Cooperative Regression-based Forecasting in Distributed Traffic Networks

Jelena Fiosina and Maksims Fiosins

Institute of Informatics, Clausthal University of Technology, Germany

E-mail: {jelena.fiosina, maksims.fiosins}@gmail.com

Abstract

The paper addresses intelligent data processing and mining in distributed multi-agent networks. We consider a distributed traffic network represented as a multi-agent system, where each agent-vehicle autonomously forecasts its travelling time. We propose a structure for such agents and their corresponding distributed data processing and mining modules. We use distributed linear and kernel-based regression models for travelling time forecasting on the basis of available values of affecting factors. To improve the forecasting quality, the agents exchange information (model parameters or observations) with other traffic participants. We describe forecasting methods, paying special attention to their implementation for streaming data. We propose a cooperative learning algorithm based on the construction of confidence limits for estimates. Simple examples of each method illustrate their application principles. A case study using real-world data from Hanover (Germany) illustrates a practical application of these methods. We propose a structure for linear and kernel-based regression models, which we implement for different multi-agent system architectures (centralised, coordinated/uncoordinated). We analyse the efficiency of linear, kernel-based and aggregated regression estimates in different architectures on the basis of their relative forecasting errors and a goodness-of-fit measure. The results demonstrate that a suitable coordination schema in a distributed architecture provides almost the same accuracy as a centralised architecture.

1. Introduction

The problems in distributed networks, which often appear in complex applications, such as sensor, traffic, or logistics networks, can be solved using multi-agent system (MAS) architectures. At present, research on MAS and distributed networks is focused mainly on decision-making, whereas little attention is paid to the problems of data processing and mining. However, adequate decision-making requires appropriate analysis and processing of input data. In this context, the development of methods for intelligent data analysis and mining of distributed sources is a very timely.

In MAS, the individual and collective behaviours of the agents depend on the observed data from distributed sources. In a typical distributed environment, analysing distributed data is a non-trivial problem because of several constraints such as limited bandwidth (in wireless networks), privacy-sensitive data, distributed computing nodes, etc.

Traditional centralised data processing and mining typically requires central aggregation of distributed data, which may not always be feasible because of the limited network bandwidth, security concerns, scalability problems, and other practical issues. Distributed data processing and mining (DDPM) carries out communication and computation by analysing data in a distributed fashion [1]. The DDPM technology offers more efficient solutions in such applications. DDPM field deals with these challenges by analysing distributed data and offers many algorithmic solutions for data analysis, processing, and mining using different tools in a fundamentally distributed manner that pays careful attention to the resource constraints [2].

In this study we focus on the urban traffic domain, where many traffic characteristics such as travelling time, travelling speed, congestion probability, etc. can be evaluated by autonomous agent vehicles in a distributed manner. Numerous data processing and mining techniques were suggested for forecasting travelling time in a centralised and distributed manner. Statistical methods, such as regression and time series, and artificial intelligence methods, such as neural networks, are successfully implemented for similar problems. However, travelling time is affected by a range of different factors. Thus, its accurate forecasting is difficult and needs considerable amount of traffic data. Understanding the factors affecting travelling time is essential for improving forecasting accuracy [3].

We focus on regression analysis, which is a powerful statistical tool for solving forecasting problems. We compare the well-known multivariate linear regression technique with modern non-parametric computationally intensive kernel-based regression.

We propose two distributed algorithms based on multivariate linear and kernel-based regression for forecasting the travelling time using real-world data from southern Hanover. We assume that each agent autonomously estimates the parameters of its local multivariate linear and kernel-based regression models, paying special attention to the implementation of these models for real-time streaming data. If an agent cannot estimate the travelling time accurately due to a lack of data, i.e., if there is no data near the point of interest (because kernel-based estimation uses approximations), it cooperates with other agents. We suggest two algorithms for multi-agent cooperative learning based on the transmission of the observations (kernel regression) or parameters (linear regression) in response to requests from agents experiencing difficulties. After obtaining the necessary data from other agents, the agent can forecast the travelling time autonomously. We also propose the use of a combination of the estimates to achieve more accurate forecasting. The travelling time estimated during the DDPM stage can be the input for the next stage of distributed decision-making by intelligent agents [4], [5].

The study contributes to the following: 1) the development of the structure of a DDPM module for an autonomous agent, which allows travelling time forecasting and cooperation with other agents; 2) the development of a distributed regression-based (kernel and linear) multi-agent cooperative learning algorithm for real-time streaming data; 3) the development of the structure of a regression model for travelling time forecasting on the basis of available real-world observations; 4) the experimental comparison of different MAS architectures and forecasting methods, including a combination of linear and kernel-based estimates on the basis of real-world data.

The remainder of this chapter is organised as follows. Section 2 describes previous work related to MAS applications for solving problems in distributed networks, the DDPM field in MASs, travelling time forecasting models, and formulates the research question. In section 3, we present a distributed cooperative recursive forecasting algorithm based on multivariate linear regression. This section concludes with a simple tutorial example to illustrate the proposed technique. Section 4 presents the multivariate kernel-based regression model used for streaming data and proposes a cooperative learning algorithm for optimal forecasting in a distributed MAS architecture. This section also concludes with a simple tutorial study. Section 5 presents case studies using data from Hanover (Germany) and it provides a comparison of different MAS architectures and forecasting methods using real-world data. Section 6 presents the discussions of the results and the final section contains the conclusions and perspectives on future work.

2. Data Processing and Mining in Distributed Traffic Networks

In this section, we provide some theoretical background and discuss work related to the problems of distributed data processing and mining when making forecasts about traffic flows.

2.1. Multi-agent System Architectures for Distributed Networks

Complex modern information systems are usually characterised by a large number of relatively autonomous components with a large number of interactions [6]. System designers can usually identify three aspects related to these components: behaviour, environment, and interactions. In most cases, the components act and interact in flexible ways in specific environments to achieve their objectives. The representation of a complex system in this manner means demands the adoption of a multi-agent approach during its engineering [7].

MASs are one of the most popular paradigms for modelling these types of complex systems. MASs contain a set of autonomous intelligent components (agents) with the following important properties [8].

(i) Autonomy. An agent makes its decision itself, without external influences. During decision-making it considers its goals and its knowledge about the current status in the environment.

(ii) Local views. An agent stores information about current state of the environment, although its knowledge may be partial because it lacks global knowledge of in the overall system. Agents may cooperate to exchange available information to produce a more complete picture so they have sufficient information for decision-making.

(iii) Decentralisation. A 'central' agent does not control the overall system. There can be a central authority, but it does not control the overall system and it does not store information in a centralised manner. Thus, agents make their own decisions, and cooperate and coordinate their actions when necessary.

When considering multi-agent representations of complex systems it is important to distinguish between distributed and decentralised systems. In a distributed system, a central authority provides services or performs system management, which involves autonomous components. This central authority can be virtual or distributed among several components. However, system-wide datasets or decisions are modelled by the distributed systems. By contrast, a distributed system does not assume any type of centralised regulation (even virtual). The fully autonomous components of decentralised systems act to achieve their goals and cooperate when necessary.

Distributed systems are usually more expensive because they require additional costs for a central authority. However, decentralised systems experience more problems with work organisation and system goal achievement. In our study, we compared distributed approach with a central authority (called 'centralised') and decentralised coordinated/uncoordinated approach to travelling time forecasting and we reached conclusions about the effect of a central authority and coordination in different situations.

The agents in MAS usually have two important modules: a DDPM module and a distributed decision support (DDS) module [4], [5], [9]. DDPM module is responsible for receiving observations of the environment and discovering patterns or dependencies that may facilitate the decision-making process. DDS module is responsible for making decisions on the basis of the information received from DDPM module and executing appropriate actions in the environment. The general structure of an agent is shown in Figure 1.

The agent acts in the environment and makes observations, which contain data and rewards (measures of the agent efficiency from the environmental perspective). DDPM module pre-processes incoming data using filtering/aggregating data streams and, if necessary, it stores the data for the agent. The necessary information is used to estimate the parameters of data models, which are essential, but not obvious, characteristics of the environment. At this stage, information can be exchanged with other agents (or central authority) to improve the quality of estimation.

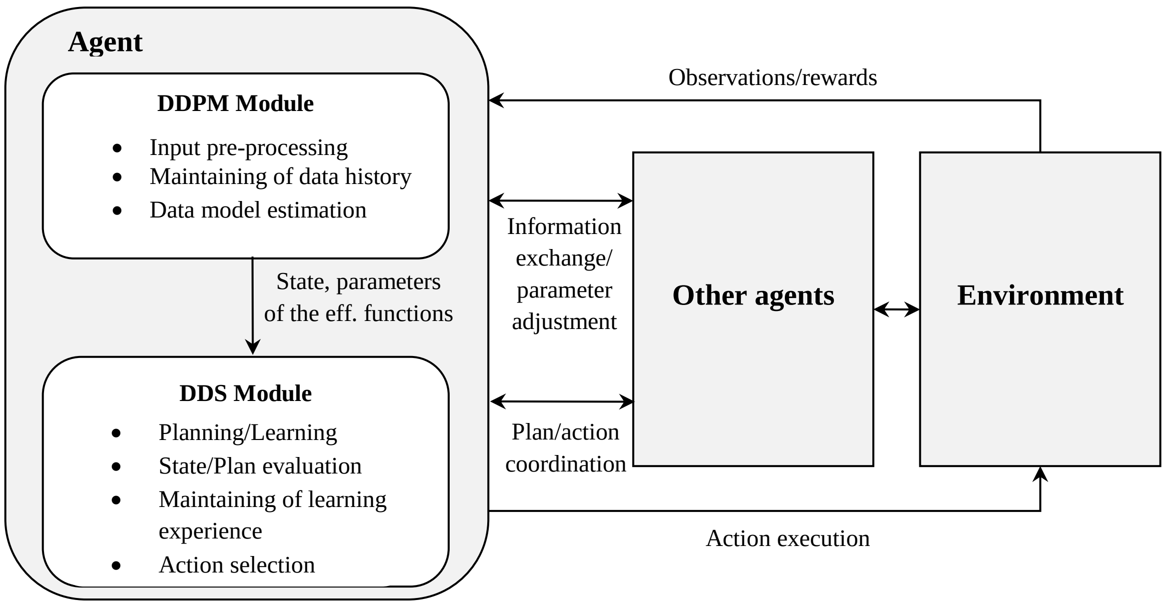

Figure 1. Structure of DDPM and DDS modules in an autonomous agent.

Finally, these parameters are used to form the state of the agent. The decision-making is based on the current state, which contains information about different system features (usually a vector). The state, the rewards, and other essential information are transferred to DDS module.

The DDS module formulates a plan for goal achievement from the current state. Each plan is evaluated with respect to the goals and capabilities of the agent to check whether it allows an agent to achieve its goal in a realistic manner. This is usually achieved based on efficiency functions, which can be fixed or improved ('learned') during the planning process and the agent's operations. The plan or a set of alternative plans can be coordinated with other agents (or central authority). The learning results are also stored in the experience storage. Based on the actual plan, an action is selected and it is executed in the environment. This affects the environment (and other agents) and its results can be observed by the DDPM module.

A road traffic system is a typical complex system, which involves self-interested autonomous participants. Information transfer and action/plan coordination between traffic participants can significantly increase the safety and quality of a traffic system. This is why many researchers have recently applied MAS architectures for modelling different services in traffic networks. In these architectures, the active components (e.g., vehicles, pedestrians, and traffic lights) are modelled as interacting autonomous agents [10], [11], [12].

The following example illustrates an application of multi-agent technology to the design of an intelligent routing support system.

Example: Intelligent routing support

We describe the general structure of an advanced driver support system. This device supports vehicle's driver during the selection of an optimal route in a street network and has the ability to collect actual information and communicate with other vehicles or a central authority if available.

The traffic system is modelled as a MAS. We model the vehicles (more precisely: intelligent devices in the vehicles) as agents, which corresponds to the architecture shown in Figure 1. The architecture of an intelligent routing device is shown in Figure 2.

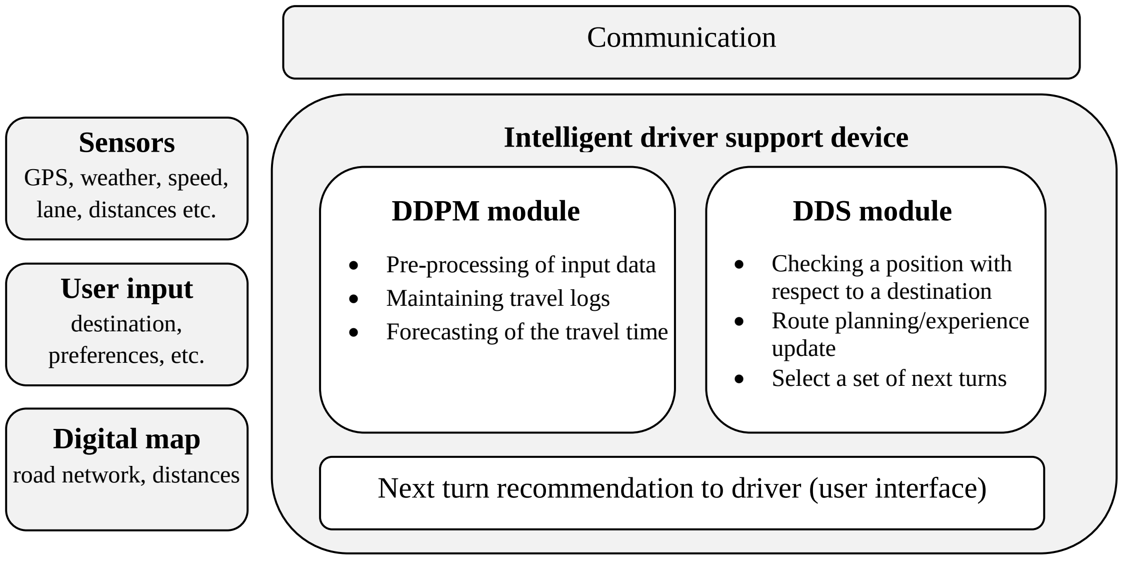
Communication
Sensors
GPS, weather, speed, lane, distances etc.
User input
destination, preferences, etc.
Digital map
road network, distances
Intelligent driver support device
DDPM module
Pre-processing of input data
Maintaining travel logs
Forecasting of the travel time
DDS module
Checking a position with respect to a destination
Route planning/experience update
Select a set of next turns
Next turn recommendation to driver (user interface)

Figure 2. Structure of an Intelligent Driver Support Device that provides route recommendations.

The inputs used in this device (agent) are sensor data that indicate the current situation on a road (e.g., the position, speed, and distances to other vehicles). The user input includes the destination and driving preferences. A digital map contains an underlying road map in the form of a graph.

This data is pre-processed to determine average values for important characteristics, which are stored in the travel log and transferred to the data model for future analysis. The most important task of the DDPM module is travelling time forecasting. An appropriate data model (e.g., regression, neural network, time series, and probabilistic forecasting) is used for this purpose.

The state of the agent includes its current position on the map. The agent produces a set of alternative plans (paths) to travel from the current position to the destination. The travelling times estimated by the DDPM module are used as parameters by the efficiency functions and, together with preferences and learning experience, they are used to evaluate each plan. The plans can be coordinated with other agents during cooperative driving or when driving in convoys. The best possible plan (route) is recommended to the driver.

2.2. Distributed Data Processing and Mining

Recent progress in automated data collection and transmission has created a need for better interpretation and exploitation of vast data sources. Thus, the IT society requires the development of new methods, models, and tools to extract useful information from data. It should also be taken into account that data sources and data processing are distributed, which is problematic for DDPM.

DDPM provides algorithmic solutions for data analysis in a distributed manner to detect hidden patterns in data and extract the knowledge necessary for decentralised decision-making [13], [14]. A promising research area is currently investigating the possibility of coupling MAS with DDPM, by exploiting DDPM methods to improve agent intelligence and MAS performance [2]. This will facilitate highly scalable data architectures, online data processing, hidden data pattern interpretation and analysis, and information extraction from distributed environments. Furthermore, the coupling of MAS with DDPM may be described in terms of ubiquitous intelligence [15], with the aim of fully embedding information processing into everyday life. This concept is very similar to the architecture of data clouds, where data and services are virtualised and provided on demand.

A strong motivation for implementing DDPM for MAS is given by Da Silva et al. [2], where authors argue that DDPM in MAS deals with pro-active and autonomous agents that perceive their environment, dynamically reason out actions on the basis of the environment, and interact with each other. In many complex domains, the knowledge of agents is a result of the outcome of empirical data analysis in addition to the pre-existing domain knowledge. DDPM of agents often involves detecting hidden patterns, constructing predictive and clustering models, identifying outliers, etc. In MAS, this knowledge is usually collective. This collective ’intelligence’ of MAS must be developed by distributed domain knowledge and analysis of the distributed data observed by different agents. Such distributed data analysis may be a non-trivial problem when the underlying task is not completely decomposable and computational resources are constrained by several factors such as limited power supply, poor bandwidth connection, privacy-sensitive multi-party data, etc.

Klusch at al. [16] concludes that autonomous data mining agents, as a special type of information agents, may perform various kinds of mining operations on behalf of their user(s) or in collaboration with other agents. Systems of cooperative information agents for data mining in distributed, heterogeneous, or homogeneous, and massive data environments appear to be quite a natural progression for the current systems to be realised in the near future.

A common characteristic of all approaches is that they aim at integrating the knowledge that is extracted from data at different geographically distributed network nodes with minimum network communication and maximum local computations [2].

Local computation is carried out on each node, and either a central node communicates with each distributed node to compute the global models or a peer-to-peer architecture is used. In the case of the peer-to-peer architecture, individual nodes might communicate with a resource-rich centralised node, but they perform most tasks by communicating with neighbouring nodes through message passing over an asynchronous network [2].

According to da Silva et. al [2], a distributed system should have the following features for the efficient implementation of DDPM:

(i) The system consists of multiple independent data sources, which communicate only through message passing;
(ii) Communication between peers is expensive;
(iii) Peers have resource constraints (e. g. battery power);
(iv) Peers have privacy concerns.

Typically, communication involves bottlenecks. Since communication is assumed to be carried out exclusively by message passing, the primary goal of several DDPM methods, as mentioned in the literature, is to minimise the number of messages sent. Building a monolithic database in order to perform non-distributed data processing and mining may be infeasible or simply impossible in many applications. The costs of transferring large blocks of data may be very expensive and result in very inefficient implementations [1].

Moreover, sensors must process continuous (possibly fast) streams of data. The resource-constrained distributed environments of sensor networks and the need for a collaborative approach to solve many problems in this domain make MAS architecture an ideal candidate for application development.

In our study we deal with homogeneous data. However, a promising approach to agent-based parameter estimation for partially heterogeneous data in sensor networks was suggested in [17]. Another decentralised approach for homogeneous data was suggested in [18] to estimate the parameters of a wireless network by using a parametric linear model and stochastic approximations.

2.3. Travelling Time Forecasting Models

Continuous traffic jams indicate that the maximum capacity of a road network is met or even exceeded. In such a situation, the modelling and forecasting of traffic flow is one of the important techniques that need to be developed [19]. Nowadays, knowledge about travelling time plays an important role in transportation and logistics, and it can be applied in various fields and purposes. From travellers' viewpoints, the knowledge about travelling time helps to reduce the travelling time and improves reliability through better selection of travelling routes. In logistics, accurate travelling time estimation could help to reduce transport delivery costs and to increase the service quality of commercial delivery by delivering goods within the required time window by avoiding congested sections. For traffic managers, travelling time is an important index for traffic system operation efficiency [3].

There are several studies in which a centralised approach is used to predict the travelling time. The approach was used in various intelligent transport systems, such as in-vehicle route guidance and advanced traffic management systems. A good overview is given in [3]. To make the approach effective, agents should cooperate with each other to achieve their common goal via so-called gossiping scenarios. The estimation of the actual travelling time using vehicle-to-vehicle communication without MAS architecture was described in [20].

On the other hand, a multi-agent architecture is better suited for distributed traffic networks, which are complex stochastic systems.

Further, by using centralised approaches the system cannot adapt quickly to situations in real time, and it is very difficult to transmit a large amount of information over the network. In centralised approaches, it is difficult or simply physically impossible to store and process large data sets in one location. In addition, it is known from practice that the most drivers rely mostly on their own experience; they use their historical data to forecast the travelling time [4], [5].

Thus, decentralised MASs are fundamentally important for the representation of these networks [19]. We model our system with autonomous agents to allow vehicles to make decisions autonomously using not only the centrally processed available information, but also their historical data.

Traffic information generally goes through the following three stages: data collection and cleansing, data fusion and integration, and data distribution. The system presented in [21] consists on three components, namely a Smart Traffic Agent, the Real-time Traffic Information Exchange Protocol and a centralised Traffic Information Centre (TIC) that acts as the backend. A similar architecture is used in this study, but the forecasting module, incorporated into Start Traffic Agent (vehicle agent), is different.

In our study we do not focus on the transmission protocol describing only the information, which should be sent from one node to another, without the descriptions of protocol packets. The centralised TIC in our model is used only for storing system information. A combination of centralized and decentralized agent-based approaches to the traffic control was presented in [22]. In this approach, the agents maintain and share the 'local weights' for each link and turn, exchanging this information with a centralized TIC. The decentralised MAS approach for urban traffic network was considered in [23] also, where the authors forecast the traversal time for each link of the network separately. Two types of agents were used for vehicles and links, and a neural network was used as the forecasting model.

Travelling time forecasting for the similar decentralised urban traffic network based on local linear regression model was presented in [24] and based on kernel regression model was presented in [25]. The comparison of parametric and non-parametric approaches for traffic-flow forecasting was made in [26], which shows the efficiency of the non-parametric kernel-based regression approach.

2.4. Problem Formulation

We consider a traffic network with several vehicles, represented as autonomous agents, which predict their travelling time on the basis of their current observations and history. Each agent estimates locally the parameters of the same traffic network. In order to make a forecast, each agent constructs a local regression model, which explains the manner in which different explanatory variables (factors) influence the travelling time (local initial forecasting). A detailed overview of such factors is provided in [3]. The following information is important for predicting the travelling time [27]: average speed before the current segment, number of stops, number of left turns, number of traffic light, average travelling time estimated by TIC. We should also take into account the possibility of an accident, network overload (rush hour) and the weather conditions.

In the case when the vehicle has no or little experience of driving in specific conditions, its local forecasting will be inaccurate. For more accurate travelling time estimation, the vehicle needs adjustment of its models (demand for adjustment). It contacts other traffic participants, which share their experience in the requested conditions. In this step, the focal agent that experiences difficulties with an initial forecast sends the description of the requested conditions to other traffic participants in the transmission radius. Each of neighbour agents tries to make a forecast itself. In the case of a successful forecast, it shares its experience by sending its parameter estimates (parametric linear regression) or observations that are nearest to the requested point (nonparametric kernel-based regression). After receiving the data from the other agents, the focal agent aggregates the obtained results, adjusting the model and increasing its experience, and makes a forecast autonomously (final forecasting). The forecasting procedure of one such vehicle is shown in Figure 3.

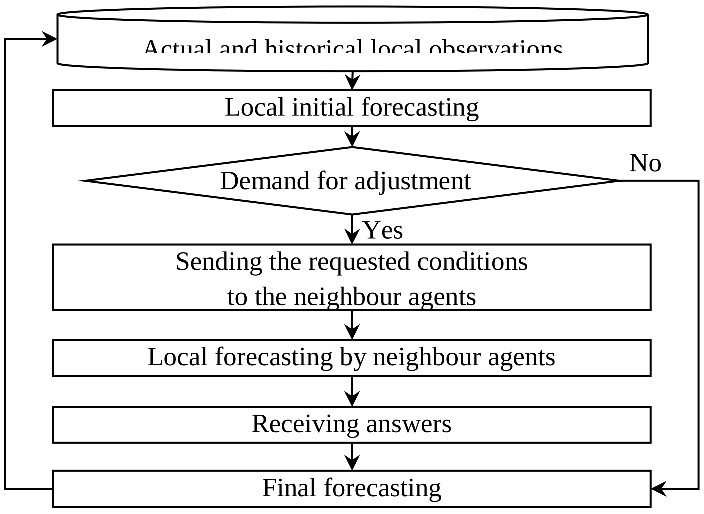
Actual and historical local observations
Local initial forecasting
Demand for adjustment
No
Yes
Sending the requested conditions to the neighbour agents
Local forecasting by neighbour agents
Receiving answers
Final forecasting

Figure 3. Algorithm for local travelling time forecasting by an individual agent.

In short, decentralised travelling time forecasting consists of four steps: 1) local initial forecasting; 2) in the case of unsuccessful initial forecasting: sending the requested conditions to the neighbour agents; 3) local forecasting by neighbour agents and sending answers; 4) aggregation of the answers and final forecasting.

3. Distributed Cooperative Recursive Forecasting Based on Linear Regression

3.1. Forecasting with Multivariate Linear Regression

Regression analysis constitutes several methods widely employed for modelling the dependency between the dependent variable (usually denoted by **Y**) and independent variables (factors, usually denoted by **X**). It describes the dependence of conditional expectation $E[\boldsymbol{Y}|\boldsymbol{X}]$ on the values of several independent variables (factors) **X.**

In order to use a regression model, the following classical assumptions should be satisfied [28]:

(i) the error is a random variable with zero expectation;

(ii) the independent variables are linearly independent (no variable can be expressed as a linear combinations of others);

(iii) the errors are uncorrelated;

(iv) the variance of the errors is constant.

A linear regression model supposes that the dependent variable **Y** is a linear combination of the parameters **X.** The linear regression model in the matrix form can be expressed as

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{3.1}$$

where $\mathbf{Y}$ is an $n \times 1$ vector of dependent variables; $\mathbf{X}$ is an $n \times d$ matrix of explanatory (dependent) variables; $\boldsymbol{\beta}$ is an $d \times 1$ vector of unknown coefficients, which are parameters of the system to be estimated; $\boldsymbol{\varepsilon}$ is an $n \times 1$ vector of random errors. The rows of the matrix $\mathbf{X}$ correspond to observations and the columns correspond to factors.

The regression assumptions can be expressed in this case as follows. We suppose that the components $\{\boldsymbol{\varepsilon}_i\}$ of the error vector $\boldsymbol{\varepsilon}$ are mutually independent, have zero expectation, $E[\boldsymbol{\varepsilon}] = \mathbf{0}$, zero covariances $cov[\boldsymbol{\varepsilon}_i, \boldsymbol{\varepsilon}_j] = 0, i \neq j$, and equal variances, $Var[\boldsymbol{\varepsilon}] = \sigma^2\mathbf{I}$, where $\mathbf{I}$ is an $n \times n$ identity matrix.

The well-known least square estimator (LSE) $\mathbf{b}$ of $\boldsymbol{\beta}$ is:

$$\mathbf{b} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}. \tag{3.2}$$

After the estimation of the parameters $\boldsymbol{\beta}$, we make a forecast for a certain t-th future values of the factors $\mathbf{x}_t$:

$$E[Y_t] = \mathbf{x}_t\mathbf{b}. \tag{3.3}$$

For quality estimation of a linear regression model, we separate the total sum of squares (SST) of the dependent variable $\mathbf{Y}$ into two parts: the sum of squares explained by the regression (SSR) and the sum of squares of errors (SSE):

$$SSR = (\mathbf{Xb} - \bar{Y})^T(\mathbf{Xb} - \bar{Y}),$$
$$SST = (\mathbf{Y} - \bar{Y})^T(\mathbf{Y} - \bar{Y}),$$
$$SSE = SST - SSR.$$

An important measure of model quality is the coefficient of determination R^2, which is defined as a relative part of the sum of squares explained by the regression:

$$R^2 = \frac{SSR}{SST} = \frac{(\mathbf{Xb} - \bar{Y})^T(\mathbf{Xb} - \bar{Y})}{(\mathbf{Y} - \bar{Y})^T(\mathbf{Y} - \bar{Y})} = \frac{\sum_{i=1}^{n}(\mathbf{x}_i\mathbf{b} - \bar{Y})^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}, \tag{3.4}$$

where $\bar{Y} = \sum_{i=1}^{n} Y_i$ when there is an intercept term and $\bar{Y} = 0$ when there is no intercept term in the model.

Precision of the forecast of $\mathbf{Y}_t$ can be measured by a confidence interval. The confidence interval with a confidence level $1 - \alpha$ for the forecast of $\mathbf{Y}_t$ is given by:

$$[\mathbf{x}_t\mathbf{b} - \Delta(\mathbf{x}_t), \qquad \mathbf{x}_t\mathbf{b} + \Delta(\mathbf{x}_t)], \tag{3.5}$$

where $\Delta(\mathbf{x}_t)$ is the width of the confidence interval with a confidence level $1-\alpha$ for a forecast at the point $\mathbf{x}_t$. It can be calculated as

$$\Delta(\mathbf{x}_t) = t_{1-\frac{\alpha}{2},n-d-1}\sqrt{\frac{SSE}{n-d-1}(1+\mathbf{x}_t(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{x}_t^T)}, \tag{3.6}$$

where $t_{1-\frac{\alpha}{2},n-d-1}$ is a $1-\frac{\alpha}{2}$ - level quantile of t-distribution with $n-d-1$ degrees of freedom.

We also use other measures of efficiency of the estimates denoted by the average forecasting error

$$AFE = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \mathbf{x}_i\mathbf{b}| \tag{3.7}$$

and the relative forecasting error for the point $\mathbf{x}_t$

$$RFE_t = \frac{|Y_t - \mathbf{x}_t\mathbf{b}|}{Y_t}. \tag{3.8}$$

3.2. Local Recursive Parameter Estimation

The described estimation procedure, (3.2), requires information about all observations, i.e., the complete matrix **X**. In practice, for real-time streaming data, the estimation is performed iteratively, being updated after each new observation. The recurrent iterative method for the LSE was suggested in [29], [30]. This method assumes the recalculation of system parameters for each new observation.

Let us briefly describe the key aspects of this algorithm. Let $\mathbf{b}_t$ be the estimate after t first observations. After receiving the t+1-th observation, we recalculate the estimates of $\boldsymbol{\beta}$ (Y_{t+1} - value of a dependent variable; and $\mathbf{x}_{t+1}$ - values of explanatory variables):

$$\mathbf{b}_{t+1} = \mathbf{b}_t + \mathbf{K}_{t+1}(Y_{t+1} - \mathbf{x}_{t+1}\mathbf{b}_t), \qquad t = 0,1,\ldots, \tag{3.9}$$

where $\mathbf{K}_{t+1}$ is an $d \times 1$ vector of proportionality, smoothness, or compensation. From (3.9), one can observe, that $\mathbf{b}_{t+1}$ is represented as a sum of the previous estimate $\mathbf{b}_t$ and the correction term $\mathbf{K}_{t+1}(Y_{t+1} - \mathbf{x}_{t+1}\mathbf{b}_t)$. Formula (3.9) is based on exponential smoothness, an adaptive forecasting method [30].

To calculate $\mathbf{K}_{t+1}$, we need values of matrices $\mathbf{A}_t$ and $\mathbf{B}_t$, obtained after the last t-th iteration. $\mathbf{A}_t$ and $\mathbf{B}_t$ are square $d \times d$ matrices. $\mathbf{B}_t$ is equal to $(\mathbf{X}^T\mathbf{X})^{-1}$ if this matrix exists, else it is equal to a pseudo-inverse matrix. Matrix $\mathbf{A}_t$ is a projection matrix, therefore, if $\mathbf{x}_{t+1}$ is the linear combination of the rows of matrix $\mathbf{X}_t$ its projection is equal to zero: $\mathbf{x}_{t+1}\mathbf{A}_t = \mathbf{0}$. Starting the algorithm we set the following initial values $\mathbf{A}_0 = \mathbf{I}, \mathbf{B}_0 = \mathbf{0}, \mathbf{b}_0 = \mathbf{0}$.

If the condition $\mathbf{x}_{t+1}\mathbf{A}_t = \mathbf{0}$ is satisfied, then

$$\mathbf{B}_{t+1} = \mathbf{B}_t - (1 + \mathbf{x}_{t+1}\mathbf{B}_t\boldsymbol{x}_{t+1}^T)^{-\mathbf{1}}\mathbf{B}_t\boldsymbol{x}_{t+1}^T(\mathbf{B}_t\boldsymbol{x}_{t+1}^T)^T,$$
$$\mathbf{A}_{t+1} = \mathbf{A}_t, \mathbf{K}_{t+1} = (1 + \mathbf{x}_{t+1}\mathbf{B}_t\boldsymbol{x}_{t+1}^T)^{-\mathbf{1}}\mathbf{B}_t\boldsymbol{x}_{t+1}^T,$$

otherwise

$$\mathbf{B}_{t+1} = \mathbf{B}_t - (\mathbf{x}_{t+1}\mathbf{A}_t\boldsymbol{x}_{t+1}^T)^{-1}\left((\mathbf{B}_t\boldsymbol{x}_{t+1}^T)(\mathbf{A}_t\boldsymbol{x}_{t+1}^T)^T + (\mathbf{A}_t\boldsymbol{x}_{t+1}^T)(\mathbf{B}_t\boldsymbol{x}_{t+1}^T)^T\right) +$$
$$+(\mathbf{x}_{t+1}\mathbf{A}_t\boldsymbol{x}_{t+1}^T)^{-2}(1 + \mathbf{x}_{t+1}\mathbf{B}_t\boldsymbol{x}_{t+1}^T)(\mathbf{A}_t\boldsymbol{x}_{t+1}^T)(\mathbf{A}_t\boldsymbol{x}_{t+1}^T)^T,$$
$$\mathbf{A}_{t+1} = \mathbf{A}_t - (\mathbf{x}_{t+1}\mathbf{A}_t\boldsymbol{x}_{t+1}^T)^{-1}\mathbf{A}_t\boldsymbol{x}_{t+1}^T(\mathbf{A}_t\boldsymbol{x}_{t+1}^T)^T,$$
$$\mathbf{K}_{t+1} = (\mathbf{x}_{t+1}\mathbf{A}_t\boldsymbol{x}_{t+1}^T)^{-\mathbf{1}}\mathbf{A}_t\boldsymbol{x}_{t+1}^T.$$

3.3. Cooperative Linear Regression-Based Learning Algorithm

Suppose we have an MAS consisting of s autonomous agents $\boldsymbol{Ag} = \{Ag^{(1)}, Ag^{(2)}, \dots, Ag^{(s)}\}$, and each of them contains a local regression model, which is estimated on the basis of its experience. We now introduce a parameter adjustment algorithm, which allows the agents to exchange their model parameters in order to improve the forecasts.

Let us introduce the following notations. We use superscript i for the variables in formula (3.1) to refer to the model of the agent $Ag^{(i)}$:

$$\mathbf{Y}^{(i)} = \mathbf{X}^{(i)}\boldsymbol{\beta} + \boldsymbol{\varepsilon}^{(i)}, \quad i = 1, \dots, s. \tag{3.10}$$

$Ag^{(i)}$ calculates the estimates $\mathbf{b}_t^{(i)}$ of $\boldsymbol{\beta}$ on the basis of its experience using (3.9). It predicts $E\left[Y_{t+1}^{(i)}\right]$ using (3.3) at a future time moment $t + 1$ for the values of factors $\mathbf{x}_{t+1}^{(i)}$.

After forecasting, $Ag^{(i)}$ checks if it needs to adjust its locally estimated parameters $\mathbf{b}_t^{(i)}$ with other agents. This is done when the agent considers a forecast $E\left[Y_{t+1}^{(i)})\right]$ as not reliable. One possible approach is to compare the width of the confidence interval of the forecast with a forecast value. The forecast $E\left[Y_{t+1}^{(i)}\right]$ is considered to be not reliable if its value is sufficiently smaller than its confidence interval width:

$$\frac{\Delta^{(i)}(\mathbf{x}_{t+1}^{(i)})}{\mathbf{x}_{t+1}^{(i)}\mathbf{b}_t^{(i)}} > p,$$

where p is an agent's parameter representing the maximal ratio of the confidence interval width to the forecast, after which a coordination takes place; $\Delta^{(i)}(\mathbf{x})$ is the confidence interval for factors $\mathbf{x}$ based on the agent's data.

Now, let us describe the parameter adjustment procedure. First, $Ag^{(i)}$ sends a request to other agents within its transmission radius; this request contains a set of factors $\mathbf{x}_{t+1}^{(i)}$ as well as a threshold $Tr^{(i)}\left(\mathbf{x}_{t+1}^{(i)}\right)$, which is set equal to the confidence interval width: $Tr^{(i)}\left(\mathbf{x}_{t+1}^{(i)}\right) = \Delta^{(i)}(\mathbf{x}_{t+1}^{(i)})$.

Each agent $Ag^{(j)}$ in the transmission radius calculates the confidence interval $\Delta^{(j)}(\mathbf{x}_{t+1}^{(i)})$ for the requested values of factors $\mathbf{x}_{t+1}^{(i)}$ on the basis of its data and compares it with the threshold. If this value is less than the threshold, $\Delta^{(j)}\left(\mathbf{x}_{t+1}^{(i)}\right) < Tr^{(i)}\left(\mathbf{x}_{t+1}^{(i)}\right)$, $Ag^{(j)}$ replies to $Ag^{(i)}$ by sending its parameters $\mathbf{b}_t^{(j)}$. Let us define $G^{(i)} \subset \boldsymbol{Ag}$, as a group of agents, who are able to reply to $Ag^{(i)}$ by sending their parameters, including $Ag^{(i)}$.

Second, $Ag^{(i)}$ receives replies from the group $G^{(i)}$. It assigns weights to each $Ag^{(j)} \in G^{(i)}$ (including itself) $c^{(i)}=\left\{c_j^{(i)}\right\}$; these weights are time-varying and represent the reliability level of each $Ag^{(j)}$ (including reliability of own experience). In our case, the agents' weights depend on the forecasting experience.

According to the logic of constructing discrete-time consensus, we assume that $c^{(i)}$ is a stochastic vector for all t (the sum of its elements is equal to 1). Then an updated estimate $\tilde{\mathbf{b}}_{\mathrm{t+1}}^{(i)}$ is calculated as

$$\tilde{\mathbf{b}}_{\mathrm{t+1}}^{(i)} = \sum_{Ag^{(j)} \in G^{(i)}} c_j^{(i)} \mathbf{b}_t^{(j)}. \tag{3.11}$$

This adjustment procedure aims to increase the reliability of the estimates, especially for insufficient or missing historical data, and to contribute to the overall estimation accuracy [18].

3.4. Numerical Example

We illustrate the linear regression algorithm by providing a numerical example. First, let us consider a single agent that maintains a dataset of four observations, where variable **Y** depends on three factors **X**. The corresponding matrices **X** and **Y** are:

$$\mathbf{X} = \begin{pmatrix} 5.4 & 3.9 & 2.2 \\ 1.7 & 4.6 & 3.5 \\ 3.2 & 2.3 & 1.2 \\ 4.3 & 2.1 & 3.2 \end{pmatrix}, \qquad \mathbf{Y} = \begin{pmatrix} 2.7 \\ 1.5 \\ 2.6 \\ 3.4 \end{pmatrix}.$$

The agent uses formula (3.9):

$$\mathbf{A}_0 = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}, \mathbf{B}_0 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \mathbf{K}_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}, \mathbf{b}_0 = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix}$$

$$\mathbf{x}_1 = (5.4 \quad 3.9 \quad 2.2), \qquad Y_1 = 2.7, \qquad \mathbf{x}_1\mathbf{A}_0 = (5.4 \quad 3.9 \quad 2.2) \neq \mathbf{0}$$

$$\mathbf{A}_1 = \begin{pmatrix} 0.41 & -0.43 & -0.24 \\ -0.43 & 0.69 & -0.17 \\ -0.24 & -0.17 & 0.90 \end{pmatrix}, \mathbf{B}_1 = \begin{pmatrix} 0.01 & 0.01 & 0.01 \\ 0.01 & 0.01 & 0.00 \\ 0.01 & 0.00 & 0.00 \end{pmatrix}, \mathbf{K}_1 = \begin{pmatrix} 0.11 \\ 0.08 \\ 0.05 \end{pmatrix}, \mathbf{b}_1 = \begin{pmatrix} 0.30 \\ 0.21 \\ 0.12 \end{pmatrix}$$

$$\mathbf{x}_2 = (1.7 \quad 4.6 \quad 3.5), \qquad Y_2 = 1.5, \qquad \mathbf{x}_2\mathbf{A}_1 = (-2.12 \quad 1.84 \quad 1.94) \neq \mathbf{0}$$

$$\mathbf{A}_2 = \begin{pmatrix} 0.02 & -0.09 & 0.11 \\ -0.09 & 0.40 & -0.48 \\ 0.11 & -0.48 & 0.58 \end{pmatrix}, \mathbf{B}_2 = \begin{pmatrix} 0.09 & -0.04 & -0.05 \\ -0.04 & 0.06 & 0.03 \\ -0.04 & 0.03 & 0.03 \end{pmatrix}, \mathbf{K}_2 = \begin{pmatrix} -0.18 \\ 0.16 \\ 0.17 \end{pmatrix}, \mathbf{b}_2 = \begin{pmatrix} 0.37 \\ 0.15 \\ 0.05 \end{pmatrix}$$

$$\mathbf{x}_3 = (3.2 \quad 2.3 \quad 1.2), \qquad Y_3 = 2.6, \qquad \mathbf{x}_3\mathbf{A}_2 = (-0.01 \quad 0.04 \quad -0.06) \neq \mathbf{0}$$

$$\mathbf{A}_3 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \mathbf{B}_3 = \begin{pmatrix} 6.42 & -25.99 & 30.98 \\ -25.99 & 106.15 & -126.80 \\ 30.98 & -126.80 & 151.59 \end{pmatrix}, \mathbf{K}_3 = \begin{pmatrix} -2.05 \\ 8.81 \\ -10.59 \end{pmatrix}, \mathbf{b}_3 = \begin{pmatrix} -1.70 \\ 9.03 \\ -10.61 \end{pmatrix}$$

$$\mathbf{x}_4 = (4.3 \quad 2.1 \quad 3.2), \qquad Y_4 = 3.4, \qquad \mathbf{x}_4\mathbf{A}_3 = (0 \quad 0 \quad 0) = \mathbf{0}$$

$$\mathbf{A}_4 = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \mathbf{B}_4 = \begin{pmatrix} 0.06 & -0.04 & -0.04 \\ -0.04 & 0.20 & -0.19 \\ -0.04 & -0.19 & 0.31 \end{pmatrix}, \mathbf{K}_4 = \begin{pmatrix} 0.09 \\ -0.35 \\ 0.43 \end{pmatrix}, \mathbf{b}_4 = \begin{pmatrix} 0.57 \\ -0.21 \\ 0.43 \end{pmatrix}$$

We can see that the agent obtains the same estimate **b** as they did using formula (3.2):

$$\mathbf{b} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y} = \begin{pmatrix} 0.57 \\ -0.21 \\ 0.43 \end{pmatrix}.$$

Now, we consider the parameter adjustment algorithm. Let us denote the above considered agent as $Ag^{(1)}$. Suppose it should make a forecast for a new point $\mathbf{x}_{t+1}^{(1)}. = (3.7 \quad 2.8 \quad 1.1)$.

$Ag^{(1)}$ makes a forecast $E\left[Y_{t+1}^{(1)}\right]$ for the point $\mathbf{x}_{t+1}^{(1)}$:

$$\mathbf{x}_{t+1}^{(1)}\mathbf{b}_t^{(1)} = (3.7 \quad 2.8 \quad 1.1)\begin{pmatrix} 0.57 \\ -0.21 \\ 0.43 \end{pmatrix} = 1.98.$$

The corresponding confidence interval is:

$$SSR = (\mathbf{Xb})^T(\mathbf{Xb}) = 27.05$$
$$SST = \mathbf{Y}^T\mathbf{Y} = 27.86$$
$$SSE = SST - SSR = 0.81$$
$$\Delta^{(1)}\left(\mathbf{x}_{t+1}^{(1)}\right) = t_{\frac{0.05}{2},5-3-1}\sqrt{\frac{\text{SSE}}{5-3-1}\left(1 + \mathbf{x}_{t+1}^{(1)}(\mathbf{X}^T\mathbf{X})^{-1}\left(\mathbf{x}_{t+1}^{(1)}\right)^T\right)} = 14.38.$$

Let the maximal ratio p=1.5. The agent checks the ratio

$$\frac{\Delta^{(1)}\left(\mathbf{x}_{t+1}^{(1)}\right)}{\mathbf{x}_{t+1}^{(1)}\mathbf{b}_t^{(1)}} = \frac{14.38}{1.98} = 7.28 > 1.5$$

and requests parameter adjustment, because it exceeds p. For this purpose, it sends a point $\mathbf{x}_{t+1}^{(1)} = (3.7 \quad 2.8 \quad 1.1)$ as well as the threshold $Tr^{(1)}\left(\mathbf{x}_{t+1}^{(1)}\right) = \Delta^{(1)}\left(\mathbf{x}_{t+1}^{(1)}\right) = 14\,.38$.

Let there are agents $Ag^{(2)}$ and $Ag^{(3)}$ in the transmission radius with equal experience 4 and the following parameters:

$$\mathbf{b}_t^{(2)} = \begin{pmatrix} 0.58 \\ 0.02 \\ 0.43 \end{pmatrix}, \mathbf{b}_t^{(3)} = \begin{pmatrix} 0.54 \\ -0.004 \\ 0.49 \end{pmatrix}.$$

$Ag^{(2)}$ and $Ag^{(3)}$ calculate their confidence limits $\Delta^{(2)}\left(\mathbf{x}_{t+1}^{(1)}\right) = 7.08$, $\Delta^{(3)}\left(\mathbf{x}_{t+1}^{(1)}\right) = 0.07$. As both are less than the threshold, they send their parameters $\mathbf{b}_t^{(2)}$ and $\mathbf{b}_t^{(3)}$ as well as their experience 4. The group $G^{(i)} = \{1,2,3\}$.

$Ag^{(1)}$ receives the answers and uses weights proportional to the experience of the agents, $c^{(1)} =< \frac{4}{12}, \frac{4}{12}, \frac{4}{12} >=< \frac{1}{3}, \frac{1}{3}, \frac{1}{3} >$. Now an updated estimate $\tilde{\mathbf{b}}_{\mathrm{t+1}}^{(i)}$ can be calculated according to (3.6):

$$\tilde{\mathbf{b}}_{\mathrm{t+1}}^{(1)} = \sum_{Ag^j \in G^1(t+1)} c_j^{(1)} \mathbf{b}_t^{(j)} = 0.33 \begin{pmatrix} 0.56 \\ -0.21 \\ 0.43 \end{pmatrix} + 0.33 \begin{pmatrix} 0.58 \\ 0.02 \\ 0.43 \end{pmatrix} + 0.33 \begin{pmatrix} 0.54 \\ -0.004 \\ 0.49 \end{pmatrix} = \begin{pmatrix} 0.56 \\ -0.06 \\ 0.45 \end{pmatrix}.$$

For a new point $\mathbf{x}_{t+1}^{(1)} = (3.7 \quad 2.8 \quad 1.1)$ an estimate $E\left[Y_{t+1}^{(1)}\right]$ after adjustment is equal to

$$\mathbf{x}_{t+1}^{(1)} \tilde{\mathbf{b}}_{\mathrm{t+1}}^{(1)} = (3.7 \quad 2.8 \quad 1.1) \begin{pmatrix} 0.56 \\ -0.06 \\ 0.45 \end{pmatrix} = 2.39.$$

Remind that the estimate without adjustment $E\left[Y_{t+1}^{(1)}\right] = 1.98.$ For this data point the corresponding true value of $Y_{t+1} = 2.5$. We can see that the adjustment facilitates better estimate of Y_{t+1}.

4. Distributed Cooperative Forecasting based on Kernel Regression

4.1. Kernel Density Estimation

Kernel density estimation is a non-parametric approach for estimating the probability density function of a random variable. Kernel density estimation is a fundamental data-smoothing technique where inferences about the population are made on the basis of a finite data sample. A kernel is a weighting function used in non-parametric estimation techniques.

Let $X_1, X_2, \cdots, X_n$ be an iid sample drawn from some distribution with an unknown density $f(x)$. We attempt to estimate the shape of $f(x)$, whose kernel density estimator is

$$\widehat{f_h}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - X_i}{h}\right), \tag{4.1}$$

where kernel $K(\bullet)$ is a non-negative real-valued integrable function satisfying the following two requirements: $\int_{-\infty}^{\infty} K(u) du = 1$ and $K(u) = K(-u)$ for all values of u; $h > 0$ is a smoothing parameter called bandwidth. The first requirement to $K(\bullet)$ ensures that the kernel density estimator is a probability density function. The second requirement to $K(\bullet)$ ensures that the average of the corresponding distribution is equal to that of the sample used [31]. Different kernel functions are available: Uniform, Epanechnikov, Gausian, etc. They differ in the manner in which they take into account the vicinity observations to estimate the function from the given variables.

An important problem in kernel density estimation is selection of the appropriate bandwidth h. It has an influence to the structure of neighbourhood in (4.1): the bigger h is selected, the more points X_i have significant influence to the estimator $\widehat{f_h}(x)$. In multi-dimensional case it regulates also a balance between factors.

The most of the bandwidth selection methods are based on the minimization of the integrated squared error $ISE(h)$ (or similar asymptotic or averaged characteristics) with respect to the bandwidth h. The integrated squared error is defined as

$$ISE(h) = \int \left[\widehat{f_h}(x) - f(x)\right]^2 dx.$$

However, the direct use of this formula is impossible due to involvement of unknown true density $f(x)$. So different heuristic approaches are used for this problem.

The methods of bandwidth selection can be divided into two main groups [32]:

(i) plug-in methods,

(ii) resampling methods.

The plug-in methods replace unknown terms in the expression for $ISE(h)$ by their heuristic approximations. One of the well known approaches is rule-of-thumb, proposed by Silvermann [33], which works for Gaussian kernels. In the case of no correlation between explanatory variables there is a simple and useful formula for bandwidth selection [34]:

$$h_j = n^{-1/(d+4)}\sigma_j, \ \ j = 1,2,\dots,d, \tag{4.2}$$

where σ_j is a variance of the j-th factor, d is a number of factors.

The resampling methods include cross-validation and bootstrap approaches. They use available data in different combinations to obtain approximation of $ISE(h)$. One of the most commonly used approaches is the minimisation of least squares cross-validation function $LSCV(h)$ introduced in [35]:

$$LSCV(h) = \int \left[\widehat{f_h}(x)\right]^2 dx - \frac{2}{n}\sum_{i=1}^{n} \widehat{f_{h,-\iota}}(X_i),$$

where $\widehat{f_{h,-\iota}}(X_i)$ is a kernel density estimation calculated without the i-th observation.

$LSCV(h)$ is a d-dimentional function, which can be minimised with respect to h.

A very suitable property of the kernel function is its additive nature. This property makes the kernel function easy to use for streaming and distributed data [31], [2], [17]. In [16], the distributed kernel-based clustering algorithm was suggested on the basis of the same property. In this study, kernel density is used for kernel regression to estimate the conditional expectation of a random variable.

4.2 Kernel-based local regression model

The non-parametric approach to estimating a regression curve has four main purposes. First, it provides a versatile method for exploring a general relationship between a dependent variable $\mathbf{Y}$ and factors **X**. Second, it can predict observations yet to be made without reference to a fixed parametric model. Third, it provides a tool for finding spurious observations by studying the influence of isolated points. Fourth, it constitutes the flexible method of substitution or interpolating between adjacent **X** values for missing observations [31].

Let us consider a non-parametric regression model [36] with a dependent variable $\mathbf{Y}$ and a vector of d regressors **X**

$$\mathbf{Y} = m(\mathbf{x}) + \boldsymbol{\varepsilon}, \tag{4.3}$$

where $\boldsymbol{\varepsilon}$ is a random error such that $\mathrm{E}[\boldsymbol{\varepsilon} \mid \mathbf{X} = \mathbf{x}] = \mathbf{0}$ and $\mathrm{Var}[\boldsymbol{\varepsilon} \mid \mathbf{X} = \mathbf{x}] = \sigma^2(\mathbf{x})$; and $m(\mathbf{x}) = \mathrm{E}[\mathbf{Y} \mid \mathbf{X} = \mathbf{x}]$. Further, let $(X_i, Y_i)_{i=1}^{n}$ be the observations sampled from the distribution of $(\mathbf{X}, \mathbf{Y})$. Then the Nadaraya-Watson kernel estimator is

$$\widehat{m}_n(\mathbf{x}) = \frac{\sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{X}_i}{h}\right) Y_i}{\sum_{i=1}^{n} K\left(\frac{\mathbf{x} - \mathbf{X}_i}{h}\right)} = \frac{p_n(\mathbf{x})}{q_n(\mathbf{x})}, \tag{4.4}$$

where $K(\bullet)$ is the kernel function of R^d and h is the bandwidth. Kernel functions satisfy the restrictions from (4.1). In our case we have a multi-dimentional kernel function $K(u) = K(u_1, u_2, \ldots, u_d)$ that can be easily presented with univariate kernel functions as: $K(u) = K(u_1) \cdot K(u_2) \cdot \ldots \cdot K(u_d)$. We used the Gaussian kernel in our experiments.

From formula (4.4), we can see that each value of the historical forecast Y_i is taken with some weight of the corresponding independent variable value of the same observation $\mathbf{X}_i$. Let us denote these weights in the following form:

$$\omega_i(\mathbf{x}) = K\left(\frac{\mathbf{x} - \mathbf{X}_i}{h}\right), \qquad \widehat{\omega}_i(\mathbf{x}) = \frac{\omega_i(\mathbf{x})}{\sum_{i=1}^{n} \omega_i}, \tag{4.5}$$

where $\omega_i(\mathbf{x})$ are the weights of the historical observations in the forecast of $\mathbf{x}$ and $\widehat{\omega}_i(\mathbf{x})$ are the corresponding normalised weights. Then, formula (4.4) can be rewritten as:

$$\widehat{m}_n(\mathbf{x}) = \frac{\sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{X}_i}{h}\right) Y_i}{\sum_{i=1}^{n} K\left(\frac{\mathbf{x}-\mathbf{X}_i}{h}\right)} = \frac{\sum_{i=1}^{n} \omega_i(\mathbf{x})\, Y_i}{\sum_{i=1}^{n} \omega_i} = \sum_{i=1}^{n} \widehat{\omega}_i(\mathbf{x}) Y_i. \tag{4.6}$$

Standard statistical and data mining methods usually deal with a dataset of a fixed size n as well as corresponding algorithms are functions of n. However for streaming data there is no fixed n: data are continually captured and must be processed as they arrive. It is important to develop algorithms that work with non-stationary datasets, handle the streaming data directly, and update their models on the fly [37]. For the kernel density estimator it is possible to use a simple method that allows recursive estimation of $\widehat{m}_n(\mathbf{x})$:

$$\widehat{m}_n(\mathbf{x}) = \frac{p_n(\mathbf{x})}{q_n(\mathbf{x})} = \frac{p_{n-1}(\mathbf{x}) + \omega_n(\mathbf{x}) Y_n}{q_{n-1}(\mathbf{x}) + \omega_n(\mathbf{x})}. \tag{4.7}$$

The quality of the kernel-based regression model can be verified by calculating R^2 as:

$$R^2 = \frac{\left[\sum_{i=1}^{n}(Y_i - \bar{Y})(\widehat{m}_n(\mathbf{X}_i) - \bar{Y})\right]^2}{\sum_{i=1}^{n}(Y_i - \bar{Y})^2 \sum_{i=1}^{n}(\widehat{m}_n(\mathbf{X}_i) - \bar{Y})^2}, \tag{4.8}$$

where $\bar{Y} = \sum_{i=1}^{n} Y_i / n$. It can be demonstrated that R^2 is identical to the standard measure for the linear regression model (3.4), fitted with least squares, and includes an intercept term.

The corresponding confidence interval for the forecast $\widehat{m}_n(\mathbf{x})$ with a confidence level $1-\alpha$ is represented by:

$$\left[\widehat{m}_n(\mathbf{x}) - z_{1-\frac{\alpha}{2}}\sqrt{\frac{\widehat{\sigma}^2(\mathbf{x})\|K\|_2^2}{nh\widehat{f}_h(\mathbf{x})}}, \widehat{m}_n(\mathbf{x}) + z_{1-\frac{\alpha}{2}}\sqrt{\frac{\widehat{\sigma}^2(\mathbf{x})\|K\|_2^2}{nh\widehat{f}_h(\mathbf{x})}}\right],$$

where $\|K\|_2^2 = [\int K^2(u)du]^2$, $\widehat{\sigma}^2(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} \widehat{\omega}_i(\mathbf{x})[Y_i - \widehat{m}_n(\mathbf{x})]^2$ and $z_{1-\frac{\alpha}{2}}$ is a $1-\frac{\alpha}{2}$ - level quantile of a standard normal distribution.

4.3 Cooperative Kernel-based Learning Algorithm

In this section, we describe the cooperation for sharing the forecasting experience among the agents in a network. While working with streaming data, one should take into account two main aspects:

(i) the nodes should coordinate their forecasting experience over some previous sampling period;

(ii) the nodes should adapt quickly to the changes in the streaming data, without waiting for the next coordination action.

As in the case of section 3.3, we have an MAS consisting of s autonomous agents $\boldsymbol{Ag} = \{Ag^{(1)}, Ag^{(2)}, \dots, Ag^{(s)}\}$, and each of them has a local kernel-based regression model, which is estimated on the basis of its experience.

Now, we introduce the following notations. Let $D^{(j)} = \left\{\left(\boldsymbol{X}_c^{(j)}, Y_c^{(j)}\right) | c = 1, \dots, n^{(j)}\right\}$ denote a local dataset of $Ag^{(j)}$, where $\mathbf{X}_c^{(j)}$ is a d-dimensional vector of factors on the c-th observation. In order to highlight the dependence of the forecasting function (4.4) on the local dataset of $Ag^{(j)}$, we denote the forecasting function by $\widehat{m}_n^{(j)}(x)$.

Consider a case when some agent $Ag^{(i)}$ wants to forecast for some d-dimensional future data point $\mathbf{x}_{n+1}^{(i)}$. Denote the weights and the corresponding normalised weights of formula (4.5) of the c-th observation from $D^{(i)}$ in the forecasting $\mathbf{x}_{n+1}^{(i)}$ as $\omega_c^{(i)}\left(\mathbf{x}_{n+1}^{(i)}\right)$ and $\widehat{\omega}_c^{(i)}\left(\mathbf{x}_{n+1}^{(i)}\right)$, respectively.

We consider a forecast for $\mathbf{x}_{n+1}^{(i)}$ as not reliable if only one of the observations is taken with a significant weight in this forecast:

$$\max\left(\widehat{\omega}_c^{(i)}\left(\mathbf{x}_{n+1}^{(i)}\right)\right) > bp,$$

where bp is a is an agent's parameter representing the maximal weight, after which a coordination takes place.

In this case, $Ag^{(i)}$ expects from other agents the points that are closer to the requested point than its ones points. For this purpose, it sends a request to other traffic participants within its transmission radius by sending the data point $\mathbf{x}_{n+1}^{(i)}$ as well as the threshold for the observation weight. This threshold is set as the weight of the second best observation:

$$Tr^{(i)}\left(\mathbf{x}_{n+1}^{(i)}\right) = \omega_{(n-1)}^{(i)}\left(\mathbf{x}_{n+1}^{(i)}\right),$$

where $\omega_{(1)}^{(i)}\left(\mathbf{x}_{n+1}^{(i)}\right), \omega_{(2)}^{(i)}\left(\mathbf{x}_{n+1}^{(i)}\right), \dots, \omega_{(n)}^{(i)}\left(\mathbf{x}_{n+1}^{(i)}\right)$ is an ordered sequence of weights in the forecast for $\mathbf{x}_{n+1}^{(i)}$.

Each $Ag^{(j)}$ that receives the request calculates the weights $\omega_c^{(j)}\left(\mathbf{x}_{n+1}^{(i)}\right)$ on the basis of its own data. If there are observations with weights $\omega_c^{(j)}\left(\mathbf{x}_{n+1}^{(i)}\right) > Tr^{(i)}\left(\mathbf{x}_{n+1}^{(i)}\right)$ it forms a reply $\widehat{D}^{(j,i)}$ from these observations (maximum 2) and sends it to $Ag^{(i)}$.

Let us define $G^{(i)} \subset \boldsymbol{Ag}$, as a group of agents, who are able to reply to $Ag^{(i)}$ by sending the requested data. All the data $\widehat{D}^{(j,i)}$, $Ag^{(j)} \in G^{(i)}$ received by $Ag^{(i)}$ are verified and duplicated data are discarded. These new observations are added to the dataset of $Ag^{(i)}$: $D^{(j)} \leftarrow \bigcup_{Ag^{(j)}\in G^{(i)}} \widehat{D}^{(j,i)} \cup D^{(j)}$. Suppose that $Ag^{(i)}$ received r observations. Then, the new kernel function of $Ag^{(i)}$ is updated by considering the additive nature of this function:

$$\widetilde{m}_{n+r}^{(i)}(\mathbf{x}) = \frac{p_n^{(i)}(\mathbf{x}) + \sum_{Ag^{(j)}\in G^{(i)}} p^{(i,j)}(\mathbf{x})}{q_n^{(i)}(\mathbf{x}) + \sum_{Ag^{(j)}\in G^{(i)}} q^{(i,j)}(\mathbf{x})}, \tag{4.9}$$

where $p^{(i,j)}(\mathbf{x})$ and $q^{(i,j)}(\mathbf{x})$ are the nominator and denominator of (4.4), respectively, calculated by $\widehat{D}^{(j,i)}$.

Finally, $Ag^{(i)}$ can autonomously make its forecast for $\mathbf{x}_{n+1}^{(i)}$as $\widetilde{m}_{n+r}^{(i)}(\mathbf{x}_{n+1}^{(i)})$.

4.4 Numerical example

To illustrate the implementation of kernel-based regression we provide an example using the same data we used for the linear regression:

$$\mathbf{X} = \begin{pmatrix} 5.4 & 3.9 & 2.2 \\ 1.7 & 4.6 & 3.5 \\ 3.2 & 2.3 & 1.2 \\ 4.3 & 2.1 & 3.2 \end{pmatrix}, \qquad \mathbf{Y} = \begin{pmatrix} 2.7 \\ 1.5 \\ 2.6 \\ 3.4 \end{pmatrix}.$$

Let us consider how an agent can calculate its forecasting function from formula (4.4). We take Gaussian kernel function and for a one-dimensional case, $K(u)$ is a density of the standard normal distribution $N(0,1)$ with the mean is 0 and the variance is equal to 1: $K(u) = \frac{1}{\sqrt{2\pi}} \mathrm{e}^{-\frac{1}{2}u^2}$.

In a multi-dimensional case, the kernel function is the product of one-dimensional kernels, as follows:

$$K(< u_1, u_2, \dots, u_d >) = K(u_1) \cdot K(u_2) \cdot \dots \cdot K(u_d).$$

We calculate the bandwidth $h = (1.30 \quad 1.00 \quad 0.86)$ according to (4.2) and we want to make a forecast according to formula (4.6) for some new point $\mathbf{x}_5 = (3.7 \quad 2.8 \quad 1.1)$ as:

$$\widehat{m}_4(\mathbf{x}_5) = \frac{\omega_1 \cdot 2.7 + \omega_2 \cdot 1.5 + \omega_3 \cdot 2.6 + \omega_4 \cdot 3.4}{\omega_1 + \omega_2 + \omega_3 + \omega_4},$$

$$\omega_1 = K\left(\frac{\mathbf{x}_5 - X_1}{h}\right) = K\left(\frac{\mathbf{x}_5 - (5.4 \quad 3.9 \quad 2.2)}{h}\right) = K\left(\frac{3.7 - 5.4}{1.30}\right)K\left(\frac{2.8 - 3.9}{1.00}\right)K\left(\frac{1.1 - 2.2}{0.86}\right) =$$
$$= K(-1.31)K(-1.10)K(-1.28) = 0.17 \cdot 0.22 \cdot 0.17 = 0.006,$$

$$\omega_2 = K\left(\frac{\mathbf{x}_5 - X_2}{h}\right) = K\left(\frac{\mathbf{x}_5 - (1.7 \quad 4.6 \quad 3.5)}{h}\right) = K\left(\frac{3.7 - 1.7}{1.30}\right)K\left(\frac{2.8 - 4.6}{1.00}\right)K\left(\frac{1.1 - 3.5}{0.86}\right) =$$
$$= K(1.54)K(-1.80)K(-2.80) = 0.12 \cdot 0.08 \cdot 0.008 = 7.5 \cdot 10^{-5},$$

$$\omega_3 = K\left(\frac{\mathbf{x}_5 - X_3}{h}\right) = K\left(\frac{\mathbf{x}_5 - (3.2 \quad 2.3 \quad 1.2)}{h}\right) = K\left(\frac{3.7 - 3.2}{1.30}\right)K\left(\frac{2.8 - 2.3}{1.00}\right)K\left(\frac{1.1 - 1.2}{0.86}\right) =$$
$$= K(0.39)K(0.50)K(-0.12) = 0.37 \cdot 0.35 \cdot 0.40 = 0.05,$$

$$\omega_4 = K\left(\frac{\mathbf{x}_5 - X_4}{h}\right) = K\left(\frac{\mathbf{x}_5 - (4.3 \quad 2.1 \quad 3.2)}{h}\right) = K\left(\frac{3.7 - 4.3}{1.30}\right)K\left(\frac{2.8 - 2.1}{1.00}\right)K\left(\frac{1.1 - 3.2}{0.86}\right) =$$
$$= K(-0.46)K(0.70)K(-2.45) = 0.36 \cdot 0.31 \cdot 0.02 = 0.002,$$

$$\widehat{m}_4(\mathbf{x}_5) = \frac{\omega_1 \cdot 2.7 + \omega_2 \cdot 1.5 + \omega_3 \cdot 2.6 + \omega_4 \cdot 3.4}{\omega_1 + \omega_2 + \omega_3 + \omega_4} = \frac{0.16}{0.06} = 2.64.$$

Next, we illustrate a cooperative learning algorithm. Let us denote the above considered agent as $Ag^{(1)}$. Suppose that the focal agent needs to make a forecast for the point $\mathbf{x}_5^{(1)} = (3.7 \quad 2.8 \quad 1.1)$.

The weights of the observations and the corresponding normalised weights during forecasting are:

$$\begin{pmatrix} \omega_1^{(1)} \\ \omega_2^{(1)} \\ \omega_3^{(1)} \\ \omega_4^{(1)} \end{pmatrix} = \begin{pmatrix} 0.006 \\ 7.5 \cdot 10^{-5} \\ 0.05 \\ 0.002 \end{pmatrix}, \qquad \begin{pmatrix} \widehat{\omega}_1^{(1)} \\ \widehat{\omega}_2^{(1)} \\ \widehat{\omega}_3^{(1)} \\ \widehat{\omega}_4^{(1)} \end{pmatrix} = \begin{pmatrix} 0.107 \\ 0.001 \\ 0.856 \\ 0.036 \end{pmatrix}.$$

The maximal weight $\widehat{\omega}_3^{(1)} = 0.856$ is greater than $bp = 0.8$, so the agent requests cooperation. We test the following threshold $Tr^{(1)} = 0.006$.

We suppose that in the neighbourhood of the focal agent are two agents $Ag^{(2)}$ and $Ag^{(3)}$ with datasets:

$$\mathbf{X}^{(2)} = \begin{pmatrix} 4.1 & 2.5 & 1.3 \\ 0.4 & 3.7 & 3.2 \\ 3.1 & 3.4 & 0.7 \\ 5.4 & 0.7 & 0.3 \end{pmatrix}, \quad \mathbf{Y}^{(2)} = \begin{pmatrix} 2.6 \\ 1.8 \\ 2.3 \\ 3.5 \end{pmatrix}, \quad \mathbf{X}^{(3)} = \begin{pmatrix} 5.0 & 2.7 & 3.5 \\ 3.2 & 2.2 & 1.4 \\ 3.3 & 3.4 & 1.7 \\ 0.8 & 4.3 & 1.2 \end{pmatrix}, \quad \mathbf{Y}^{(3)} = \begin{pmatrix} 4.4 \\ 2.4 \\ 2.6 \\ 1.0 \end{pmatrix}.$$

They calculate the weights $\omega^{(j)}, j = 2..3$, on the basis of their own historical data in the same manner as the focal agent has done it in the beginning of this example. Let the obtained weights are:

$$\begin{pmatrix} \omega_1^{(2)} \\ \omega_2^{(2)} \\ \omega_3^{(2)} \\ \omega_4^{(2)} \end{pmatrix} = \begin{pmatrix} 0.06 \\ 8.2 \cdot 10^{-5} \\ 0.04 \\ 1.9 \cdot 10^{-3} \end{pmatrix}, \quad \begin{pmatrix} \omega_1^{(3)} \\ \omega_2^{(3)} \\ \omega_3^{(3)} \\ \omega_4^{(3)} \end{pmatrix} = \begin{pmatrix} 7.5 \cdot 10^{-4} \\ 0.05 \\ 0.04 \\ 1.6 \cdot 10^{-3} \end{pmatrix}.$$

Both agents $Ag^{(2)}$ and $Ag^{(3)}$ have observations with the weights $\omega_i^{(j)} > Tr^{(1)}$. So they send these observations, as follows:

$$\widehat{D}^{(2,1)} = \left(\begin{bmatrix} 4.1 & 2.5 & 1.3 \\ 3.1 & 3.4 & 0.7 \end{bmatrix}, \begin{bmatrix} 2.6 \\ 2.3 \end{bmatrix} \right), \widehat{D}^{(3,1)} = \left(\begin{bmatrix} 3.2 & 2.2 & 1.4 \\ 3.3 & 3.4 & 1.7 \end{bmatrix}, \begin{bmatrix} 2.4 \\ 2.6 \end{bmatrix} \right).$$

Next, $Ag^{(1)}$ updates its matrices:

$$\mathbf{X}^{(1)} = \begin{pmatrix} 5.4 & 3.9 & 2.2 \\ 1.7 & 4.6 & 3.5 \\ 3.2 & 2.3 & 1.2 \\ 4.3 & 2.1 & 3.2 \\ 4.1 & 2.5 & 1.3 \\ 3.1 & 3.4 & 0.7 \\ 3.2 & 2.2 & 1.4 \\ 3.3 & 3.4 & 1.7 \end{pmatrix}, \quad \mathbf{Y}^{(1)} = \begin{pmatrix} 2.7 \\ 1.5 \\ 2.6 \\ 3.4 \\ 2.6 \\ 2.3 \\ 2.4 \\ 2.6 \end{pmatrix}$$

and calculates the new weights and the corresponding normalised weights during forecasting for the data point $\mathbf{x}_5^{(1)} = (3.7 \quad 2.8 \quad 1.1)$:

$$\begin{pmatrix} \omega_1^{(1)} \\ \omega_2^{(1)} \\ \omega_3^{(1)} \\ \omega_4^{(1)} \\ \omega_5^{(1)} \\ \omega_6^{(1)} \\ \omega_7^{(1)} \\ \omega_8^{(1)} \end{pmatrix} = \begin{pmatrix} 0.006 \\ 7.5 \cdot 10^{-5} \\ 0.05 \\ 0.002 \\ 0.06 \\ 0.04 \\ 0.05 \\ 0.04 \end{pmatrix}, \quad \begin{pmatrix} \hat{\omega}_1^{(1)} \\ \hat{\omega}_2^{(1)} \\ \hat{\omega}_3^{(1)} \\ \hat{\omega}_4^{(1)} \\ \hat{\omega}_5^{(1)} \\ \hat{\omega}_6^{(1)} \\ \hat{\omega}_7^{(1)} \\ \hat{\omega}_8^{(1)} \end{pmatrix} = \begin{pmatrix} 0.026 \\ 3.1 \cdot 10^{-4} \\ 0.211 \\ 0.009 \\ 0.230 \\ 0.174 \\ 0.189 \\ 0.161 \end{pmatrix}.$$

Now, we can make the estimation based on the new data:

$$\hat{m}_8(\mathbf{x}) =$$
$$= \frac{\omega_1^{(1)} \cdot 2.7 + \omega_2^{(1)} \cdot 1.5 + \omega_3^{(1)} \cdot 2.6 + \omega_4^{(1)} \cdot 3.4 + \omega_5^{(1)} \cdot 2.6 + \omega_6^{(1)} \cdot 2.3 + \omega_7^{(1)} \cdot 2.4 + \omega_8^{(1)} \cdot 2.6}{\omega_1^{(1)} + \omega_2^{(1)} + \omega_3^{(1)} + \omega_4^{(1)} + \omega_5^{(1)} + \omega_6^{(1)} + \omega_7^{(1)} + \omega_8^{(1)}} =$$
$$= \frac{0.619}{0.245} = 2.52.$$

Remind that for this data point the corresponding true value $Y_{t+1} = 2.5$. We can see that the adjustment facilitates better estimate $\hat{m}_8(\mathbf{x}) = 2.52$ of Y_{t+1} while before adjustment the estimate was $\hat{m}_4(\mathbf{x}) = 2.61$.

5. Case Studies

5.1. Problem formulation

We simulated a traffic network in the southern part of Hanover (Germany). The network contained three parallel and five perpendicular streets, which formed 15 intersections with a flow of approximately 5000 vehicles per hour. The network is shown in Figure 4.

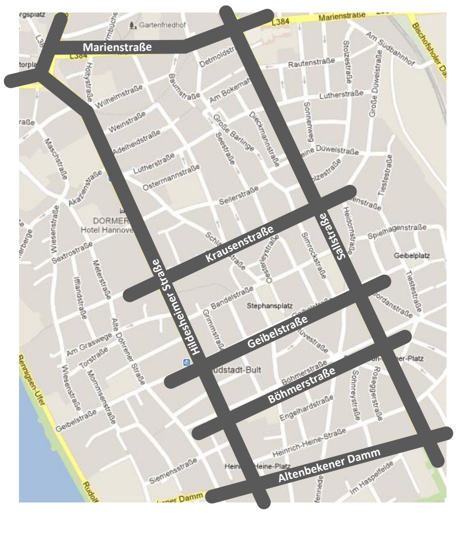

Figure 4. Road network in the southern part of Hanover, Germany.

We assumed that only a small proportion of vehicles were equipped with corresponding devices, which helped to solve the travelling time forecasting problem. These devices could communicate with other vehicles and make the necessary calculations used by DDPM module. In the current case, the DDPM module was implemented for regression based forecasting models. Other vehicles played the role of the 'environment', creating corresponding traffic flows.

The vehicles received information from TIC about the centrally estimated system variables (such as average speed, number of stops, congestion level, etc.) in this city district, which they combined with their stored historical information, before making adjustments after exchanging information with other vehicles using cooperative learning algorithms.

We assumed that each traffic participant considered the same factors during travelling time forecasting. These factors are shown in Table 1. Only significant factors were also used, which were tested specifically for linear regression model using the *t*-distribution.

Table 1. Factors used for travelling time forecasting.

Variable	Description
Y	travelling time (min);
X_1	route length (km)
X_2	average speed in system (km/h)
X_3	average number of stops(units/min)
X_4	congestion level (vehicles/h)
X_5	number of the traffic lights in the route (units)
X_6	number of the left turns in the route (units)

To obtain more reliable results, we considered the travel frequency based on statistics from a survey [38] in Germany (2012) of drivers aged over 14 years, as shown in Figure 5.

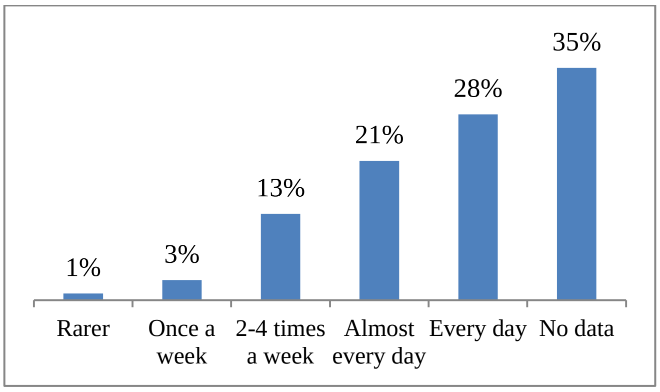

Figure 5. Frequency of driving according to the survey [38].

The system was implemented and simulated in the R package. The agents were trained using the available real-world dataset (n = 6500). The data were normalised in the interval [0,1]. No data transformation was performed because experiments showed that they provide no additional quality improvements in the current models. During kernel-based regression, we used bandwidths h calculated by formula (4.2), which are listed in Table 2.

Table 2. Factors and corresponding bandwidth values h.

Variable	X_1	X_2	X_3	X_4	X_5	X_6
Bandwidth	h_1	h_2	h_3	h_4	h_5	h_6
Bandwidth value	0.01	0.03	0.2	0.2	0.1	0.05

We combined the linear and kernel-based regression results using two additional estimates. The first estimate was the optimum, were we supposed an ‘oracle’ model helped to select the best forecast (with a low forecasting error) from every forecast. The oracle model represented the best forecast that could be achieved using the linear and kernel regressions.

The second ‘average’ estimate was the average of the kernel and linear estimates. There was a strong positive correlation between the linear and kernel estimates (about 0.8) but we demonstrated that the average estimate was often better than the kernel or linear estimates.

We conducted experiments using three types of models: a centralised architecture with an authority, a decentralised uncoordinated, and a decentralised coordinated architecture. The linear and kernel-based regression models were implemented in each model, as well as their combinations (oracle and average estimates).

We compared the results by analysing the average forecasting errors (3.7), the relative forecasting errors (3.8) and coefficients of determination R^2 (3.4 and 4.8). These characteristics are well-known measures of the effectiveness in predicting the future outcomes using regression models and they can be used with parametric and non-parametric models [39].

5.2. Centralised architecture

This model assumes that agents transmit their observations to a central authority, which requires the transmission of a large amount of data so it is very expensive. However, its forecast precision is the best of all the models considered because this method is based on all datasets.

First, we considered the linear regression model described in section 3.2. This model was constructed using 2000 observations, which were selected randomly from the available dataset of 6500. The parameter estimates are shown in Table 3. All of the parameters satisfied the significance test, which was a measure of the effect on the target variable, i.e., the travelling time. These parameters could be used as the ‘true’ parameter values in linear models with decentralised architectures.

Table 3. Factors and corresponding parameters.

Variable	X_1	X_2	X_3	X_4	X_5	X_6
Coefficient estimate	b_1	b_2	b_3	b_4	b_5	b_6
Estimated value	0.222	-0.032	0.003	0.056	0.086	-0.017

Second, we also considered kernel-based regression model for the same data.

To test the efficiency of the models by cross-validation, we used 500 observations that were not used in the construction of the model. The corresponding R^2 values are provided in Table 4, which shows that kernel regression gave more reliable results with $R^2 = 0.913$.

Table 4. Values of the coefficient of determination R^2 using the centralised approach.

	Linear	Kernel
R^2 value	0.829	0.913

Figure 6 shows the average forecasting errors for the linear, kernel-based, oracle, and average models. Clearly, the kernel model yielded better results compared with the linear model (c.15% smaller errors). The kernel model and average model yielded the same results after 300 time units, except the average model had 5% smaller standard deviation of errors. Before 300 time units, the average model was slightly better. In addition, the oracle model gave the best results, i.e. 25% better than the kernel model. The best method for constructing the oracle model remains unclear.

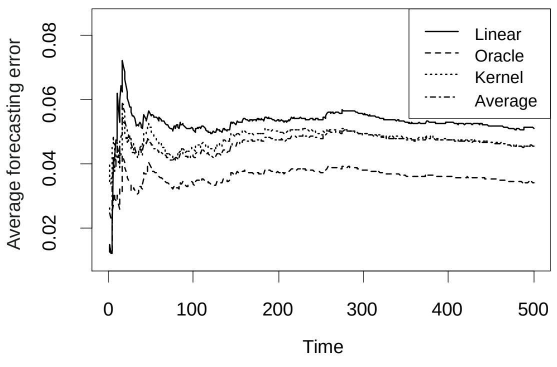

Figure 6. Average forecasting errors using different models.

5.3. Decentralised uncoordinated architecture

In this section, we consider an estimation model with fully decentralised uncoordinated architecture. There were no transmission costs in this case, although it was assumed that each agent was equipped with a special device for making forecasts. The main problem was that each agent constructed its own local model using a relatively small amount of data. This led to major forecasting errors, which could not be corrected due to the lack of coordination.

We simulated 20 agents (vehicles) with an initial experience of 20 observations. The simulation experiments ran for 150 time units. The data used in the simulations were homogeneous. However, the LSE function (3.2) used for parameter estimation in the linear model had a number of local optima. If the parameters **b** of the agent fell into these local optima, could remain there for a long time. A typical situation is shown in Figure 7.

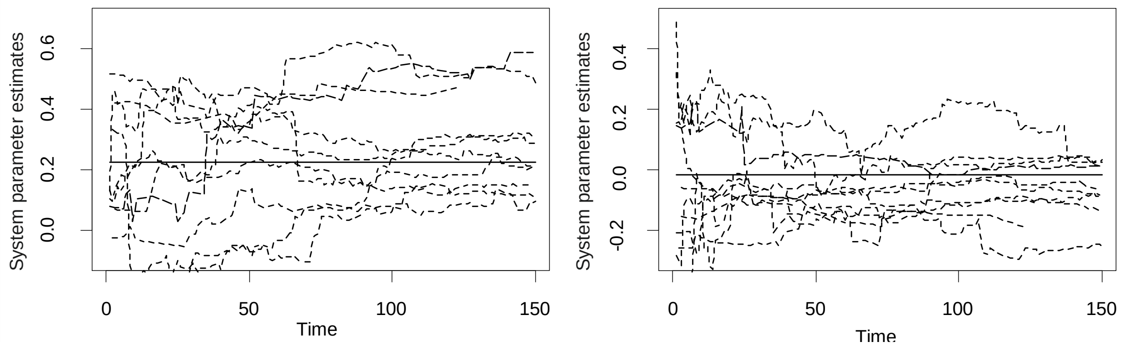

Figure 7. System parameter estimates $\mathbf{b_1}$ (left) and $\mathbf{b_6}$ (right) for a linear model using different agents.

Figure 7 shows the dynamics when changing the parameter values over time with several agents. The straight horizontal line shows the ‘true’ value of the corresponding parameter estimated in the centralised model. It can be seen that some agents had parameters that did not converge. Moreover, ‘clusters’ of agents were observed that converged to different values. Clearly, this had a negative effect on the quality of the forecasts.

The quality of the agent models was checked by cross-validation. In this case, the cross-validation checked the i-th agent’s model based on observations of the other agents as test data. This procedure generated 20 coefficients of determination R^2 (one for each agent) values for the kernel-based and linear models. Figure 8 shows the corresponding histograms. The kernel model was better with a higher number of agents, although its performance degraded for some agents.

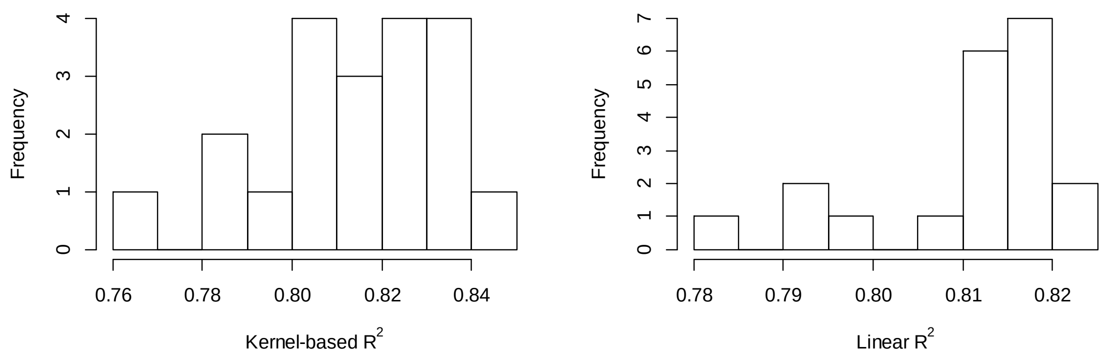

Figure 8. R^2 values for the overall system based on kernel (left) and linear (right) models.

As with the centralised architecture, we compared the average forecasting errors for the linear, kernel-based, oracle and average models. The linear and kernel model results were approximately the same quality (the kernel model produce c. 2% smaller errors). However, the average model yielded considerably better results (6% smaller errors compared with the kernel model). The oracle model yielded the best results, i.e. c. 35% better than the kernel model.

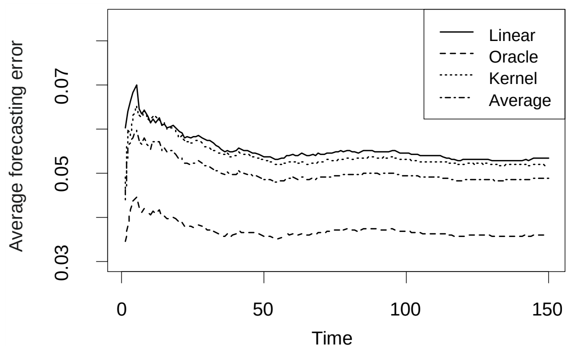

Figure 9. Average forecasting errors using the kernel, linear, and combined models.

5.4. Decentralised coordinated model

This section considers an estimation model with decentralised coordinated architecture. The agents made a local estimation of the system parameters and used cooperative mechanisms to adjust their parameters (linear model) or observations (kernel-based model) according to those of other agents. The amount of information transmitted was lower than that in the centralised model, because only locally estimated parameters were transmitted rather than global data.

We used the same simulation parameters as those used in the uncoordinated architecture, i.e. 20 agents with the initial experience of 20 observations. Most simulation experiments ran for 100 time units. The agents were equipped with a special device for making forecast calculations and communicating. In the linear model, the experiences of agents were used as the weights (reliability level) for their parameters during the adjustment process. In the kernel-based, model the agents transmitted two best observations.

First, we show the system dynamics for one randomly selected agent (Figure 10). Large errors disappeared over time and the number of communication events also decreased.

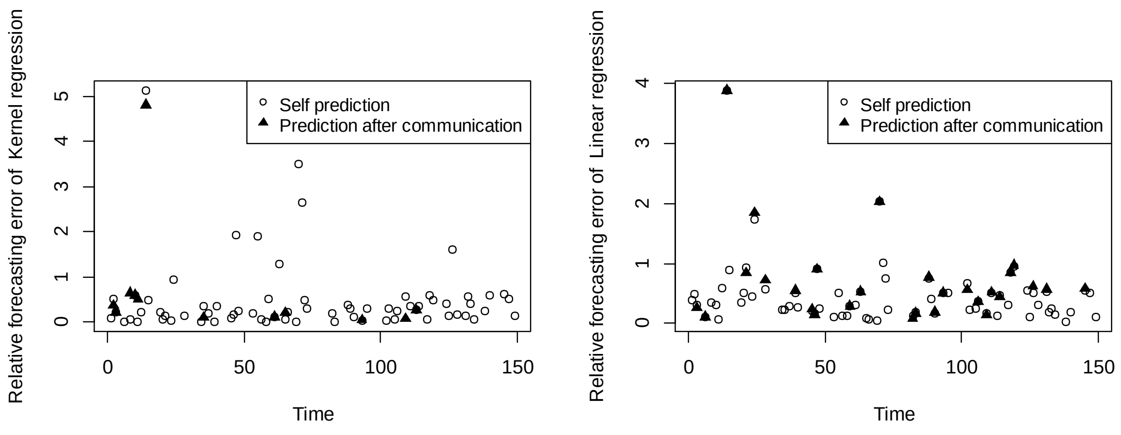

Figure 10. System dynamics of single agent using kernel and linear models, $bp = 1.5, p = 0.85$.

We analysed the number of communication events in greater detail. Figure 11 shows that the number of communication events with the linear and kernel-based models depended on the model parameters p and bp, which regulated the communication conditions. There was a significant decrease in the number of communication events with the kernel-based model. With the linear model, there was a significant difference in the number of communication events that depended on parameters that did not decrease over time. The number of communication events also decreased slightly with time.

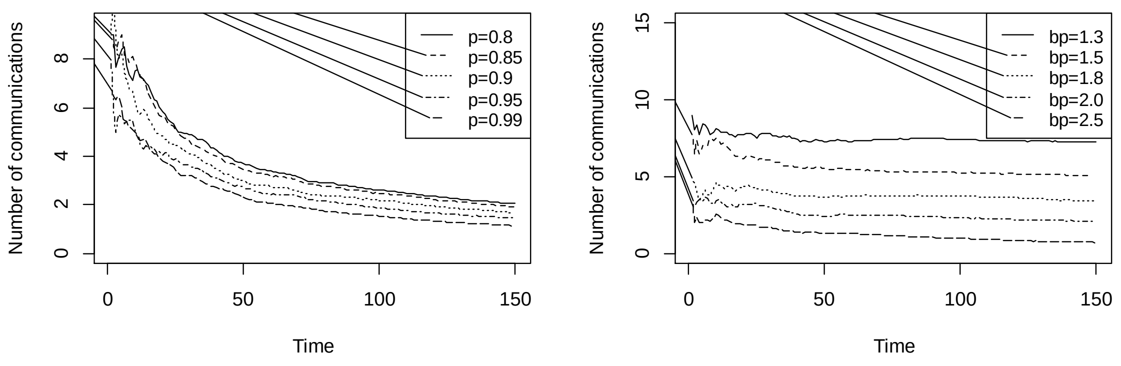

Figure 11. Number of communication events using the kernel and linear models.

We analysed the dynamics of the parameter changes in the linear model (Figure 12). The straight horizontal line shows the ‘true’ value of the corresponding parameter, which was estimated by the centralised architecture. There was good convergence of the parameters to the ‘true’ values. Further, there were no clusters of agents with different parameters, so the quality of forecasts should be good.

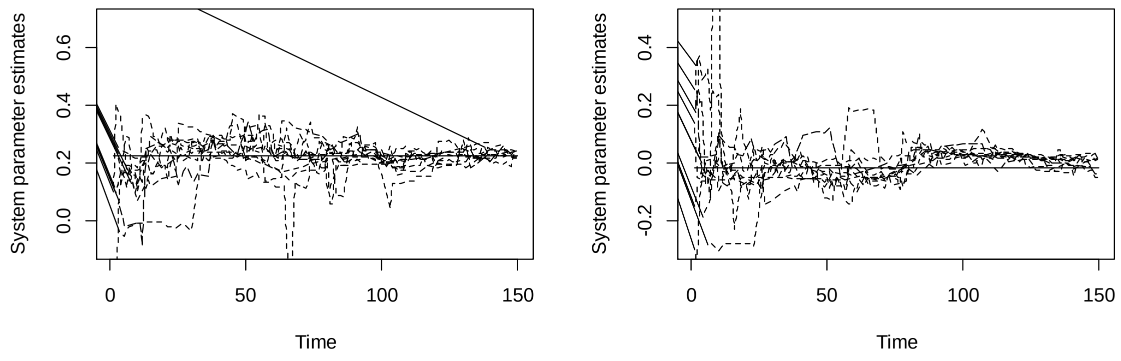

Figure 12. System parameter estimates b_1 (left) and b_6 (right) for linear model with different agents, $\boldsymbol{bp = 1.5, p = 0.85}$.

The quality of the agent models was also checked by cross-validation technique, as with the previous model. Figure 13 shows that the kernel model yielded better results than the previous uncoordinated architecture. The linear model did not yield much better results, but the ‘bad’ agents disappear so all agents had equal R^2 values. Thus, it is necessary to make a trade-off between system accuracy (represented by R^2) and the number of necessary communication events. The trade-off depended on the communication and accuracy costs.

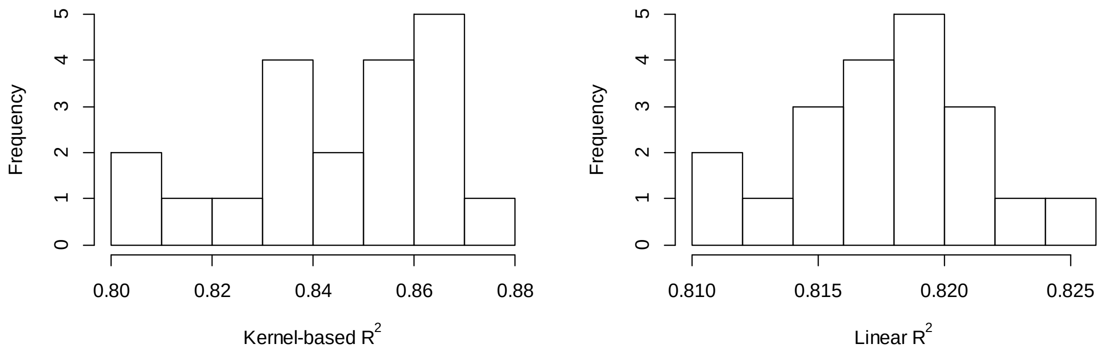

Figure 13. $\boldsymbol{R^2}$ for the whole system for kernel (left) and linear (right) models, $\boldsymbol{bp = 1.5, p = 0.85}$.

As previously discussed and shown in Figure 11, changes in the model parameter values p and bp affected the number of communication events. Fewer communication events meant less precision. Figure 14 shows how the dynamics of the average forecasting errors changed, depending on the system parameters. With the kernel model, a change in parameter p from 0.8 to 0.99 decreased the errors by ≥12% while the number of communication events increased by 35%. Changes in the linear model parameter bp from 1.3 to 2.5 yielded approximately the same increase in quality of about 10%, however the number of communication events increased by 300%. This demonstrated the importance of making a trade-off between an amount of communication and accuracy.

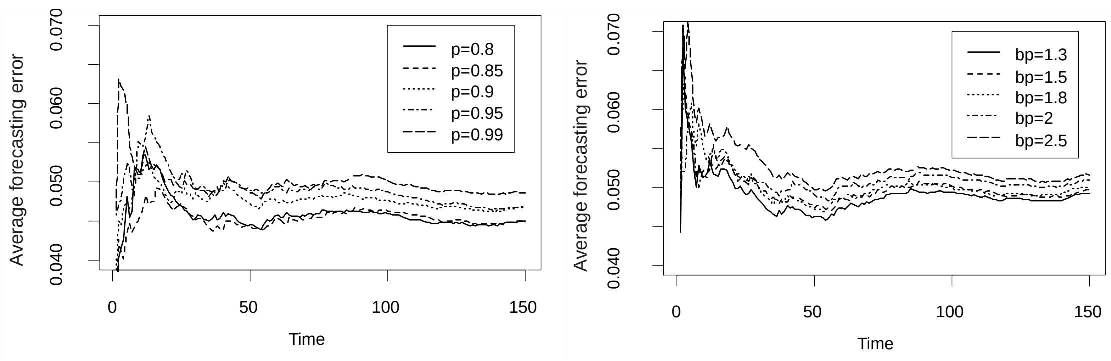

Figure 14. Average forecasting errors using kernel-based and linear models.

We compared the average forecasting errors using different models (Figure 15), as with the centralised and uncoordinated architecture. The results were similar to the centralised architecture results, i.e., the kernel model was c. 5% better than the linear model. The average estimate was similar to the kernel estimate, but it was slightly better (2%). The oracle model is considerably better than the other estimates (c. 40% better than the kernel model).

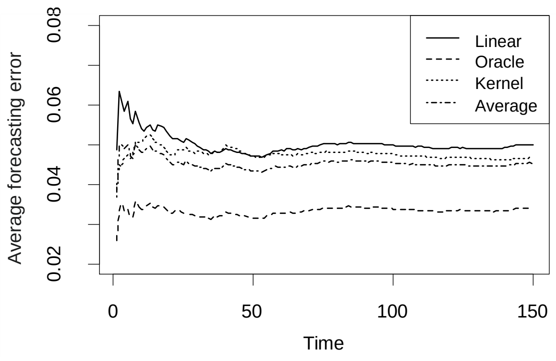

Figure 15. Average forecasting errors using the kernel, linear and combined approaches, $bp = 1.5$, $p = 0.85$.

In the coordinated architecture, the agents with more experience (historical dataset) helped the agents with less experience to improve their forecasts. This cooperation helped to improve the quality of forecasts, especially those of less experienced agents. As a result, the average quality of the system was increased compared with the uncoordinated architecture (by c. 15% for the kernel-based, c. 8% for linear, c. 6% for the average, and c. 9% for the oracle estimate). There was almost no difference from the centralised architecture, although the amount of communication was smaller and the communication events occurred at a local level.

6. Discussion

Table 5 summarises the results of the average forecasting errors and goodness-of-fit criteria R^2 using the different forecasting models (linear, kernel, average, and oracle) with various architectures (centralised, decentralised uncoordinated/coordinated). This demonstrated the advantages and disadvantages of the algorithms we implemented.

The linear regression model used all the data points for forecasting. This was is a positive aspect if there were no historical observations close to the requested point. However, the equal effect of the distant and nearby data points, and the linear dependency, led to inaccurate forecasts. Furthermore, the coordination of the parameters always increased the squared error calculated using the agent’s data because the current parameters were calculated based on the squared error minimisation. This meant that the straight regression line was being adjusted continuously via coordination and new data points, so it could not be fitted well for all data points. Thus, linear regression gave the worst forecasts: $R^2 = 0.83$ for the centralised architecture and $R^2 = 0.82$ for the coordinated architecture. The centralised architecture was worse because a single line could not make good forecasts for 2000 data points. Figure 6 shows that there was a decrease in the average forecasting errors with about 100 points, followed by an increase. The linear model within the uncoordinated architecture was worse due to the convergence of the parameters of several agents to local optima instead of global optima.

Table 5. Average forecasting errors and goodness-of-fit criteria R^2 for the different forecasting models, $bp = 1.5, p = 0.85$.

Model	Average forecasting errors	R^2 (After cross-validation)
Linear		
Centralised architecture	0.051	0.829
Uncoordinated architecture	0.054	0.811
Coordinated architecture	0.050	0.819
Kernel-based		
Centralised architecture	0.045	0.913
Uncoordinated architecture	0.053	0.814
Coordinated architecture	0.047	0.859
Aggregated (average)		
Centralised architecture	0.046	---
Uncoordinated architecture	0.049	---
Coordinated architecture	0.046	---
Aggregated (oracle)		
Centralised architecture	0.034	---
Uncoordinated architecture	0.037	---
Coordinated architecture	0.034	---

The kernel-based regression model used only neighbouring points for forecasting. In contrast to the linear model, the absence of nearby points prevented good forecasting. However, there was no strong linear dependency and the model fitted well to the data points with non-linear dependencies. Coordination greatly improved the quality of forecasting because new neighbouring data points were provided. The centralised architecture made the best forecasts because there were many nearby points for most requested point ($R^2 = 0.91$). However, absence of experience (uncoordinated architecture) produced bad forecasts ($R^2 = 0.81$), which were comparable with the results of the linear model. Coordination improved the R^2 to 0.86 (with the linear model only to 0.82). For large amounts of data kernel-based model required numerous computations (all data points needed to be checked to ensure nearby points for each forecast).

With the average estimator, i.e. the mean of the kernel-based and linear estimators, R^2 could not be calculated because of the nature of the model. However, analysis of the relative forecasting errors showed that the average estimator in coordinated and uncoordinated architectures for relatively small amounts of data produced relatively better results than kernel-based or linear estimators. This was not explained by a negative correlation between the linear and kernel-based forecasting errors, because there was a strong positive correlation of about 0.8), and instead in was due to the structure of these errors. As shown in Figure 10, the kernel model was more accurate on average, but it sometimes yielded highly inaccurate forecasts (if only one neighbouring observation was used for forecasting). The linear model was less accurate, but it did not produce such big outliers. The average method avoided big outliers of the kernel model and it provided more accurate forecasts. However, this advantage was lost when the kernel model had sufficient data to avoid outliers (in Figure 6 this occurred after c. 300 observations).

The oracle-based estimator, which combined the kernel-based and linear estimators, provided a possible lower bound for the average forecasting error. This algorithm could improve forecasts by c. 25%. However, it is still unclear, how to choose between kernel-based and linear estimates. It is important to identify the factors or characteristics of estimators, which allow a comparison of linear and kernel-based regression estimates for one actual point.

7. Conclusions

This study considered the problem of intelligent data processing and mining in decentralised multi-agent networks. We focused on cooperative regression-based travelling time forecasting in distributed traffic networks.

We proposed a multi-agent architecture and structure for intelligent autonomous agents and paid special attention to the data processing and mining modules. This approach allows the input data processing and making travelling time forecasting, while adjusting the agent models if necessary.

We analysed linear and kernel-based regression models, which can be applied efficiently for travelling time forecasting. Regression models were tested with special attention to their implementation for streaming data, which is important for intensive data flows in traffic systems. We proposed regression-based coordination learning algorithms based on the confidence intervals of estimates, which allowed agents to improve the quality of their forecasts. Each model was illustrated with simple examples in a tutorial style, which facilitated understanding of the algorithms.

We demonstrated the practical application of the proposed approaches using real-world data from Hanover, Germany. We proposed the structure of regression models by selecting significant factors. Three types of architectures were compared: distributed centralised and decentralised coordinated/uncoordinated.

For each of the proposed architectures, we tested accuracy of the linear and kernel-based regression estimators of the travelling time, as well as combinations of both. We used the relative error and determination coefficient as goodness-of-fit criteria. The quality of each agent model was checked by cross-validation.

The results demonstrated the appropriate goodness-of-fit of all the models considered (>0.8). The kernel model usually produces better results compared with the linear model, although it was more sensitive to outliers. A simple combination of the linear and kernel-based estimates (average) reduced the effect of outliers and provided better estimates with small amounts of data (<300). However, a more effective algorithm for choosing between the kernel-based and linear estimation method could theoretically yield 25–30% better results, which was demonstrated by the oracle model.

The centralised architecture provided the best quality of forecasts. However, it required a large amount of communication and computation, which was frequently impossible to implement. In decentralised architectures, the cooperation between agents and the corresponding model adjustment improved the forecasts considerably. It was possible to obtain almost the same average forecasting errors as in centralised architectures, especially when we aggregated the kernel-based and linear estimates. This showed that a decentralised architecture with communication could be a good alternative to the centralised architecture, although it dispenses with the expensive authority, uses a fewer amount of communication and computation, which are mostly local and executed in parallel.

Our future work will be continued in three directions: (a) construction of the distributed model of multiple multivariate regression, which allows forecasting of several response variables simultaneously from the same set of explanatory variables (factors); (b) application of other regression model types and different methods of combination of their estimates; (c) modification of the parameter adjustment algorithm (new strategies for the calculation of the reliability level of agents, resampling approach [40], etc.) in order to be more reliable to outliers.

ACKNOWLEDGEMENTS

The research leading to these results has received funding from the European Union Seventh Framework Programme (FP7/2007-2013) under grant agreement No. PIEF-GA-2010-274881. We thank also Prof. J.P. Müller for his useful and valuable ideas during this paper preparation.

REFERENCES

[1] H. Kargupta and P. Chan, Eds., Advances in Distributed and Parallel Knowledge Discovery, California, USA: AAAI Press/MIT Press, 2000.

[2] J. C. da Silva, C. Giannella, R. Bhargava, H. Kargupta and M. Klusch, "Distributed data mining and agents," *Eng. Appl. Artif. Intell.,* vol. 18, no. 7, pp. 791-801, 2005.

[3] H. Lin, R. Zito and M. Taylor, "A review of traveltime prediction in transport and logistics," *Proceedings of the Eastern Asia Society for Transportation Studies,* vol. 5, pp. 1433-1448, 2005.

[4] M. Fiosins, J. Fiosina, J. Müller and J. Görmer, "Agent-based integrated decision making for autonomous vehicles in urban traffic," *Advances on Practical Applications of Agents and Multiagent Systems (Advances in Intelligent and Soft Computing),* vol. 88, pp. 173-178, 2011.

[5] M. Fiosins, J. Fiosina, J. P. Müller and J. Görmer, "Reconciling Strategic and Tactical Decision Making in Agent-Oriented Simulation of Vehicles in Urban Traffic," in *Proc. of 4th Int. ICST Conf. on Simulation Tools and Techniques (SimuTools'2011),* pp. 144-151, 2011.

[6] H. A. Simon, The Sciences of the Artificial, MIT Press, 1996.

[7] N. R. Jennings, "An agent-based approach for building complex software systems.," *Communications of the ACM,* vol. 44, no. 4, pp. 35-41, 2001.

[8] M. Wooldridge, An Introduction to Multi-Agent Systems, John Wiley & Sons, 2002.

[9] P. A. Mitkas, D. Kehagias, A. L. Symeonidis and I. N. Athanasiadis, "A Framework for Constructing Multi-Agent Applications and Training Intelligent Agents," in *In Proc. of the 4-th Int.l Workshop on Agent-oriented Software Engineering (AOSE-2003) in conjunction with Conference on Automous Agents and Multi-Agent Systems (AAMAS 2003},* pp. 96-109, 2003.

[10] A. Bazzan, "Opportunities for multiagent systems and multiagent reinforcement learning in traffic control," *Autonomous Agents and Multiagent Systems,* vol. 18, no. 3, pp. 342-375, 2009.

[11] M. Rieser, K. Nagel, U. Beuck, M. Balmer and J. Rümenapp, "Truly agent-oriented coupling of an activity-based demand generation with a multi-agent traffic simulation," *Transportation Research Record,* vol. 2021, pp. 10-17, 2007.

[12] P. Davidsson, L. Henesey, L. Ramstedt, J. Törnquist and F. Wernstedt, "An analysis of agent-based approaches to transport logistics," *Transportation Research Part C: Emerging Technologies,* vol. 13, no. 14, pp. 255-271, 2005.

[13] A. Freitas, Data Mining and Knowledge Discovery with Evolutionary Algorithms, Springer, 2002, 264 p.

[14] A. L. Symeonidis and P. A. Mitkas, “Agent Intelligence Through Data Mining,” in *Multiagent Systems, Artificial Societies, and Simulated Organizations*, New York, Springer-Verlag, 2005.

[15] L. Cao, D. Luo and C. Zhang, “Ubiquitous Intelligence in Agent Mining,” *Agents and Data Mining Interaction (Lecture Notes in Computer Science),* vol. 5680, pp. 23-35, 2009.

[16] M. Klusch, S. Lodi and G. Moro, “Agent-based distributed data mining: The kdec scheme,” *Proc. of Int. Conf. on Intelligent Information Agents - The AgentLink Perspective; Lecture Notes in Computer Science,* vol. 2586, pp. 104-122, 2003.

[17] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin and S. Madden, “Distributed regression: an efficient framework for modeling sensor network data,” in *Proc. of the 3rd Int. Sym. on Information Processing in Sensor Networks,* Berkeley, California, USA, pp. 1-10, 2004.

[18] S. S. Stankovic, M. S. Stankovic and D. M. Stipanovic, “Decentralized parameter estimation by consensus based stochastic approximation,” *IEEE Trans. Automatic Controll,* vol. 56, pp. 1535-1540, 2009.

[19] A. Bazzan, J. Wahle and F. Kluegl, “Agents in traffic modelling - from reactive to social behaviour,” *Advances in Artificial Intelligence (Lecture Notes in Artificial Intelligence),* vol. 1701, pp. 509-523, 1999.

[20] G. Malnati, C. Barberis and C. Cuva, “Gossip: Estimating actual travelling time using vehicle to vehicle communication,” in *Proceedings of Fourth International Workshop on Intelligent Transportation*, Hamburg, 2007.

[21] W. Lee, S. Tseng and W. Shieh, “Collaborative real-time traffic information generation and sharing framework for the intelligent transportation system,” *Information Scienses,* vol. 180, pp. 62-70, 2010.

[22] J. Ehmke, M. Fiosins, J. Görmer, D. Schmidt, H. Schumacher and H. Tchouankem, “Decision Support for Dynamic City Traffic Management Using Vehicular Communication,” in *Proc. of 1st International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2011),* pp. 327-332, 2011.

[23] R. Claes and T. Holvoet, “Ad hoc link traversal time prediction,” in *Proceedings of the 14th International IEEE conference on Intelligent Transportation Systems,* pp. 1803-1808, 2011.

[24] J. Fiosina, “Decentralised regression model for intelligent forecasting in multi-agent traffic networks,” *Distributed Computing and Artificial Intelligence (Advances in Intelligent and Soft Computing),* vol. 151, p. 255–263, 2012.

[25] J. Fiosina and M. Fiosins, “Cooperative Kernel-Based Forecasting in Decentralized Multi-Agent Systems for Urban Traffic Networks,” in *Proc. of Ubiquitous Data Mining (UDM) Workshop at the 20th European Conference on Artificial Intelligence,* pp. 3-7, 2012.

[26] B. L. Smith, B. L. Williams and R. K. Oswald, “Comparison of parametric and nonparametric models for traffic flow forecasting,” *Transportation Research Part C,* vol. 10, pp. 303-321, 2002.

[27] C. McKnight, H. S. Levinson, C. Kamga and R. Paaswell, “Impact of traffic congestion on bus travel time in northern New Jersey,” *Transportation Research Record Journal,* vol. 1884, pp. 27-35, 2004.

[28] N. Draper and H. Smith, Applied Regression Analysis, New York: John Wiley and Sons, 1986.

[29] A. Albert, Regression and the Moor-Penrose Pseudoinverse, New York and London: Academic Press, 1972.

[30] A. Andronov, A. Kiselenko и E. Mostivenko, Forecasting of the development of Regional Transport System, Syktyvkar: KNZ UrO RAN (in Russian), 1991.

[31] W. Härdle, Applied Nonparametric Regression, Cambridge: Cambridge University Press, 2002.

[32] W. Härdle and M. Müller, “Multivariate and semiparametric kernel regression,” in *Smoothing and Regression,* M. Schimek, Ed., New York, Wiley, 2000, pp. 357-391.

[33] B. W. Silverman, Density Estimation for Statistics and Data Analysis, Chapman and Hall, 1986.

[34] D. W. Scott, Multivariate Density Estimation: Theory, Practice, and Visualization, Wiley, 1992.

[35] P. Hall, “Large Sample Optimality of Least Squares Cross-Validation in Density Estimation,” *The Annals of Statistics,* vol. 11, no. 4, pp. 1156-1174, 1983.

[36] W. Härdle, M. Müller, S. Sperlich and A. Werwatz, Nonparametric and Semiparametric Models, Berlin/Heidelberg: Springer, 2004.

[37] J. Gentle, W. Härdle and Y. Mori, Eds., Handbook of Computational Statistics: Concepts and Methods, Berlin/Heidelberg: Springer, 2004.

[38] "Frequency: Travelling with passenger cars," Statista, 2012. [Online]. Available: http://de.statista.com/statistik/daten/studie/176124/umfrage/haeufigkeit-pkw-fahren/.

[39] J. S. Racine, "Consistent significance testing for nonparametric regression," *Journal of Business and Economic Statistics*, vol. 15, pp. 369-379, 1997.

[40] H. Afanasyeva and A. Andronov, "On robustness of resampling estimators for linear regression models," *Communications in Dependability and Quality Management: An international Journal*, vol. 9, no. 1, pp. 5-11, 2006.