Learning to Rank from Concept-Drifting Network Data Streams

Lucrezia Macchia, Michelangelo Ceci, and Donato Malerba

Dipartimento di Informatica, Università degli Studi di Bari "Aldo Moro", via Orabona, 4 - 70126 Bari, Italy
lucrezia.macchia@uniba.it, {ceci,malerba}@di.uniba.it

Abstract. Networked data are, nowadays, collected in various application domains such as social networks, biological networks, sensor networks, spatial networks, peer-to-peer networks etc. Recently, the application of data stream mining to networked data, in order to study their evolution over time, is receiving increasing attention in the research community. Following this main stream of research, we propose an algorithm for mining ranking models from networked data which may evolve over time. In order to properly deal with the concept drift problem, the algorithm exploits an ensemble learning approach which allows us to weight the importance of learned ranking models from past data when ranking new data. Learned models are able to take the network autocorrelation into account, that is, the statistical dependency between the values of the same attribute on related nodes. Empirical results prove the effectiveness of the proposed algorithm and show that it performs better than other approaches proposed in the literature.

1 Introduction

The coming of new technologies has determined the generation, at a rapid rate, of massive structured and complex data that in most of cases can be represented as data networks. The networks have become ubiquitous in several social, economical and scientific fields ranging from the Internet to social sciences, biology, epidemiology, geography, finance and many others. Indeed, researchers in these fields have proven that systems of different nature can be represented as networks [19]. For instance, the Web can be considered as a network of webpages, which may be connected with each other by edges representing various explicit relations, such as hyperlinks. Sensor networks are networks where nodes represent sensors and edges represent the (spatial) distance between two sensors.

In the real world, network data may evolve over time. This evolution can be both in the structure of the network (nodes can be added or removed, edges can be added or removed) and in the distribution of the attribute values associated with the nodes. As an example, consider a sensor network whose nodes collect temperature, humidity, etc. at single positions in a specific environment. In this case, new sensors can be either added to the network or removed from it as well as the underlying data distribution of some variables may change. Moreover, as observed by Swanson [24], in this situation, data can be affected by temporal autocorrelation according to which two values of the some variable are cross correlated over a certain time lag.

Another important aspect that we have to consider when mining networked data is that they are characterized by a particular form of autocorrelation [12] according to which a value observed at a node depends on the values observed at neighboring nodes in the network [22]. The major difficulty due to the autocorrelation is that the independence assumption (i.i.d.), which typically underlies machine learning and data mining methods, is no longer valid. The violation of the instance independence has been identified as the main responsible of poor performance of traditional machine learning methods [18]. To remedy the negative effects of the violation of independence assumptions, autocorrelation has to be explicitly accommodated in the learned models.

By taking into account these two aspects, in this paper we face the problem of mining ranking functions. Although in the recent years learning ranking functions has received increasing attention due to its potential application to problems raised in information retrieval, machine learning, data mining and recommendation systems [10], at the best of our knowledge, very few works in the literature face the problem of learning to rank from network data which may evolve over time.

At this aim, we propose an ensemble learning approach in order to weight the importance of ranking models which are learned from past data. This weighting schema is then used for ranking new data. In this way, individual models can be learned from different time windows and can work as a team to enhance a final model. By appropriately defining the weighting schema, it is possible to give more importance to models learned from recent time windows than to models learned from distant time windows. As in [11][3][13] the ranking problem is boiled down into a regression problem. Individual models are tree-structured and allow us to consider network autocorrelation in the data. This is performed by considering a locally weighted regression function to be associated with the nodes of the tree, where the local weighting explicitly considers the network proximity of single elements of the network.

The paper is organized as follows. The next section and Section 3 report relevant related work. Section 4 describes the proposed algorithm. Section 5 describes the datasets, experimental setup and reports relevant results. Finally, in Section 6, some conclusions are drawn and some future work are outlined.

2 Related Works

In the following, we report some related works which *i)* mine ranking models able to work on network data and *ii)* exploit ensemble learning techniques for data stream mining.

2.1 Mining Ranking Models Able to Work on Network Data

The task considered in this work is that of learning to rank. This task has received increasing attention due to its potential application to problems in information retrieval and recommendation systems [1,5,10]. The aim of the methods developed in this field is to learn a ranking model which returns the output predictions in the form of a ranking of the examples given in input. According to [4], it is possible distinguish three types of ranking problems. The first type is *Label ranking*, where the goal is to learn a "label ranker" in the form of an $X \rightarrow S_Y$ mapping, where the output space S_Y is given by the set of all total orders (permutations) of the set of labels Y. The second type is *Instance ranking*, where an instance $x \in X$ belongs to one among a finite set of classes $Y = y_1, y_2, \ldots, y_k$ for which a natural order $y_1 < y_2 < \ldots < y_k$ is defined. The third type is *Object ranking*. In this case, the goal is to learn a ranking function $f(\cdot)$ which, given a subset of an underlying referential set of objects as an input, produces a ranking of these objects.

By focusing on object ranking, studies reported in the literature solve this problem by resorting to two alternative approaches. The first approach, determines a regression function that assigns a numerical value to each element of a set, then the same is used to sort the items. The second approach aims at learning preference functions, which are able to perform pairwise comparisons in order to define a relative order between two objects. The first approach is generally more efficient but it is applicable only when a single total order between objects is acceptable. While, when not all the objects have to necessarily be included in the ranking, the second approach is preferable. Since in this work we do not consider the problem of defining partial orders, we consider the first approach.

According to this approach, Herbrich et al. [11] propose to learn a function which, given an object description, returns an item belonging to an ordered set. The function is determined so that a loss function is minimized. A similar approach was proposed by Crammer et al. [3], in which the learned functions are modelled by perceptrons. Tesauro [25] proposed a symmetric neural network architecture that can be trained with representations of two states and a training signal that indicates which of the two states is preferable. In the framework of *constraint classification*, some works [8,9] exploit linear utility functions to find a way to express a constraint in the form $f_i(x) - f_j(x) > 0$, in order to transform the original ranking problem into a single binary classification problem.

By keeping in mind that we intend to learn ranking functions from network data, we propose to use regression trees which are proved to be easily adapted to work with this kind of data [22]. Regression trees [2] are supposed to be more comprehensible then classical regression models. They are built top-down by recursively partitioning the sample space. An attribute may be of varying importance for different regions of the sample space. A constant is associated with each leaf of a regression tree, so that the prediction performed by a regression tree is the same for all sample data falling in the same leaf. A generalisation of regression trees is represented by model trees, which associate multiple linear models with each leaf. Hence, different values can be predicted for sample data falling in the same leaf. Some of the model tree induction systems are RETIS [14], M5' [29], HTL [26], TSIR [15] and SMOTI [17].

One of the main peculiarities that led us to consider model trees is that their tree structure allows us to deal with the so-called "ecological fallacy" problem [21] according to which individual sub-regions do not have the same data distribution of the entire region. This is coherent with the research reported in [28], where the authors argue that the concept drift is not uniform over the feature space. Moreover, the tree structure allows us to capture different effects of the autocorrelation either at a global (higher levels of the tree) or at a local (lower levels of the tree) granularity level.

In this work we consider the system SMOTI, which is characterized by a tree structure with two types of nodes: regression nodes, which perform only straight-line regression, and splitting nodes, which partition the feature space. The multiple linear model associated with each leaf is then the composition of the straight-line regressions reported along the path from the root to the leaf. A brief description of SMOTI is reported in Section 3.

2.2 Ensemble Learning for Data Streams

The idea of *ensemble learning* is to employ multiple learners and combine their predictions. As observed in several works found in the literature, (see, for example [20]), the resulting ensemble is generally more accurate than any of the individual classifiers that contribute to the ensemble.

In this work we intend to exploit the idea of the ensemble to deal with data whose underlying distribution may evolve over time. Indeed, this idea is not novel and several papers in the literature exploit ensemble learning techniques in order to work with streaming data. For instance, in [23], the authors propose an ensemble learning of individual decision trees learned at different time windows. Decision trees are built through the classical Quinlan's C4.5 learning algorithm and ensemble aims at preserving only the k most accurate trees for future data. New classifies are added to the ensemble if the ensemble size does not exceed k, otherwise, new classifiers are added only if they improve the ensemble performance. Obviously, this leads to sacrifice exiting classifiers. Wang et al. [27] propose weighted classifier ensembles to mine streaming data affected by the concept drift phenomenon. They train an ensemble of classifiers from sequential data chunks in the stream. Subsequently, they associate each classifier with a weight which represents the expected prediction accuracy of the classifier on the current test examples. In [28] the authors propose to capture the non-deterministic nature of concept drifts in a region of the feature space and model the concept drift process as a continuous time Markov chain. In particular, they propose to train k classifiers from data in recent time windows (they use a decay factor) and assume that each classifier partitions the feature space into a set of non-overlapping regions (as decision trees do). When a new example arrives, on the basis of the region it belongs to, a score that represents the drift is computed (on the basis of the continuous time Markov chain). This score is then used to compute the probability that the example belongs to a class.

Although all cited works are proved to be effective in their capability to use models learned in the past for new data, none of them consider the problem of mining ranking models from networked data, which is the main objective of the present work. The only work that, at the best of our knowledge, faces the same problem we consider in this works, is reported in [16], where the authors modify the SVMRank algorithm in order to emphasize the importance of models learned in time periods during which data follow a data distribution that is similar to that observed in the time period for which prediction has to be made. However, their approach (called SVMRankT) does not consider the “ecological fallacy” problem (see section 2.1).

3 Background: Model Tree Induction in SMOTI

SMOTI (Stepwise Model Tree Induction) performs the top-down induction of model trees by considering not only a partitioning procedure, but also by some intermediate prediction functions [17]. This means that there are two types of nodes in the tree: regression nodes and splitting nodes. The former compute straight-line regressions, while the latter partition the sample space. They pass down training data to their children in two different ways. For a splitting node t, only a subgroup of the $N(t)$ training data in t is passed to each child, with no change on training cases. For a regression node t, all the data are passed down to its only child, but the values of both the dependent and independent numeric variables not included in the multiple linear model associated with t are transformed in order to remove the linear effect of those variables already included. Thus, descendants of a regression node will operate on a modified training set. Indeed, according to the statistical theory of linear regression [6], the incremental construction of a multiple linear model is made by removing the linear effect of introduced variables each time a new independent variable is added to the model. For instance, let us consider the problem of building a multiple regression model with two independent variables through a sequence of straight-line regressions:

$$\hat{Y} = a + bX_1 + cX_2 \tag{1}$$

We start regressing Y on X_1, so that the model $\hat{Y} = a_1 + b_1X_1$ is built. This fitted equation does not predict Y exactly. By adding the new variable X_2, the prediction might improve. Instead of starting from scratch and building a model with both X_1 and X_2, we can build a linear model for X_2 given

$$X_1 : \hat{X_2} = a_2 + b_2X_1 \tag{2}$$

Then we compute the residuals on $X_2 : X_2' = X_2 - (a_2 + b_2X_1)$ and on $Y : Y' = Y - (a_1 + b_1X_1)$. Finally, we regress Y' on X_2' alone:

$$\hat{Y'} = a_3 + b_3X_2' \tag{3}$$

By substituting the equations of X_2' and Y' in the last equation we have:

$$\big(Y - (\widehat{a_1 + b_1X_1})\big) = a_3 + b_3(X_2 - (a_2 + b_2X_1)) \tag{4}$$

Since $Y - (\widehat{a_1 + b_1}X_1) = \hat{Y} - (a_1 + b_1X_1)$, we have:

$$\hat{Y} = (a_3 + a_1a_2b_3) + (b_1 - b_2b_3)X_1 + b_3X_2 \tag{5}$$

It can be proven that this last model coincides with (1), that is, $a = a_3 + a_1a_2b_3$, $b = b_1 - b_2b_3$ and $c = b_3$. Therefore, when the first regression line of Y on X_1 is built we pass down both the residuals of Y and the residuals of the regression of X_2 on X_1. This means that we remove the linear effect of the variables already included in the model (X_1) from both the response variable (Y) and those variables to be selected for the next regression step (X_2).

4 Considering Network Autocorrelation in Model Tree Induction and Ranking Models' Ensemble

The original SMOTI algorithm uses the least squares method in order to identify the parameters to be used in all the linear regression functions such as (2) and (3). The least squares method works as follows: Let $\hat{Y} = \alpha + \beta X_j$ $(j = 1, \ldots, m)$ be a generic regression function to be built between variables X_j and Y,

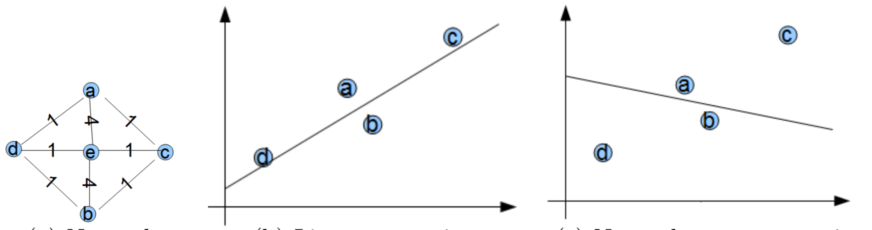
$$\begin{cases} \beta = (\mathbf{X}_j^T\mathbf{X}_j)^{-1}\mathbf{X}_j^T\mathbf{Y} = \dfrac{\sum\limits_{i=1,..,n} (x_{ji} - \overline{x_j})(y_i - \overline{y})}{\sum\limits_{i=1,..,n} (x_{ji} - \overline{x_j})^2} \\ \alpha = \overline{y} - \beta\overline{x_j} \end{cases} \tag{6}$$

where $\mathbf{X}_j = [x_{j1}, x_{j2}, \ldots, x_{jn}]$ and $\mathbf{Y} = [y_1, y_2, \ldots, y_n]$ are vectors of values of X_j and Y, respectively, and $\overline{y}$ and $\overline{x_j}$ are the averages of values in $\mathbf{X}_j$ and $\mathbf{Y}$, respectively.

However, although standard, this method neglects possible autocorrelation in the network data. In other words, the least squares method would lead to regression models (and so, in our case, piecewise ranking functions) which do not consider the network. In fact, in networks, data are organized according to a graph structure $G = (V, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is a set of vertices $v_i = (\mathbf{x}_i, y_i)$, $i = 1, \ldots, n$, $E = \{\langle v_i, v_h, w_{i,v}\rangle | i \neq h\}$ is a set of edges and $w_{i,h}$ represents the strength of the connection among the nodes. As stated before, neglecting the network structure (i.e. autocorrelation), may result in inaccurate prediction models [18].

To better explain how networks can be exploited in order to extract a ranking model that takes autocorrelation into account (i.e. models able to generate ranking labels which are more coherent in the network), we report a simple example.

Example 1. Let Figure 1(a) be a network structure, let $\mathbf{X}_j = [1, 5, 4, 8]$ and $\mathbf{Y} = [2, 3, 4, 6]$ be the values for the nodes d, b, a, c and let e be the example for which a ranking label has to be identified (query point). According to the standard least squares method, the (ranking) regression line is the regression



(a) Network (b) Linear regression (c) Network aware regression

Fig. 1. A simple network (a) and two (ranking) regression functions. In (c), the regression line gives more importance to vertices a and b since the query point is e.

line reported in Figure 1(b). However, by taking into account the network, the regression line should change (see Figure 1(c)) by giving more importance to the effect of the nodes a and b, which are strongly correlated with e.

To take autocorrelation into account, the alternative solution we intend to exploit in this work is that of locally weighted regression method (LWR). The basic idea in LWR, is that a local model should fit to nearby data, according to a proximity measure defined on the feature space $X_1, \ldots, X_m$. Differently, in this work we exploit the network in order to account the degree of correlation between pairs of vertices.

Formally, let $a_i \in V$ be a vertex, it is possible to build a diagonal matrix

$$\mathbf{W_i} = \begin{pmatrix} w_{i1} & 0 & \ldots \\ 0 & w_{i2} & \ldots \\ \vdots & \vdots & w_{in} \end{pmatrix}$$

where each non-zero element w_{iv}, $v = 1, \ldots, n$ represents the strength of the connection between vertices a_i and a_v and n represents the number of examples falling in the node of the tree. Then, it is possible to build a regression function between Y and X_j associated with each example falling in the node.

$$Y = \alpha_i + \beta_i X_j \tag{7}$$

in this case, $\beta_i = ((\mathbf{W}_i\mathbf{X}_j)^T(\mathbf{W}_i\mathbf{X}_j))^{-1}(\mathbf{W}_i\mathbf{X}_j)^T(\mathbf{W}_i\mathbf{Y})$. This means that

$$\begin{cases} \beta_i = \dfrac{\sum\limits_{t=1,..,n} (w_{it}x_{jt} - r)(w_{it}y_t - s)}{\sum\limits_{t=1,..,n} (w_{it}x_{jt} - r)^2} \\ \alpha_i = s - \beta_i r \end{cases} \tag{8}$$

where $r = \frac{1}{n}\sum_t w_{it}x_{jt}$ and $s = \frac{1}{n}\sum_t w_{it}y_t$.

In the regression step, in order to uniformly modify all the examples that are passed down in the model tree construction, we use the averages of the parameters' values. More, formally, suppose we are introducing a regression node on X_1, then residuals are computed as follows: $X'_2 = X_2 - (\overline{\alpha_2} + \overline{\beta_2} X_1)$ and $Y' = Y - (\overline{\alpha_1} + \overline{\beta_1} X_1)$ where $\overline{\alpha} = \frac{1}{n}\sum_i \alpha_i$ and $\overline{\beta} = \frac{1}{n}\sum_i \beta_i$ for each regression problem (on X_1 and on Y).

It is noteworthy that this approximation does not compromise the correct consideration of the autocorrelation since residuals will be explained at lower levels of the model tree. Moreover, autocorrelation is mainly considered during the prediction phase. In fact, during prediction, once the example to be classified reaches the leaf, the system uses the α_i and β_i parameters associated with the training example for which the connection is the strongest (according to the network structure). In case more than one training example satisfies this property, the average of the parameters' values is considered. This approach allows the system to predict values for network nodes that are not available during the learning phase.

It is worth to notice that this approach also solves problems coming from the collinearity phenomenon, that is, when some of the independent variables are related to each other. Indeed, as recognized in [7], when some variables are (approximately) collinear, several problems may occur, such as indeterminacy of regression coefficients, unreliability of the estimates of the regression coefficients, and impossibility of evaluating the relative importance of the independent variables. In our approach, this problem is solved by introducing one variable at a time in the model tree.

By introducing the temporal dimension in the analysis, we resort to a temporal sliding window framework. Let $\langle G_1, \ldots G_{t-1}\rangle$ be a sequence of time windows associated with $t-1$ time points, and $\tau : \mathcal{T} \times V \rightarrow \mathbb{R}$ be a ranking labeling function which maps a model tree $T_q \in \mathcal{T}$ and a vertex $v \in V$ in the space of ranking labels ($\mathbb{R}$), then it is possible to define a general framework able to classify a generic vertex v at time t according to all the ranking models $T_1, T_2, \ldots, T_{t-1}$ learned from $\langle G_1, \ldots G_{t-1}\rangle$:

$$\tau'(T_1, T_2, \ldots, T_{t-1}, v) = \frac{1}{t * (t-1)/2} \sum_{q=1,\ldots,t-1} \frac{q}{t} * \tau(T_q, v) \tag{9}$$

where the first factor is only used for normalization purposes and $\frac{q}{t}$ is a decay factor which gives more importance to models learned in recent time windows than to models learned in distant time windows.

It is noteworthy that the method well adapts to cases in which the network structure changes over time. In fact, it is not strictly necessary that the vertex v belongs to the network in the time windows $\{G_1, \ldots G_{t-1}\}$. Moreover, our choice to ignore network properties in the splitting nodes of the model tree allows us to use the ranking model in different (but related) networks.

5 Experiments

In order to evaluate the effectiveness of the proposed solution, we performed experiments on three real world datasets, that is, Intel Lab database, Portuguese rivers database and California Truck database. The proposed approach has been compared with results obtained with SVMRankT [16].

As a measure to evaluate the learned ranking models, we use the Spearman's rank correlation coefficient. Let $V^{(t)} = \{v_1^{(t)}, v_2^{(t)}, \ldots, v_n^{(t)}\}$ be the real dataset at time t, $\hat{y}_i^{(t)} = \tau'(T_1, T_2, \ldots, T_{t-1}, v_i)$ be the estimated ranking for the example v_i and $y_i^{(t)}$ be the real ranking of v_i at time t, the Spearman's rank correlation coefficient is defined as:

$$\rho = \frac{\sum\limits_{i=1,\ldots,n_t} \left(y_i^{(t)} - \overline{y^{(t)}}\right)\left(\hat{y}_i^{(t)} - \overline{\hat{y}^{(t)}}\right)}{\sqrt{\sum\limits_{i=1,\ldots,n_t} \left(y_i^{(t)} - \overline{y^{(t)}}\right)^2 \left(\hat{y}_i^{(t)} - \overline{\hat{y}^{(t)}}\right)^2}} \tag{10}$$

where $\overline{y^{(t)}}$ and $\overline{\hat{y}^{(t)}}$ are the average true and estimated rankings, respectively. ρ ranges in the interval [-1,1], where -1 means negative correlation in the ranking and 1 means perfect ranking.

The first dataset considered in the experiments is the Intel Lab Database which contains real information collected from 54 sensors deployed in the Intel Berkeley Research lab between February 28th and and March 21st, 2004. The dataset is taken from http://db.csail.mit.edu/labdata/labdata.html. The sensors which we consider in this experiment have collected timestamped temperature, humidity and luminosity values once every 31 seconds. Networks are built by considering the spatial distance between sensors and the target attribute is represented by the temperature (we removed temperature and we used the temperature ranking as y value). In the experiments, we only considered working days and we used 1-day time-intervals, this means that we built 15 networks in all.

In Table 1, results of the Spearman's rank correlation coefficient are reported. In this dataset, the proposed approach is not able to show the same performances as those shown by SVMRankT. This is mainly due to the small number of attributes that prevents the model trees learning approach to learn accurate models. This is different from what happens in the case of SVMRankT that is based on SVMs that are able to create oblique partitions of the feature space. Moreover, in this dataset, autocorrelation phenomenon is quite uniform across the space (ecological fallacy does not hold).

The Portuguese rivers dataset holds water's information of the rivers Douro and Paiva. The dataset may be incomplete because the controls are manually done and are not done systematically. The original dataset is composed of a fact table and six additional relational tables: The fact table (ANALYSIS) contains information on the measures under control (pH, % Coliformi Bacteria, conductivity, turbidity, % Escherichia Coli Bacteria) and the gathering method. Additional tables are directly (or indirectly) connected to ANALYSIS according to a snowflake logic schema. They are: PARAMETERS (that are considered in the analysis), INSTITUTIONS (that collected data), DAY, CONTROL POINTS and PATH (that specifies the position of a control point according to the course of the rivers). From the table PATH we got the course of the river and the

Table 1. Intel lab Dataset: Spearman's rank coefficient

Train	**Test**	*SVMRankT*	***Our approach***
1 2	3	**0.9021739**	0.8312905
1 2 3	4	**0.9529370**	0.9062211
1 2 3 4	5	**0.9364014**	0.8566142
1 2 3 4 5	6	**0.9240286**	0.8507169
1 2 3 4 5 6	7	**0.9084181**	0.8576550
1 2 3 4 5 6 7	8	0.5514569	**0.5795560**
1 2 3 4 5 6 7 8	9	**0.8632053**	0.8554579
1 2 3 4 5 6 7 8 9	10	**0.8931544**	0.8331406
1 2 3 4 5 6 7 8 9 10	11	**0.8375346**	0.8252775
1 2 3 4 5 6 7 8 9 10 11	12	**0.8341813**	0.5730804
1 2 3 4 5 6 7 8 9 10 11 12	13	**0.7514743**	0.7318166
1 2 3 4 5 6 7 8 9 10 11 12 13	14	**0.8169229**	0.4641246
1 2 3 4 5 6 7 8 9 10 11 12 13 14	15	**0.9021739**	0.5225197
Avg.		**0.8518510**	0.7451901

Table 2. Portuguese rivers dataset: Spearman's rank coefficient

Train	**Test**	*SVMRankT*	***Our approach***
2004-2005	2006	0.4024926686	**0.6146444282**
2004-2005-2006	2007	0.4761730205	**0.6251832845**
2004-2005-2006-2007	2008	0.4769061584	**0.6917155425**
2004-2005-2006-2007-2008	2009	0.4226539589	**0.6462609971**
Avg.		0.44455645161	**0.64445106305**

position of the control points in order to build the network structure. The weights on the edges represent the navigation distance between the control points (in all we have 115 control points). We considered data aggregated by year and for each node, we represented institution, gathering method, pH, % Califormi Bacteria, conductivity, turbidity, % Escherichia Coli Bacteria. Aggregation is performed by considering mode (average) for discrete (continuous) values. In all, we considered 6 years (from 2004 to 2009). The experiments are performed using the pH feature as target since it is recognized to be a good indicator of river pollution.

Results of the Spearman's rank correlation coefficient are reported in Table 2. Differently from conclusions drawn in the case of Intel lab dataset, in this case our approach outperforms SVMRankT approach of a great margin. This confirms our intuitions since in this dataset, autocorrelation phenomenon is not uniform across the space.

The California traffic dataset concerns the traffic on the highways of California. The dataset is taken from http://traffic-counts.dot.ca.gov/index.htm. The experiments are carried out using the following independent attributes observed by sensors on highways: the percentage of trucks, the percentage of 2-axle vehicles, the percentage of 3-axle vehicles, the percentage of 4-axle vehicles, the percentage of 5-axle vehicles. The goal is to rank sensors' positions on the basis of the sum of the volumes of traffic on a road in both directions. Each sensor

Table 3. California traffic dataset: Spearman's rank coefficient

Train	Test	*SVMRankT*	Our approach
2001-2002	2003	0.7484557	**0.8064138**
2001-2002-2003	2004	0.7417657	**0.8064138**
2001-2002-2003-2004	2005	0.7402272	**0.7970344**
2001-2002-2003-2004-2005	2006	0.7468870	**0.8122890**
2001-2002-2003-2004-2005-2006	2007	0.7447633	**0.8125018**
2001-2002-2003-2004-2005-2006-2007	2008	0.7500496	**0.8247747**
2001-2002-2003-2004-2005-2006-2007-2008	2009	0.7375568	**0.7725733**
Avg.		0.7443340	**0.8045715**

represents a node in the network (in all, we have 969 sensors), while weights on the edges represent the driving distance between two sensors. However, the network is not fully connected and only nodes whose driving distance is less than 25 miles are connected (in all, there are 34093 edges). The dataset refers to the period 2001-2009 and for each year, a network is created.

Results of the Spearman's rank correlation coefficient confirm results obtained on the Portuguese rivers dataset (see Table 3). This result is quite interesting since, in this case, the network edges are quite sparse. This means that the LWR model is able to concentrate the attention only on truly interesting vertices.

6 Conclusions

In this paper we have faced the problem of mining ranking models from networked data whose data distribution may change over time. The first contribution of this paper is to provide a way to learn ranking models based on model trees which are able to consider (different effects of) the autocorrelation phenomenon. This is obtained by resorting to a modified version of the locally weighted regression method. The second contribution of this paper is to provide a time windows framework able to combine models learned in the past for prediction in future time windows.

We evaluate our approach on several real world problems of learning to rank from network data, coming from the area of sensor networks. An empirical comparison with an SVM-based approach shows the superiority of our approach when working with datasets where the effect of the autocorrelation phenomenon is not uniform in the network and with not fully connected networks.

For future works, we intend compare our approach with additional existing regressions/ranking approaches (after adaptation). Moreover, we intend to modify our approach in order to allow it to distinguish between smooth and abrupt changes. Finally, we intend to apply our approach in the context of biological (literature) data analysis for gene prioritization, that is, identification of the most relevant genes that are "connected" to a given disease (e.g. regulate the disease).

References

1. Aiolli, F.: A preference model for structured supervised learning tasks. In: ICDM, pp. 557–560. IEEE Computer Society (2005)
2. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Statistics/Probability Series. Wadsworth Publishing Company, Belmont (1984)
3. Crammer, K., Singer, Y.: Pranking with ranking. In: NIPS, pp. 641–647. MIT Press (2001)
4. Dembczyski, K., Kotlowski, W., Slowiski, R., Szelag, M.: Learning of rule ensembles for multiple attribute ranking problems. In: Fürnkranz, J., Hüllermeier, E. (eds.) Preference Learning, pp. 217–247. Springer (2010)
5. Doyle, J.: Prospects for preferences. Computational Intelligence 20(2), 111–136 (2004)
6. Draper, N.R., Smith, H.: Applied regression analysis. Wiley series in probability and mathematical statistics. Wiley, New York (1996)
7. Draper, N.R., Smith, H.: Applied regression analysis. John Wiley & Sons (1982)
8. Har-Peled, S., Roth, D., Zimak, D.: Constraint Classification: A New Approach to Multiclass Classification. In: Cesa-Bianchi, N., Numao, M., Reischuk, R. (eds.) ALT 2002. LNCS (LNAI), vol. 2533, pp. 365–379. Springer, Heidelberg (2002)
9. Har-Peled, S., Roth, D., Zimak, D.: Constraint classification for multiclass classification and ranking. In: Becker, S., Thrun, S., Obermayer, K. (eds.) Advances in Neural Information Processing Systems 15 (NIPS 2002), pp. 785–792 (2003)
10. Herbrich, R., Graepel, T., Bollmann-sdorra, P., Obermayer, K.: Learning preference relations for information retrieval (1998)
11. Herbrich, R., Graepel, T., Obermayer, K.: Large margin rank boundaries for ordinal regression. MIT Press (2000)
12. Jensen, D., Neville, J.: Linkage and autocorrelation cause feature selection bias in relational learning. In: Proc. 9th Intl. Conf. on Machine Learning, pp. 259–266. Morgan Kaufmann (2002)
13. Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2002, pp. 133–142. ACM, New York (2002)
14. Karalic, A.: Linear regression in regression tree leaves. In: Proceedings of ECAI 1992, pp. 440–441. John Wiley & Sons (1992)
15. Lubinsky, D.: Tree structured interpretable regression. In: Fisher, D., Lenz, H.J. (eds.) Learning from Data. Lecture Notes in Statistics. Springer (1994)
16. Macchia, L., Ceci, M., Malerba, D.: Mining Ranking Models from Dynamic Network Data. In: Perner, P. (ed.) MLDM 2012. LNCS, vol. 7376, pp. 566–577. Springer, Heidelberg (2012)
17. Malerba, D., Esposito, F., Ceci, M., Appice, A.: Top-down induction of model trees with regression and splitting nodes. IEEE Trans. Pattern Anal. Mach. Intell. 26(5), 612–625 (2004)
18. Neville, J., Simsek, O., Jensen, D.: Autocorrelation and relational learning: Challenges and opportunities. In: Wshp. Statistical Relational Learning (2004)
19. Newman, M.E.J., Watts, D.J.: The structure and dynamics of networks. Princeton University Press (2006)
20. Opitz, D., Maclin, R.: Popular ensemble methods: An empirical study. Journal of Artificial Intelligence Research 11, 169–198 (1999)

21. Robinson, W.S.: Ecological Correlations and the Behavior of Individuals. American Sociological Review 15(3), 351–357 (1950)
22. Stojanova, D., Ceci, M., Appice, A., Džeroski, S.: Network Regression with Predictive Clustering Trees. In: Gunopulos, D., Hofmann, T., Malerba, D., Vazirgiannis, M. (eds.) ECML PKDD 2011, Part III. LNCS, vol. 6913, pp. 333–348. Springer, Heidelberg (2011)
23. Street, W.N., Kim, Y.: A streaming ensemble algorithm (sea) for large-scale classification. In: Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2001, pp. 377–382. ACM, New York (2001)
24. Swanson, B.J.: Autocorrelated rates of change in animal populations and their relationship to precipitation. Conservation Biology 12(4), 801–808 (1998)
25. Tesauro, G.: Connectionist learning of expert preferences by comparison training. In: Advances in Neural Information Processing Systems 1, pp. 99–106. Morgan Kaufmann Publishers Inc., San Francisco (1989)
26. Torgo, L.: Functional models for regression tree leaves. In: Fisher, D.H. (ed.) ICML, pp. 385–393. Morgan Kaufmann (1997)
27. Wang, H., Fan, W., Yu, P.S., Han, J.: Mining concept-drifting data streams using ensemble classifiers. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2003, pp. 226–235. ACM, New York (2003)
28. Wang, H., Yin, J., Pei, J., Yu, P.S., Yu, J.X.: Suppressing model overfitting in mining concept-drifting data streams. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2006, pp. 736–741. ACM, New York (2006)
29. Wang, Y., Witten, I.H.: Induction of model trees for predicting continuous classes. In: Poster papers of the 9th European Conference on Machine Learning. Springer (1997)