Mining Ranking Models from Dynamic Network Data

Lucrezia Macchia, Michelangelo Ceci, and Donato Malerba

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

Abstract. In recent years, improvement in ubiquitous technologies and sensor networks have motivated the application of data mining techniques to network organized data. Network data describe entities represented by nodes, which may be connected with (related to) each other by edges. Many network datasets are characterized by a form of autocorrelation where the value of a variable at a given node depends on the values of variables at the nodes it is connected with. This phenomenon is a direct violation of the assumption that data are independently and identically distributed (i.i.d.). At the same time, it offers the unique opportunity to improve the performance of predictive models on network data, as inferences about one entity can be used to improve inferences about related entities. In this work, we propose a method for learning to rank from network data when data distribution may change over time. The learned models can be used to predict the ranking of nodes in the network for new time periods. The proposed method modifies the SVM-Rank 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 new time period. We evaluate our approach on several real world problems of learning to rank from network data, coming from the area of sensor networks.

1 Introduction

In recent years, learning preference functions has received increasing attention due to its potential application to problems raised in information retrieval, machine learning, data mining and recommendation systems [1], [6], [11].

As in many data mining tasks, when facing the problem of learning preference functions, new research frontiers in new application domains require capabilities of dealing with structured and complex data that in most of cases can be represented as data networks. In fact, 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 [18]. For instance, the Web can be considered as a network of web-pages, which may be connected with each other by edges representing various

© Springer-Verlag Berlin Heidelberg 2012

explicit relations, such as hyperlinks. Sensor networks are networks where nodes represent sensors and edges represent the (spatial) distance between two sensors.

This particular organization of data adds additional complexity to the task at hand since networked data are characterized by a particular form of autocorrelation [14] according to which a value observed at a node depends on the values observed at neighboring nodes in the network [20]. The major difficulty due to the autocorrelation is that the independence assumption (i.i.d.), which typically underlies machine learning 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 [17]. To remedy the negative effects of the violation of independence assumptions, autocorrelation has to be explicitly accommodated in the learned models.

Moreover, 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. Indeed, as observed by Swanson [21], 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.

In this paper, we argue that a method for learning preference functions from network data should take both network and temporal autocorrelation into account. At this aim, we propose two solutions, both based on the well known SVMRank algorithm [15]. The proposed solutions allow us to learn preference functions over a series of consecutive time intervals by taking network autocorrelation into account. The first solution uses a fading factor that allows the algorithm to give more importance to models learned in recent time periods than models learned in the past. The second solution allows the algorithm to give more importance to models associated to more correlated time intervals than models associated to less correlated time intervals. This means that, while models learned according to the first solution give more importance to the order in which the time intervals are considered, in the second solution, more importance is given to the similarity between data distributions observed in two distinct time intervals (periodicity, if present, can be captured).

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

2 Related Work

The task considered in this work is that of preference leaning functions. 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. It is possible distinguish three types of ranking problems [5]:

- *Label ranking*: the goal is to learn a "label ranker" in the form of an $X \rightarrow S_Y$ mapping, where the input space X is the feature space and the output space S_Y is given by the set of all total orders (permutations) of the set of labels Y.
- *Instance ranking*: an instance x ∈ 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.
- *Object ranking*: the goal is to learn a ranking function $f(\cdot)$ which, given a subset Z of an underlying referential set Z of objects as an input, produces a ranking of these objects.

In this work we consider the object ranking problem, where objects $x \in X$ are described in terms of an attribute-value representation, but can be linked each other on the basis of a network structure. As training information, an object ranker has access to exemplary rankings or pairwise preferences of the form $x_i > x_j$ suggesting that x_i should be ranked higher than x_j.

Studies reported in the literature solve this problem by resorting to two alternative solutions. The first solution determines a function that assigns a numerical value to each element of a set, then the same is used to sort the items. The second solution aims at learning preference functions, which permit to perform pairwise comparisons in order to define a relative order between two objects [8,13,3]. The first solution 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 solution is preferable. Since pairwise total orders can lead to define partial orders, in this work we consider the first solution.

Concerning this solution, Herbrich et al. [12] 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. [4], in which the learned functions are modeled by perceptrons. Tesauro [22] 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* [9,10], some authors 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.

However, most of the works presented in the literature neither consider the possible network structure according to which examples can be arranged nor consider the possible evolution of the network. Exceptions are represented by ranking algorithms used in information retrieval to rank web pages by taking hyperlinks into account (e.g. PageRank-like algorithms [19]). In this case, however, the possible evolution of the network is not taken into account. In addition, they do not consider autocorrelation properly since they do not consider the fact that the values observed at a node depend on the values observed at linked nodes in the network. Other exceptions are represented by approaches that resort to a multi-relational data mining framework which implicitly takes autocorrelation into account [2,16]. These works, however, resort to the second solution since they are able to obtain a relative order between two objects.

3 Learning Ranking Functions with Different Time Windows

Support vector machines (SVMs) are a set of related supervised learning methods used for classification and regression. The foundations of SVMs have been presented by Vapnik in [23] and are related to the computational learning theory.

In the classical classification case, the problem solved by SVMs can be formalized in the following way: given a set of positive and negative examples $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, where $x_i \in X \subseteq \mathbb{R}^m$ (x_i is a feature vector) and $y_i \in \{-1, +1\}$, an SVM identifies the hyperplane in $\mathbb{R}^m$ that linearly separates positive and negative examples with the maximum margin (optimal separating hyperplane). In the case of network data, the weights associated to the edges allow us to represent nodes as examples and their position in the feature space. In this way, the identified hyperplane takes into account the "position" of the examples in the feature space.

In this work, we exploit SVMs in order to rank examples instead of generating an optimal separating hyperplane. Indeed, this idea is not novel and in SVM-Rank[1] [15] an optimization problem that permits the definition of a ranking function is defined.

In detail, SVMRank algorithm resolves the following optimization problem:

Given a training set $(x_1, y_1), \ldots, (x_n, y_n)$ with $x_i \in \mathbb{R}^m$ and $y_i \in Y$, where x_i represents the example and y_i its ordinal value in the ranking, the problem is to find a ranking function $h : \mathbb{R}^m \rightarrow \mathbb{R}$ defined as $h(x) = w^T x$, such that the following optimization problem is solved:

$$\begin{aligned}
&argmin_{w,\xi \geq 0} : \tfrac{1}{2}\vec{w}^T \cdot \vec{w} + C\xi \\
&s.t. \forall (i,j) \in P, \forall c_{ij} \in \{0,1\} : \tfrac{1}{|P|} w^T \Sigma_{i=1}^{|P|} c_{ij}(x_i - x_j) \geq \tfrac{1}{|P|} \Sigma_{i=1}^{|P|} c_{ij} - \xi
\end{aligned}$$

where ξ is a slack variable, P is the set of pairs (i, j) for which example x_i has a higher rank than example x_j, i.e. $P = \{(i, j) | y_i > y_j\}$ and C is a positive regularization constant. The regularization constant in the cost function defines the trade-off between a large margin and misclassification error (i.e. empirical risk minimization). Intuitively, this formulation finds a large-margin linear function $h(x)$ that minimizes the number of pairs of training examples that are swapped w.r.t. their desired order.

However, this optimization, permits us to learn a ranking model for *static* training data. In our case, we can learn a different ranking model for each time

[1] Downloaded from `http://www.cs.cornell.edu/people/tj/svm_light/svm_rank.html`

interval. The models can then be combined in order to identify the final model to be used for the next time interval.

More formally, let

$$S' = \bigcup_{t=1}^{z} S_t \tag{1}$$

be the complete training dataset where $S_t = \{(x_{t,1}, y_{t,1}), \ldots, (x_{t,n_t}, y_{t,n_t})\}$ represents the training dataset for the t-th time-interval, z be the total number of time-intervals.

Then the following optimization problem is solved:

$$argmin_{w_t,\xi \geq 0} : \frac{1}{2} w_t^T \cdot w_t + C\xi \tag{2}$$

$$s.t. \forall (t = 1, \ldots, z), \forall (x_{t,i}, x_{t,j}) \in S_t, \forall c_{ij} \in \{0, 1\} :$$
$$\tfrac{1}{|P|} w_t^T \Sigma_{i=1}^{|P|} c_{ij}(x_{t,i} - x_{t,j}) + 1 - \varsigma(x_{t,i}, x_{t,j}, S', S_{z+1}) \geq \tfrac{1}{|P|} \Sigma_{i=1}^{|P|} c_{ij} - \xi$$

where S_{z+1} represents the nodes of the network at the $z+1$ time interval (on which prediction is performed) and $\varsigma(x_{t,i}, x_{t,j}, S', S_{z+1})$ modifies the constraint by considering both network autocorrelation and possible dependence with models observed at previous time windows.

In order to compute $\varsigma(x_{t,i}, x_{t,j}, S', S_{z+1})$, two variants are considered. In the first variant (called $SVMRank_R$), the function $\varsigma(x_{t,i}, x_{t,j}, S', S_{z+1})$ emphasizes recent data with respect to past data:

$$\varsigma(x_{t,i}, x_{t,j}, S', S_{z+1}) = \frac{t}{z} + dist(x_{t,i}, x_{t,j}) \tag{3}$$

where $dist(x_{t,i}, x_{t,j})$ is the distance between node i and node j and $\frac{t}{z}$ is a fading factor.

In the second variant (called $SVMRank_T$), the function $\varsigma(x_{t,i}, x_{t,j}, S', S_{z+1})$ emphasizes the models observed at previous time windows which are (supposed to be) more similar to the model at time $z+1$. This similarity is computed according to the Durbin-Watson statistic [7] $d(S', S_{z+1})$ defined on all the features and all the nodes[2]:

$$d(S', S_{z+1}) = \frac{1}{m} \sum_{l=1,\ldots,m} \left(\frac{1}{n} \sum_{i=1,\ldots,n} \frac{\Sigma_{t=2}^{z+1} (e_{t,i}^{(l)} - e_{t-1,i}^{(l)})^2}{\Sigma_{t=1}^{z} {e_{t,i}^{(l)}}^2} \right) \tag{4}$$

where $e_{t,i}^{(l)}$ is the value of the l-th feature of $x_{t,i}$. When there is high positive (negative) temporal autocorrelation, $d(S', S_{z+1})$ approaches to 0 (4), if there is no temporal autocorrelation, $d(S', S_{z+1})$ approaches to 2.

Once all w_t, $t = 1, \ldots, z$ are computed, the average vector $w = 1/z \sum_{t=1,\ldots,z} w_t$ is the ranking vector used in the prediction of the ranking for nodes at time

[2] Computation of $d(S', S_{z+1})$ only considers nodes that are present in all time windows in S'.

$z+1$. In particular, the orthogonal projection of nodes over w implicitly defines the ranking (see figure 1).

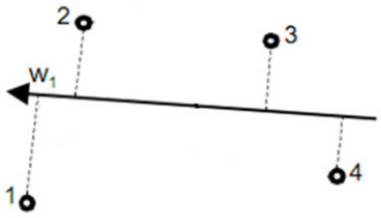

Fig. 1. A ranking function that perfectly ranks objects according to their label

In this variant, the function $\varsigma(x_{t,i}, x_{t,j}, S', S_{z+1})$ is defined as:

$$\varsigma(x_{t,i}, x_{t,j}, S', S_{z+1}) = d(S', S_{z+1}) + dist(x_{t,i}, x_{t,j}) \tag{5}$$

Intuitively, the first solution returns a final model that gives more importance to models learned in the recent time period than models learned in the past, while the second solution returns a final model that considers the relationship between values separated from each other by a given time lag, computed with the Durbin-Watson statistic.

The two variants are implemented by exploiting a modified version of the algorithm proposed in [15], where new constraints are used. Similarly to what proposed in [15], also in our case, the considered optimization problem in (2) can be solved in logarithmic time by means of the algorithm reported in Algorithm 1, where $c^+ = [c_i^+]_{i=1,\dots,n}$ and $c^- = [c_i^-]_{i=1,\dots,n}$ are vectors whose values are defined as follows:

$$c_i^+ = |\{j|(x_j, y_j) \in S_t \wedge y_i > y_j \wedge ((w^T x_i) - (w^T x_j) < 1 + 1 - \varsigma(x_{t,i}, x_{t,j}, S', S_{z+1}))\}|$$
$$c_i^- = |\{j|(x_j, y_j) \in S_t \wedge y_j > y_i \wedge ((w^T x_j) - (w^T x_i) < 1 + 1 - \varsigma(x_{t,i}, x_{t,j}, S', S_{z+1}))\}|$$

Intuitively, these are constraints over false positive and false negative errors that take constraints in (2) into account. Coherently, constraints in (2) are also taken into account in the definition of the stopping criterion (line 24).

4 Experiments

In order to evaluate the effectiveness of the proposed solution, we performed experiments on three real world datasets, that is, Intel Lab Database, California Truck and Portuguese rivers database. In all the experiments, we set C=20 (after preliminary experiments aiming at identify the best C value among the values in the set {10, 15, 20, 30}). In the experiments, in turn, we used the last time interval as testing set.

Algorithm 1. Training Ord. Regr. SVMs

```
Input: $S = ((x_1, y_1), ..., (x_n, y_n)), C$
 1: $W \leftarrow \oslash$
 2: repeat
 3:   $(w, \xi) \leftarrow argmin_{w,\xi>0} \frac{1}{2}\vec{w}^T \cdot \vec{w} + C\xi$
 4:   $s.t.\forall(c^+, c^-) \in W : \frac{1}{|P|} w^T \Sigma_{i=1}^n (c_i^+ - c_i^-) x_i \geq \frac{1}{2|P|} \Sigma_{i=1}^n (c_i^+ - c_i^-) - \xi$
 5:   sort $S$ by decreasing $w^T x_i$
 6:   $c^+ \leftarrow 0; c^- \leftarrow 0$
 7:   $n_r \leftarrow$ number of examples with $y_i = r$
 8:   for $r = 2$ to $R$ do
 9:     $i \leftarrow 1; j \leftarrow 1; a \leftarrow 0; b \leftarrow 0$
10:     while $1 \leq n$ do
11:       if $y_i = r$ then
12:         while $(j \leq n) \bigwedge (w^T x_i - w^T x_j < 1)$ do
13:           if $y_i < r$ then
14:             $b + +; c_j^- \leftarrow c_j^- + (n_r a + 1)$
15:           end if
16:           $j + +$
17:         end while
18:         $a + +; c_i^+ \leftarrow c_i^+ + b$
19:       end if
20:       $i + +$
21:     end while
22:   end for
23:   $W \leftarrow W \bigcup \{(c^+, c^-)\}$
24: until $\frac{1}{2|P|} \Sigma_{i=1}^n (c_i^+ - c_i^-) - \frac{1}{|P|} \Sigma_{i=1}^n (c_i^+ - c_i^-)(w^T x_i) + 1 - \varsigma(x_{t,i}, x_{t,j}, S', S_{z+1})) \geq \xi + \epsilon$
```

In order to evaluate the learned ranking models, we used the Spearman's rank correlation coefficient.

Spearman's rank correlation coefficient is the non-parametric alternative to correlation and can be used when the data do not meet, as in this case, the assumptions about normality, homoscedasticity and linearity. Let $S_{z+1} = \{(x_{z+1,1}, y_{z+1,1}), \ldots, (x_{z+1,n_{z+1}}, y_{z+1,n_{z+1}})\}$ be the real dataset at time $z + 1$ and $y'_{z+1,i} = h(x_{z+1,i})$ be the estimated ranking for the example $x_{z+1,i}$, the Spearman's rank correlation coefficient is defined as:

$$\rho = \frac{\sum_{i=1,\ldots,n_{z+1}} (y_{z+1,i} - \overline{y_{z+1}})(y'_{z+1,i} - \overline{y'_{z+1}})}{\sqrt{\sum_{i=1,\ldots,n_{z+1}} (y_{z+1,i} - \overline{y_{z+1}})^2 (y'_{z+1,i} - \overline{y'_{z+1}})^2}} \tag{6}$$

where $\overline{y_{z+1}}$ ($\overline{y'_{z+1}}$) is the average ranking. ρ ranges in the interval [-1,1], where -1 means negative correlation in the ranking and 1 means perfect ranking.

Intel Lab Database contains real information collected from 54 sensors deployed in the Intel Berkeley Research lab between February 28th and and March 21st, 2004. 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

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

Train	**Test**	**SVMRank**	$SVMRank_R$	$SVMRank_T$
1 2	3	0.8743061	0.8743061	**0.9021739**
1 2 3	4	0.9531683	**0.9532839**	0.9529370
1 2 3 4	5	0.9237974	**0.9382516**	0.9364014
1 2 3 4 5	6	0.9153561	0.9210222	**0.9240286**
1 2 3 4 5 6	7	0.9084181	**0.9087650**	0.9084181
1 2 3 4 5 6 7	8	**0.5570074**	0.5498381	0.5514569
1 2 3 4 5 6 7 8	9	0.8592738	0.8591581	**0.8632053**
1 2 3 4 5 6 7 8 9	10	**0.8953515**	0.8943108	0.8931544
1 2 3 4 5 6 7 8 9 10	11	**0.8441258**	0.8236586	0.8375346
1 2 3 4 5 6 7 8 9 10 11	12	0.8316373	0.8156799	**0.8341813**
1 2 3 4 5 6 7 8 9 10 11 12	13	**0.7602624**	0.7513586	0.7514743
1 2 3 4 5 6 7 8 9 10 11 12 13	14	0.7264974	0.7209470	**0.8169229**
1 2 3 4 5 6 7 8 9 10 11 12 13 14	15	0.8147259	0.8111413	**0.9021739**

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. They show, as expected, that $SVMRank_R$ and $SVMRank_T$ in most of cases outperform the *SVMRank* algorithm. By comparing $SVMRank_T$ with $SVMRank_R$, it is possible to see that $SVMRank_T$ shows better performances. This is due to the consideration that taking also into account temporally distant time windows is beneficial. Moreover, in presence of a fast concept drift, the algorithm is not able to immediately adapt to the data. This last aspect is confirmed by results on day 12, when there is small temporal autocorrelation with day 11 (see Table 2).

The Portuguese rivers dataset holds water's information of the rivers Douro e 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 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, % Coliformi Bacteria, conductivity, turbidity, % Escherichia Coli Bacteria. Aggregation is performed

Table 2. Durbin-Watson - Intel lab dataset

Temporal Series	Durbin-Watson
1 - 2	0.0010
2 - 3	0.0017
3 - 4	0.0010
4 - 5	0.0004
5 - 6	0.0028
6 - 7	0.0006
7 - 8	0.0006
8 - 9	0.0033
9 - 10	0.0048
10 - 11	0.0008
11 - 12	0.2649
12 - 13	0.0003
13 - 14	0.0013

Table 3. Portuguese rivers dataset: Spearman's rank coefficient

Train	Test	SVMRank	$SVMRank_R$	$SVMRank_T$
2004-2005	2006	0.2969208	0.3643695	**0.4024926**
2004-2005-2006	2007	0.3592375	0.3526392	**0.4761730**
2004-2005-2006-2007	2008	0.2459677	0.2203079	**0.4769061**
2004-2005-2006-2007-2008	2009	0.2078445	0.2214076	**0.4226539**

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 (reported in Table 3) show that $SVMRank_R$ is not able to improve SVMRank algorithm. This is mainly due to the fact that this dataset does not exhaustively represent the network structure. However, as in the case of the Intel Lab dataset, $SVMRank_T$ significantly outperforms SVMRank and $SVMRank_R$.

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

Table 4. Durbin-Watson - Portuguese rivers dataset

Temporal Series	Durbin-Watson
2004-2005	0.0004
2005-2006	0.0002
2006-2007	0.0001
2007-2008	0.0001
2008-2009	0.0338

Table 5. California traffic dataset: Spearman's rank coefficient

Train	Test	SVMRank	$SVMRank_R$	$SVMRank_T$
2001-2002	2003	**0.7489309**	0.7398230	0.7484557
2001-2002-2003	2004	**0.7484557**	0.7419486	0.7417657
2001-2002-2003-2004	2005	**0.7402829**	0.7255635	0.7402272
2001-2002-2003-2004-2005	2006	0.7465108	0.7412590	**0.7468870**
2001-2002-2003-2004-2005-2006	2007	0.7456102	0.7376070	**0.7447633**
2001-2002-2003-2004-2005-2006-2007	2008	0.7462663	0.7490326	**0.7500496**
2001-2002-2003-2004-2005-2006-2007-2008	2009	0.7370597	0.736184	**0.7375568**

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 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 previously analyzed datasets. Moreover, $SVMRank_T$ performances improve with an increasing history. This result can be motivated by the high temporal autocorrelation of the dataset (see Table 6).

Table 6. Durbin-Watson - California traffic dataset

Temporal Series	Durbin-Watson
2001-2002	0.0013
2002-2003	0.0017
2003-2004	0.0014
2004-2005	0.0009
2005-2006	0.0005
2006-2007	0.0007
2007-2008	0.0010
2008-2009	0.0002

5 Conclusions

In this paper we have faced the problem of mining ranking models from networked data whose data distribution may change over time. The proposed method modifies the well known 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. Extensions are framed in an ensemble learning framework and allow us to take both network and temporal autocorrelation into account. At this aim, we propose two solutions. The first solution uses a fading factor that allows the algorithm to give more importance to models learned in recent time periods than models learned in the past. The second solution allows the algorithm to give more importance to models associated to more correlated time intervals than models associated to less correlated time intervals.

We evaluate our approach on several real world problems of learning to rank from network data, coming from the area of sensor networks. Experimental results empirically prove that the second solution significantly outperforms the first solution in capturing the concept drift.

Acknowledgment. The authors thank Luis Torgo for kindly providing Portuguese rivers dataset. This work is supported in fulfillment of the research objectives of the project: "EMP3: Efficiency Monitoring of Photovoltaic Power Plants" funded by "Fondazione Cassa di Risparmio di Puglia".

References

1. Aiolli, F.: A preference model for structured supervised learning tasks. In: ICDM, pp. 557–560. IEEE Computer Society (2005)
2. Ceci, M., Appice, A., Loglisci, C., Malerba, D.: Complex objects ranking: a relational data mining approach. In: Shin, S.Y., Ossowski, S., Schumacher, M., Palakal, M.J., Hung, C.C. (eds.) SAC, pp.1071–1077. ACM (2010)
3. Cohen, W.W., Schapire, R.E., Singer, Y.: Learning to order things. J. Artif. Int. Res. 10, 243–270 (1999)
4. Crammer, K., Singer, Y.: Pranking with ranking. In: NIPS, pp. 641–647. MIT Press (2001)
5. 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)
6. Doyle, J.: Prospects for preferences. Computational Intelligence 20(2), 111–136 (2004)
7. Draper, N., Smith, H.: Applied Regression Analysis, 2nd edn. Wiley, New York (1981)
8. Fürnkranz, J., Hüllermeier, E.: Pairwise Preference Learning and Ranking. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) ECML 2003. LNCS (LNAI), vol. 2837, pp. 145–156. Springer, Heidelberg (2003)
9. 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)
10. 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)
11. Herbrich, R., Graepel, T., Bollmann-sdorra, P., Obermayer, K.: Learning preference relations for information retrieval (1998)
12. Herbrich, R., Graepel, T., Obermayer, K.: Large margin rank boundaries for ordinal regression. MIT Press (2000)
13. Hüllermeier, E., Fürnkranz, J., Cheng, W., Brinker, K.: Label ranking by learning pairwise preferences. Artif. Intell. 172(16-17), 1897–1916 (2008)
14. 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)

15. 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)
16. Malerba, D., Ceci, M.: Learning to Order: A Relational Approach. In: Raś, Z.W., Tsumoto, S., Zighed, D.A. (eds.) MCD 2007. LNCS (LNAI), vol. 4944, pp. 209–223. Springer, Heidelberg (2008)
17. Neville, J., Simsek, O., Jensen, D.: Autocorrelation and relational learning: Challenges and opportunities. In: Wshp. Statistical Relational Learning (2004)
18. Newman, M.E.J., Watts, D.J.: The structure and dynamics of networks. Princeton University Press (2006)
19. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab (November 1999), `http://ilpubs.stanford.edu:8090/422/`, previous number = SIDL-WP-1999-0120
20. 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. LNCS, vol. 6913, pp. 333–348. Springer, Heidelberg (2011)
21. Swanson, B.J.: Autocorrelated rates of change in animal populations and their relationship to precipitation. Conservation Biology 12(4), 801–808 (1998)
22. 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)
23. Vapnik, V., Golowich, S.E., Smola, A.: Support vector method for function approximation, regression estimation, and signal processing. In: Advances in Neural Information Processing Systems 9, pp. 281–287. MIT Press (1996)