*, Berkeley, CA, 2004.
- [14] R. Harris, A. Singleton, D. Grose, C. Brunson, and P. Longley. Grid-enabling geographically weighted regression: A case study of participation in higher education in England. *Transactions in GIS*, 14(1):43–61, 2010.
- [15] D. E. Hershberger and H. Kargupta. Distributed multivariate regression using wavelet-based collective data mining. *Journal of Parallel and Distributed Computing – Special issue on high-performance data mining*, 61(3), 2001.
- [16] C. Kaiser and A. Pozdnoukhov. Enabling real-time city sensing with kernel stream oracles and mapreduce, June 2011. The First Workshop on Pervasive Urban Applications (PURBA).
- [17] S. Klanke, S. Vijayakumar, and S. Schaal. A library for locally weighted projection regression. *Journal of Machine Learning Research*, 9:623–626, Apr. 2008.
- [18] L. Ljung and T. Soderstrom. *Theory and Practice of Recursive Identification (Signal Processing, Optimization, and Control)*. The MIT Press, 1983.
- [19] R. H. Myers. *Classical and modern regression with applications*, volume Second. Duxbury, 1990.
- [20] L. Neumeier, B. Robbins, A. Nair, and A. Kesari. S4: distributed stream computing platform. In *KDCloud 2010: International Workshop on Knowledge Discovery Using Cloud and Distributed Computing Platforms*, Sydney, Australia, 2010.
- [21] R. L. Plackett. Some theorems in least squares. *Biometrika*, 37(1/2), 1950.
- [22] A. Pozdnoukhov. Spatial extensions to kernel methods. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in GIS*, pages 498–501, New York, NY, USA, 2010. ACM.
- [23] C. Ranger, R. Raghuraman, A. Penmetsa, G. Bradski, and C. Kozyrakis. Evaluating mapreduce for multi-core and multiprocessor systems. In *Proceedings of the 13th Intl. Symposium on High-Performance Computer Architecture (HPCA)*, Phoenix, AZ, February 2007.
- [24] I. Richardson, M. Thomson, D. Infield, and C. Clifford. Domestic electricity use: A high-resolution energy demand model. *Energy and Buildings*, 42(10):1878 – 1887, 2010.
- [25] S. Schaal and C. G. Atkeson. Constructive incremental learning from only local information. *Neural Computation*, 10(8):2047–2084, Nov. 1998.
- [26] H. Spath. Clusterwise linear regression. *Computing*, 4(22):367–373.
- [27] G. Wahba. *Spline models for observational data*, volume 59 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1990.