Network Regression with Predictive Clustering Trees

Daniela Stojanova[1], Michelangelo Ceci[2], Annalisa Appice[2], and Sašo Džeroski[1]

[1] Jožef Stefan Institute, Department of Knowledge Technologies
Jamova cesta 39, SI-1000 Ljubljana, Slovenia
{daniela.stojanova,saso.dzeroski}@ijs.si
[2] Dipartimento di Informatica, Università degli Studi di Bari "Aldo Modo"
via Orabona 4, I-70126 Bari, Italy
{ceci,appice}@di.uniba.it

Abstract. Regression inference in network data is a challenging task in machine learning and data mining. 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 values of the response variable at a given node depend on the values of the variables (predictor and response) at the nodes connected to the given node. This phenomenon is a direct violation of the assumption of independent (i.i.d.) observations: At the same time, it offers a 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 paper, we propose a data mining method that explicitly considers autocorrelation when building regression models from network data. The method is based on the concept of predictive clustering trees (PCTs), which can be used both for clustering and predictive tasks: PCTs are decision trees viewed as hierarchies of clusters and provide symbolic descriptions of the clusters. In addition, PCTs can be used for multi-objective prediction problems, including multi-target regression and multi-target classification. Empirical results on real world problems of network regression show that the proposed extension of PCTs performs better than traditional decision tree induction when autocorrelation is present in the data.

1 Introduction

Network data describe entities represented by nodes generally of the same type such as web-pages or telephone accounts, which may be connected with (related to) each other by edges which represent various explicit relations such as hyperlinks between web-pages or people calling each other. Recently, data networks such as social networks, financial transaction networks, sensor networks and communication networks have become ubiquitous in everyday life. This ubiquity of data networks motivates the recent focus of research in data mining to extend traditional inference techniques in order to learn in network data.

D. Gunopulos et al. (Eds.): ECML PKDD 2011, Part III, LNAI 6913, pp. 333–348, 2011.

However, the extension of the traditional inference techniques opens several issues when dealing with data networks. In particular, many data networks are characterized by a form of autocorrelation where the values of the response variable at a given node depend on the values of the variables (predictor and response) at the nodes connected to the given node. Here autocorrelation is defined as the property that a value observed at a node depends on the values observed at neighboring nodes in the network. Different definitions of autocorrelation are in use depending on the field of study which is being considered and not all of them are equivalent. In statistics, autocorrelation is defined as the cross-correlation of a variable with itself at certain time lag. In spatio-temporal and time-series analysis, spatial autocorrelation has been defined as the correlation among data values, which is strictly due to the relative location proximity of the objects that the data refer to. It is justified by the Tobler's [13] first law of geography according to which "everything is related to everything else, but near things are more related than distant things" whereas in network studies the autocorrelation is defined by the homophily's principle as the tendency of nodes with similar values to be linked with each other [16]. The major difficulty due to the autocorrelation is that the independence assumptions (i.i.d.), which typically underlies machine learning methods, are no longer valid. The violation of the instance independence has been identified as the main responsible of poor performance of traditional machine learning methods [20]. To remedy the negative effects of the violation of independence assumptions, autocorrelation has to be explicitly accommodated in the learned models.

Recently, there has been a number of methods that consider the autocorrelation phenomenon and their success depends on the characteristics of the target domain. One limitation of models that represent and reason with global autocorrelation is that the methods assume the autocorrelation dependencies are stationary throughout the relational data graph [1]. Recent research has explored the use of collective inference techniques to exploit this phenomenon. These techniques achieve more accurate predictions than traditional algorithms that predict data instances individually, without regard to the relationships or statistical dependencies that are prevalent in networked data. Collective inference techniques, on the other hand, collectively predict the target values of related instances simultaneously using similarities that appear among groups of similar objects, for example tendency of friends to share political beliefs, siblings to have similar speech patterns, and linked webpages to have similar topics.

In this work, we develop an approach to modeling ('labeling') non-stationary autocorrelation in network data by using predictive clustering [3]. Predictive clustering combines elements from both prediction and clustering. As in clustering, clusters of examples that are similar to each other are identified, but a predictive model is associated to each cluster. The predictive model assigns new instances to clusters based on their description and provides a prediction for the target property. The benefit of using predictive clustering methods, as in conceptual clustering [18], is that, besides the clusters themselves, they also

provide symbolic descriptions of the constructed clusters. However, in contrast to conceptual clustering, predictive clustering is a form of supervised learning.

Predictive clustering trees (PCTs) are tree structured models that generalize decision trees. Key properties of PCTs are that *i)* they can be used to predict many or all attributes of an example at once, *ii)* they can be applied to a wide range of prediction tasks (classification and regression) and *iii)* they can work with examples represented by means of a complex representation [8], which is achieved by plugging in a suitable distance metric for the task at hand, and *iv)* their tree structure permits to consider different effects of autocorrelation (non-stationariness). In the context of this paper, PCTs can be easily extended to network data in order to take (local and global) autocorrelation into account. The method extends the predictive clustering framework implemented in the CLUS system [3][1]. Given a fully described network (nodes and edges), we evaluate our models on real-world data networks (among them several geographical data networks), comparing to models that reason regardless to the global and local dependencies into the network.

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

2 Related Work

The motivation for this work comes from research reported in the literature for mining networked data and predictive clustering. In the following subsections, we discuss background and related work from both research lines.

2.1 Mining Network Data

In the recent years, numerous algorithms have been designed for modeling a partially labeled network and providing estimates of unknown labels associated with nodes. In general, network learning assumes that data for inference are already in the form of a network and exploit the structure of the network to allow the collective inference. Collective inference targets learning algorithms where various interrelated values are inferred simultaneously such that estimates of neighboring labels influence one another [14,10,24]. Since exact inference is known to be an NP-hard problem and there is no guarantee that data network satisfy the conditions that make exact inference tractable for collective learning, most of the research in collective learning has been devoted to the development of approximate inference algorithms.

Popular approximate inference algorithms include iterative inference, Gibbs sampling, loopy belief propagation and mean-field relaxation labeling. An outline of strengths and weakness of these algorithms is reported in [24]. In general, one of the major advantages of collective learning lies in its powerful ability to learn

[1] The CLUS system is available at http://www.cs.kuleuven.be/∼dtai/clus

various kinds of dependency structures (e.g., different degrees of correlation) [12]. However, as pointed out in [19], when the labeled data is very sparse, the performance of collective classification might be largely degraded due to the lack of sufficient neighbors. This is overcome by incorporating informative "ghost edges" into the networks to deal with sparsity issues [15,19].

Interestingly learning problems similar to the tasks addressed in network learning have been recently addressed outside the areas of network learning and graph mining. In particular, in the area of semi-supervised learning and transductive learning [25] a corpus of data without links is given to the algorithms. The basic idea is to connect data into a weighted network by adding edges (in various ways) based on the similarity between entities and to estimate a function on the graph which guarantees the consistency with the label information and the smoothness over the whole graph [26]. The constraint on smoothness implicitly assumes positive autocorrelation in the graph, that is, nearby nodes tend to share the same class labels (i.e., homophily).

Due to the recent efforts of various researchers, numerous algorithms have been designed for network learning. Anyway, at the best of our knowledge, these algorithms address the prediction problem only in the classification case. Exclusively, in an recent work of Appice et.al. [2] the authors have considered the problem of within network regression with an approach that follows the main idea of iterative inference described in [24]. The learning process resorts in the transductive setting, it is robust to sparse labeling and low label consistency and improves traditional model tree induction across a range of several geographical data networks. However, no final model is provided to the user.

2.2 Building Predictive Clustering Trees

The task of learning predictive clustering trees can be formalized in this way:
Given

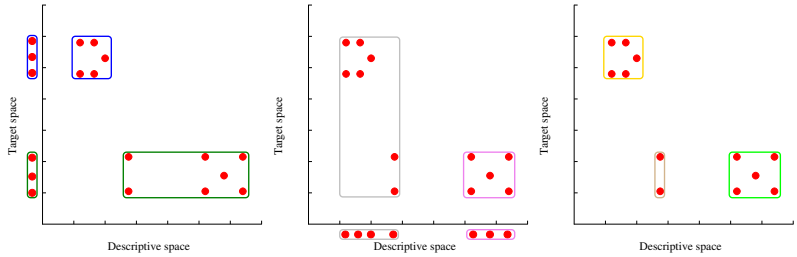
- a descriptive space $\mathbf{X} = X_1, X_2, \ldots X_m$,
- a target space Y,
- a set T of examples (x_i, y_i) with $x_i \in \mathbf{X}$ and $y_i \in Y$

Find

- a set of hierarchically organized clusters defined according to $\mathbf{X} \times Y$,
- a predictive piecewise function $f : \mathbf{X} \rightarrow Y$, defined according to the hierarchically organized clusters.

The clusters to be found are defined on the basis of examples in T and represented according to both the descriptive space and the target space $\mathbf{X} \times Y$ (Figure 1(c)). This is different from what is commonly done in predictive modelling (Figure 1(a)) and classical clustering (Figure 1(b)), where only one of the the spaces is considered.

Note that this general formulation of the problem can take into account different aspects:



(a) (b) (c)

Fig. 1. Illustration of predictive clustering: (a) clustering in the target space, (b) clustering in the descriptive space, and (c) clustering in both the target and descriptive spaces. Note that the target and descriptive spaces are presented as one-dimensional axes for easier interpretation, but can be of higher dimensionality.

- multiple target attributes can be considered at the same time
- the distance function used in the clustering phase can consider the (possible) complex nature of the data
- this formulation is valid both for classification and regression problems (it depends on the nature of Y and on how the function $f(\cdot)$ is built)

In PCTs [3], a decision tree is viewed as a hierarchy of clusters: the top-node corresponds to one cluster containing all data, which is recursively partitioned into smaller clusters while moving down the tree. The construction of PCTs is not very different from that of standard decision tree learners: at each internal node t, a test has to be defined according to a given evaluation function. The main difference is that PCTs select the best test by maximizing the (inter-cluster) variance reduction, defined as $\Delta_Y(A, \mathcal{P}) = Var(A) - \sum_{A_k \in \mathcal{P}} \frac{|A_k|}{|A|} Var(A_k)$, where A represents the examples in T and $\mathcal{P}$ defines the partition $\{A_1, A_2\}$ of A.

If the variance $Var(\cdot)$ and the predictive function $f(\cdot)$ are considered as parameters, instantiated for the specific learning task at hand, it is possible to easily adapt PCTs to different domains and different tasks. To construct a regression tree, for example, the variance function returns the variance of the given instances' target values, and the predictive function is the average of target values in a cluster. Indeed, by appropriately defining the variance and predictive functions, PCTs have been used for clustering ([3]), multi-objective classification and regression ([3,7]), and time series data analysis ([8]).

In this paper, we propose to extend the problem of constructing PCTs by considering the network dimension in addition to the descriptive and target spaces, to explicitly taking autocorrelation into account.

3 Learning PCTs from Network Data

3.1 The Problem

A network is a set of entities connected by edges. Each entity is called a node of the network. A number (which is usually taken to be positive) called weight is

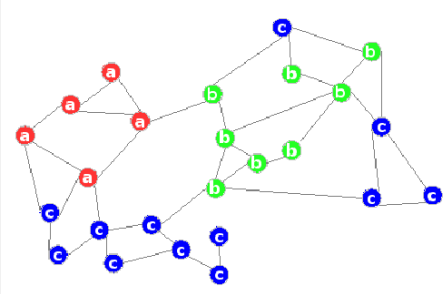


Fig. 2. Autocorrelation in network data. Different labels are given in different colors

associated with each edge. In a general formulation, a network can be represented as a (weighted) graph that is a set of nodes and a ternary relation which represent both the edges between nodes and the weight associated to each edge. Formally, a data network G is a pair (V, E), where:
1. V is a set of nodes, and
2. E is a set of weighted edges between nodes, that is,
$G = \{\langle u, v, w\rangle | u, v \in V, w \in \Re^+\}$.

The network G is represented by an adjacency matrix W with entries $w_{ij} > 0$ if there is an edge between i and j, and $w_{ij} = 0$ otherwise. We impose $w_{ii} = 0$ and we define the degree of a node u_i as: $d(u_i) = \sum_j w_{ij}$. Figure 2 shows an example of a data network where different colors represent different node labels. In practice, when the original data come in the form of a network, the weight w_{ij} usually has a natural interpretation. It could be the number of hyperlinks between two web pages, or a binary value indicating whether proteins i and j interact. Many times when the weights are not readily available from the data, they are computed based on symmetric and nonnegative similarity measures. For instance, if each node is represented in the Euclidean space R_d, a popular choice is to use the Gaussian similarity measure (1)

$$w_{ij} = \exp(\parallel l_i - l_j \parallel^2 / 2\, b^2) \tag{1}$$

where $l_i \in R_d$ describes the location of node i and b is referred to as the bandwidth. In addition, we use a weighting function linearly dependent on the inverse Euclidean distance between objects (2) and a modified Gaussian kernel density function (3):

$$w_{ij} = (1 - \parallel l_i - l_j \parallel / b) \tag{2}$$

$$w_{ij} = (1 - \parallel l_i - l_j \parallel^2 / b^2) \tag{3}$$

which we refer to as "Euclidean" and "Modified", respectively. Whatever weighting function is selected, the estimated parameter surfaces will be, in part, functions of the definition of that weighting function.

In this work, each node of the network is associated with a data observation $(x, y) \in \mathbf{X} \times Y$. $\mathbf{X}$ is a feature space spanned by m predictor variables X_i with $i = 1, \ldots, m$, while $\mathbf{Y}$ is the possibly unknown response variables with a range in $\Re$.

In order to formalize the learning task we are referring to, we need to define the network arrangement of the data with the goal of explicitly taking autocorrelation into account. For this purpose, in addition to the descriptive space X and the target space Y, it is necessary to add information on the connectedness of the data within the network (e.g., the links between the objects involved in the analysis or the pairwise distances between them).

The problem of network regression is formulated as follows.
Given:
1. a set of labeled nodes $L = \{u | u \in \mathbf{X} \times \mathbf{Y}\}$
2. a set of unlabeled nodes $U = \{u | u \in \mathbf{X}\}$
3. a neighborhood function $\eta_k : V \longmapsto (\mathbf{V} \times \Re^+)^k$ such that:
$\eta_k(u) = \{(v_1, w_1), ..., (v_k, w_k)\}$ with $(u, v_i, w_i) \in A, i = 1...k$
Find an estimate for the unknown value of the response variables $\mathbf{Y}$ for each node $u \in U$, such that it is as accurate as possible.

3.2 Measures of Network Autocorrelation

The original CLUS algorithm uses variance reduction as an evaluation measure for the tests used in the internal nodes in the tree. However, in order to take the autocorrelation into account when partitioning the descriptive space, a different measure is necessary. Therefore, 3 autocorrelation indexes are introduced below.

In spatial data analysis, several spatial autocorrelation statistics have been defined. The most common one is the Global Moran's I [13]. This requires a spatial weight matrix that reflects the intensity of the spatial relationship between observations in a neighborhood.

The Global Moran's I is defined as

$$I_Y = \frac{N \sum_i \sum_j w_{ij}(Y_i - \overline{Y})(Y_j - \overline{Y})}{\sum_i \sum_j w_{ij} \sum_i (Y_i - \overline{Y})^2} \tag{4}$$

where N is the number of spatial objects indexed by i and j; Y_i and Y_j are the values of the variable Y for the nodes u_i and u_j, respectively; Y is the variable of interest; $\overline{Y}$ is the overall mean of Y; and $w_{ij}, i, j = 1, \ldots, N$ are the values of a N x N matrix of spatial weights. Values that are more positive than expected indicate positive autocorrelation, while more negative values indicate negative autocorrelation. Values generally range from -1 to +1 and 0 indicates a random distribution of the data.

Connectivity (Randic) Index (CI) is the index of connectivity of a network (or graph) [23]. For a given network G, CI is defined by (5)

$$CI_Y(G) = \sum_{u,v \in V(G)} \frac{1}{\sqrt{d(u)d(v)}} \tag{5}$$

Algorithm 1. Top-down induction of NetworkPCTs

1: **procedure** NetworkPCT(A) **returns** tree
2: **if** stop(A) **then**
3: **return** leaf(Prototype(A))
4: **else**
5: $(c^*, h^*, \mathcal{P}^*) = (null, 0, \emptyset)$
6: **for each** possible test c **do**
7: $\mathcal{P}$ = partition induced by c on A
8: $h = \frac{\alpha}{|Y|} \sum_{\mathbf{Y}} \Delta_Y(A, \mathcal{P}) + \frac{(1-\alpha)}{|Y|} \sum_{\mathbf{Y}} S_Y(A, \mathcal{P})$
9: **if** $(h > h^*)$ **then**
10: $(c^*, h^*, \mathcal{P}^*) = (c, h, \mathcal{P})$
11: **end if**
12: **end for**
13: **for each** $A_k \in \mathcal{P}^*$ **do**
14: $tree_k$ = NetworkPCT(A_k)
15: **end for**
16: **return** node(c^*, $\bigcup_k \{tree_k\}$)
17: **end if**

where $d(u)$ and $d(v)$ represent the degree of nodes u and v respectively, and $V(G)$ represents the node set. This index gives the connectedness (or branching) of a network G and can be used to compare the connectivity among networks. It is typically used in chemistry, since it can be well correlated with a variety of physico-chemical properties of alkanes, such as boiling points, surface area and solubility in water.

Relational autocorrelation measures the strength of statistical dependencies between values of a single attribute Y on related (linked) instances [1]. Any traditional measure of association, such as the χ_2 statistics or information gain, can be used to assess the association between these values of Y. For example, given a set of related pairs P_R, we can measure the autocorrelation of a continuous variable Y as the correlation between related Y_i and Y_j :

$$P_Y = \frac{\sum_{ij \ s.t. \ (u_i,u_j) \in P_R} (Y_i - \overline{Y})(Y_j - \overline{Y})}{\sum_{ij \ s.t. \ (u_i,u_j) \in P_R} (Y_i - \overline{Y})^2} \tag{6}$$

3.3 PCTs for Network Regression

The Algorithm. We can now proceed to describe the top-down induction algorithm for building PCTs from network data (Algorithm 1). It is a recursive method which takes as input a set of training instances A and partitions the descriptive space until a stopping criterion is satisfied (Algorithm 1 line 2).

The main loop (Algorithm 1, lines 6-11) searches for the best attribute-value test c^* that can be associated to a node t. It associates the best test c^* to the internal node t and calls itself recursively to construct a subtree for each subset (cluster) in the partition P^* induced by c^* on the training instances.

Possible tests are of the form $X \leq \beta$ for continuous attributes, and $X \in \{x_{i_1}, x_{i_2}, \ldots, x_{i_o}\}$ (where $\{x_{i_1}, x_{i_2}, \ldots, x_{i_o}\}$ is a subset of the domain D_X of X) for discrete attributes. For continuous attributes, possible values of β are found by sorting the distinct values of X in the training set associated to t, then considering a threshold between each pair of adjacent values. Therefore, if the cases in t have k distinct values for X, at most $k-1$ thresholds are considered. When selecting a subset of values for a discrete attribute, we rely on a non-optimal greedy strategy [17]. It starts with an empty set $Left_t = \oslash$ and a full set $Right_t = D_X$, where D_X is the domain of X. It moves one element from $Right_t$ to $Left_t$ such that the move results in increased variance reduction. This differs from the classical solution [4], where some ordering on the possible values of D_X is defined apriori, according to the data distribution. However, the classical solution cannot deal with multi-objective predictive tasks as PCTs can.

The algorithm evaluates the best split according to the formula reported in Algorithm 1, line 8. This formula is a linear combination of the variance reduction and the statistic $S_Y(A, \mathcal{P})$, computed only on labeled examples. According to the above discussion for the selection of an appropriate evaluation measure for the tests, $S_Y(A, \mathcal{P})$ can be defined in terms of the three indexes we introduce (Moran's I, CI and and P). However, since they all range in different intervals, it is necessary to appropriately scale them. Since the variance reduction is non-negative, we decided to scale them in the interval [0,2], where 2 means high positive autocorrelation and 0 means high negative autocorrelation. For example, for Moran's I, $S_Y(A, \mathcal{P})$ is:

$$S_Y(A, \mathcal{P}) = \sum_{A_k \in \mathcal{P}} \frac{|A_k|}{|A|} \widehat{I_Y}(A_k)$$

where $\widehat{I_Y}(A_k)$ is the scaled Moran's I computed on A_k.

Moreover, in order to guarantee a fair combination of the variance reduction and the statistic $S_Y(A, \mathcal{P})$, we also need to scale the variance reduction in the interval [0,2]. For that purpose, we use a common scaling function:

$$\Delta_Y(A, \mathcal{P}) = 2\frac{\Delta_Y(A, \mathcal{P}) - \Delta min}{\Delta max - \Delta min} \tag{7}$$

where Δmax and Δmin are the maximum and the minimum values of $\Delta_Y(A, \mathcal{P})$ for a particular split.

The search stops when the number of the examples in a leaf is less than $\sqrt{N}$, which is considered a good locality threshold that does not permit to lose too much in accuracy also for rule based classifiers [11]. When the stopping criterion is satisfied, the algorithm creates a leaf and labels it with a predictive function (in this case, the average) defined for the instances falling in that leaf. When predicting multiple variables, the predictive function is an aggregation function (in this case, the average) over tuples of target values. Each target variable contributes equally to the overall h value (Algorithm 1, line 8).

Estimating the Bandwidth. The choice of the bandwidth (denoted by b in (1)) is perhaps the most critical decision to be taken in the modeling process. This parameter controls the degree of smoothing. A small bandwidth results in very rapid distance decay, whereas a larger value will result in a smoother weighting scheme. At the same time, this parameter influences the level of autocorrelation.

The bandwidth may be defined manually or by using some form of adaptive method, such as cross validation and the corrected Akaike Information Criterion (AIC), as used in GWR [9]. A wrapper solution would significantly increase (by a logarithmic factor, in the worst case) the NCLUS complexity. In this study, for the bandwidth estimation we minimize the leave-one-out cross validated - Root Mean Square Error (RMSE). Minimization is performed by means of the Golden section search [5] that aims, in this case, at binary recursively partitioning of the bandwidth domain. Partitions are not uniform in width, but maintain the *golden ratio* $\gamma = \frac{1+\sqrt{5}}{2}$. For each couple of bandwidth values, b_1 and b_2 (at the first iteration, they are initialized as minimum and maximum bandwidth, respectively), the algorithm identifies a point b_3 between them, according to the golden ratio and computes the cross validated - RMSE for that point ($RMSE_{b_3}$). The algorithm than identifies the only parabola with a vertical axis that intersects the points $\{(b_1, RMSE_{b_1}), (b_3, RMSE_{b_3}), (b_2, RMSE_{b_2})\}$. On the basis of the position of the minimum of this parabola, the system decides whether to consider (b_1, b_3) or (b_3, b_2) as the next couple of bandwidth values. The search stops when there is no cross validated - RMSE reduction. In the algorithm, RMSE is computed by fitting a weighted linear model for the example left out. A wrapper solution would significantly increase (by a logarithmic factor, in the worst case) the NCLUS complexity. While we optimize b only for (1), additional experiments have shown only minor differences among for (2) and (3).

Time Complexity. The computational complexity of the algorithm depends on the computational complexity of adding a splitting node t to the tree, which in fact depends on the complexity of selecting a splitting test for t. A splitting test can be either continuous or discrete. In the former case, a threshold a has to be selected for a continuous variable. Let N be the number of examples in the training set, then the number of distinct thresholds can be N-1 at worst. They can be determined after sorting the set of distinct values. If m is the number of descriptive variables, the determination of all possible thresholds has a complexity $O(m * N * logN)$ when an optimal algorithm is used for sorting.

For each of the possible thresholds, the system has to compute the measure used of the evaluation of a single split. This computation has, in principle, time-complexity $O(N^2)$; however, it is not necessary to recompute it at each splitting evaluation since partial sums can be incrementally updated depending on the examples that are moved from the right to the left branch. This optimization makes the complexity of the evaluation of a single split $O(N)$. This means that the worst case complexity of adding a splitting node on a continuous attribute is $O(m * (NlogN + N^2))$, that is $O(m * N^2)$. Similarly, for a discrete splitting test, the worst case complexity is $O(m*k*N)$, where k is the maximum number

of distinct values of a discrete variable ($k \leq N$). Therefore, finding the best splitting node (either continuous or discrete) has a complexity of $O(m * N^2)$. For the induction of a complete clustering tree, this complexity, in the worst case, is $O(z * m * N^2)$, where z is the number of internal nodes in the tree.

4 Empirical Evaluation

Before we proceed to presenting empirical results, we provide a description of the used datasets and experimental settings.

4.1 Datasets

In this experimental evaluation, we use six network data obtained from spatial datasets. They are described in the following.

NWE (North-West England) contains census data collected in the European project SPIN!. The data concerns North West England, an area that is decomposed into censual sections or wards for a total of 1011 wards. Census data provided by the 1998 Census is available at ward level. We consider percentage of mortality (target variable) and measures of deprivation level in the ward according to index scores such as the Jarman Underprivileged Area Score, Townsend score, Carstairs score and the Department of the Environments Index, as well as the coordinates of the ward centroid.

The datasets SIGMEA_MS and SIGMEA_MF (MS and MF) [7] are derived from one multi-objective dataset containing measurements of pollen dispersal (crossover) rates from two lines of plants (target variables), that is, the transgenic male-fertile (MF) and the non-transgenic male-sterile (MS) line of oilseed rape. The predictor variables are the coordinates of the sampling point, the cardinal direction and distance of the sampling point from the center of the donor field, the visual angle between the sampling plot and the donor field, and the shortest distance between the plot and the nearest edge of the donor field.

The FOIXA dataset contains measurements of the contamination rate at sampling points located within a conventional field that comes from the surrounding genetically modified (GM) fields within a 400 *ha* large maize production area in the Foixa region in Spain. This includes the coordinates of sampling points, the number of GM fields, the size of the surrounding GM fields, the ratio of the size of the surrounding GM field and the size of conventional field, the average distance between conventional and GM fields.

The GASD (USA Geographical Analysis Spatial Dataset) [22] contains 3106 observations on USA county votes cast in 1980 presidential election. Specifically, it contains the number of votes cast in the 1980 presidential election per county (target attribute), the coordinates, the number of owner-occupied housing units, the aggregate income and the population in each county over 18 years of age.

The Forest Fires (FF) dataset [6] is public available for research at UCI Machine Learning Repository[2]. It collects 512 forest fire observations from the Montesinho park in Portugal, including the coordinates and the burned area of the

[2] http://archive.ics.uci.edu/ml/

forest (response variable), the Fine Fuel Moisture Code (FFMC), the Duff Moisture Code (DMC), the Drought Code (DC), the Initial Spread Index (ISI), the temperature, the relative humidity and the wind speed.

4.2 Experimental Setup

Each geo-referenced dataset is mapped into a data network $G = (V, E)$ that includes a node $u \in V$ and associates it with for each observation $(x_1, ..., x_n, y)$. Let u and v be two distinct nodes in V, there is an edge from u to v labeled with w in E (i.e., $(u, v, w) \in G$) iff v is within a bandwidth b. At the same time b is used in the weighting functions for the measures of network autocorrelation that use weights. If u and v are associated with observations taken at the same geographical site, the weighting of observations collected at this site would be unity. The weighting of other observations will decrease according to a Gaussian curve as the Euclidean distance between u and v increases. For each data network, experiments are performed in order to show the impact of the different weighting schemes (weights are defined according to equations (1), (2) and (3)).

In this paper, we consider the distances between objects based on the distances over the descriptive attributes, the spatial attributes and both of them together. Moreover, several data networks are constructed from the same dataset by varying the bandwidth b, i.e., using 1%, 5%, 10%, 20% from the maximum distance and an automatically estimated bandwidth. The bandwidth b is set using training data only.

Errors are estimated by 10 fold cross-validation. The predictive performance of the proposed system NCLUS is compared with that of the CLUS algorithm, as well as to a modification of CLUS that considers the coordinates as target variables, along with the actual response variables, for the computation of the evaluation measure (henceforth CLUS*). The latter introduces the network arrangement into CLUS without modifying the algorithm itself. We also compare with the Iterative Transductive Learning (ITL) algorithm of Appice et.al. [2] that deals with network regression tasks and also considers autocorrelation.

To evaluate the effect of different definitions of distances between objects in the network, we use the non-parametric Wilcoxon two-sample paired signed rank test [21]. To perform the test, we assume that the experimental results of the two methods compared are independent pairs $\{(q_1, r_1), (q_2, r_2), \ldots, (q_n, r_n)\}$ of sample data. We then rank the absolute value of the differences $q_i - r_i$. The Wilcoxon test statistics WT^+ and WT^- are the sum of the ranks from the positive and negative differences, respectively. We test the null hypothesis H_0: "no difference in distributions" against the two-sided alternative H_1: "there is a difference in distributions". Intuitively, when $WT^+ \gg WT^-$ and viceversa, H_0 is rejected. Whether WT^+ should be considered "much greater than" WT^- depends on the considered significance level. The basic assumption of the statistical test is that the two populations have the same continuous distribution. Since, in our experiments, q_i and r_i are average MSE, $WT^+ \gg WT^-$ implies that the second method (R) is better than the first (Q). In all experiments reported in this empirical study, the significance level used in the test is set at 0.05.

Table 1. The effect of different definitions of distances between objects in the network

Dataset	Desc.+Spatial vs. Desc.				Desc.+Spatial vs. Spatial				Desc. vs. Spatial			
	++	+	−	−−	++	+	−	−−	++	+	−	−−
NWE	3	4	3	0	2	5	5	0	0	6	6	0
MS	0	0	11	0	0	5	7	0	0	10	2	0
MF	0	6	6	0	0	12	0	0	0	6	5	0
FOIXA	0	4	4	4	1	4	4	3	3	4	3	2
GASD	11	1	0	0	0	12	0	0	0	0	12	0
FF	0	0	0	0	0	0	0	0	0	0	0	0
Total.	14	15	24	4	3	38	16	3	3	26	28	2

4.3 Results and Discussion

Table 1 shows the summary of the different definition of the distances between the objects in the network. Here, we consider three different ways of computing the distances (distance over the descriptive attributes, the spatial attributes and both of them together), all based on the computation of the Euclidean distance. For each of these, we construct the autocorrelation indices (CI, Global Moran's I), two ways of combining the variance reduction and autocorrelation ($\alpha = 0$ and $\alpha = 0.5$) and three different weighting functions for the neighborhood, given by equations (1), (2) and (3). This gives a total of 12 options: The Wilcoxon test is calculated for each of these and the summaries are given in Table 1. The results are presented in terms of the counts of the better (denoted with +) and significantly better (denoted with ++) results at significance level 0.05 for each dataset and the average counts over all datasets when calculating the distance between objects based on the distances over the descriptive attributes, the spatial attributes and both of them together. Although the results very much depend on the datasets characteristics (e.g., NWE, MF and GASD benefit from the inclusion of non-spatial attributes), the distance should be calculated over all (descriptive attributes and spatial) attributes.

The extension that we propose in this paper, introduces several parameters that can be changed by the user within the modeling process. This opens several dimensions through which one can compare the predictive performance of the proposed method, i.e., one can change the the bandwidth, weighting functions, the evaluation measure and the level of consideration of the autocorrelation.

Comparing the results obtained at different bandwidths (1%, 5%, 10%, 20%) enables us to see the influence of the neighborhood similarity on the accuracy of the results. However, in practice, this parameter is very much dependent from the specific data network. Therefore, manual selection of the bandwidth does not lead to a general conclusion for all data networks. Moreover, analysis with different bandwidths (done outside this paper due to the space limitation) confirms that an automatic estimation of the bandwidth as the one explained in 3.3, in most cases, improves the predictive power of the models obtained with a manual selection of the bandwidth. Therefore, we use the estimated bandwidth for the evaluation of the obtained results. Table 2 shows the effect of the weight-

Table 2. Average of the MSE values of NCLUS, CLUS, CLUS* and ITL. NCLUS is run with different weighting functions, evaluation measures and an automatically estimated bandwidth. For each dataset, the best results are given in bold.

Dataset	est b (%)	NCLUS CI						NCLUS P	
		α=0			α=0.5			α=0	α=0.5
		Mod.	Gauss.	Euc.	Mod.	Gauss.	Euc.		
NWE	7.67	0.0023	0.0023	0.0023	0.0024	**0.0022**	0.0024	0.0026	0.0024
MS	4.8	7.1220	6.1312	7.1220	6.8863	6.8863	6.8863	6.2860	7.1380
MF	9.14	2.4718	3.2346	2.4718	2.4718	2.4718	2.4718	2.4981	2.5133
FOIXA	64.62	1.0672	0.9220	1.0672	**0.7666**	0.8011	0.8011	0.9687	0.7313
GASD	2.5	0.1800	0.1808	0.1800	0.1770	0.1663	0.1780	0.1762	0.1734
FF	100	**47.224**	**47.224**	**47.224**	**47.224**	**47.224**	**47.224**	47.385	47.385

Dataset	NCLUS Global Moran						CLUS	CLUS*	ITL
	α=0			α=0.5					
	Mod.	Gauss.	Euc.	Mod.	Gauss.	Euc.			
NWE	0.0024	0.0024	0.0023	0.0023	0.0023	0.0024	0.0025	0.0025	0.0025
MS	7.1844	6.1311	7.3152	6.7851	6.9259	**5.0951**	5.9114	6.6845	5.8532
MF	2.4718	3.0922	2.4718	2.4718	2.4718	4.2877	**2.3532**	2.5390	2.4085
FOIXA	0.8231	0.8240	0.7751	0.8445	1.0201	0.7718	0.8920	0.8710	
GASD	0.1790	0.1688	0.1719	0.1695	0.1688	0.1688	0.1590	0.1590	**0.1316**
FF	**47.224**	**47.224**	**47.224**	**47.224**	**47.224**	**47.224**	47.950	47.998	64.731

ing function and its contribution within the splitting criterion. The results are presented in terms of the average Mean Square Error (MSE) for each evaluation measure, the best results are given in bold. The analysis of the results reveals that the best results are obtained by combining the Euclidian weighting function with the Global Moran spatial statistic and Gaussian weighting function with the Connectivity Index. Note that for this comparison we set $\alpha = 0$.

The selection of the user-defined parameter α is a very important step, influencing the learning process. The simplest solution is to set this parameter to 0 (consider only the network arrangement) or 1 (consider only the variance reduction for regression, as in the original CLUS algorithm). Any other solution will combine the effects, allowing both criterion to influence the split selection. Table 2 also presents the predictive performance (in terms of average MSE) of the proposed algorithm, obtained by varying the parameter α in {0, 0.5, 1}. The best results are obtained for α=0.5 for datasets where the effect of the autocorrelation is not limited to small neighborhoods. Overall, there is a gain in performance due to the used linear combination.

In Table 2, we can see that Connectivity Index CI and Global Moran I show the best measures for network autocorrelation. Moreover, we can also compare NCLUS with the original CLUS and the CLUS* version. The results show that NCLUS outperforms CLUS, CLUS* and ITL when the effect of autocorrelation are relatively high (high values of estimated bandwidth).

5 Conclusions

In this paper, we propose a data mining method that explicitly considers autocorrelation when building regression models from network data. The resulting models adapt to local properties of the data, providing, at the same time, smoothed predictions. The novelty of our approach is that, due to the generality of PCTs, it can work for different predictive modeling tasks, including regression and multi-objective regression, as well as some clustering tasks. We use well known measures of (spatial and relational) autocorrelation, since we deal with a range of several geographical data networks. The heuristic we use in the construction of PCTs is a weighted combination of variance reduction (related to predictive performance) and the autocorrelation of the response variable(s). It can also consider different sizes of neighborhoods (bandwidth) and different weighting schemes (degrees of smoothing) when calculating the autocorrelation. We identify suitable combinations of autocorrelation metrics and weighting schemes and automatically determine the appropriate bandwidth.

We evaluate our approach on six sets of geographical data. Empirical results on real world problems of network regression show that the proposed extension of PCTs performs better than both PCTs that capture local regularities but do not take into account autocorrelation and iterative transductive learner with co-training that takes into account autocorrelation. Future work will further exploit the network structure and study additional evaluation measures.

Acknowledgment. This work is in partial fulfillment of the research objectives of the project "EMP3: Efficiency Monitoring of Photovoltaic Power Plants" funded by Fondazione Cassa di Risparmio di Puglia. Džeroski is supported by the Slovenian Research Agency (grants P2-0103, J2-0734, and J2-2285), the European Commission (grant HEALTH-F4-2008-223451), the Centre of Excellence for Integrated Approaches in Chemistry and Biology of Proteins (operation no. OP13.1.1.2.02.0005) and the Jožef Stefan International Postgraduate School. Stojanova is supported by the Jožef Stefan Institute and the grant J2-2285.

References

1. Angin, P., Neville, J.: A shrinkage approach for modeling non-stationary relational autocorrelation. In: Proc. 8th IEEE Intl. Conf. on Data Mining, pp. 707–712 (2008)
2. Appice, A., Ceci, M., Malerba, D.: An iterative learning algorithm for within-network regression in the transductive setting. In: Discovery Science, pp. 36–50 (2009)
3. Blockeel, H., De Raedt, L., Ramon, J.: Top-down induction of clustering trees. In: Proc. 15th Intl. Conf. on Machine Learning, pp. 55–63 (1998)
4. Breiman, L., Friedman, J., Olshen, R., Stone, J.: Classification and Regression trees. Wadsworth & Brooks, Belmont (1984)
5. Brent, R.: Algorithms for Minimization without Derivatives. Prentice-Hall, Englewood Cliffs (1973)
6. Cortez, P., Morais, A.: A Data Mining Approach to Predict Forest Fires using Meteorological Data. In: Proc. 13th Portuguese Conf. Artificial Intelligence, New Trends in Artificial Intelligence, pp. 512–523 (2007)

7. Demšar, D., Debeljak, M., Lavigne, C., Džeroski S.: Modelling pollen dispersal of genetically modified oilseed rape within the field. In: Abstracts of the 90th ESA Annual Meeting, p. 152. The Ecological Society of America (2005)
8. Džeroski, S., Gjorgjioski, V., Slavkov, I., Struyf, J.: Analysis of time series data with predictive clustering trees. In: Proc. 5th Intl. Wshp. on Knowledge Discovery in Inductive Databases, pp. 63–80. Springer, Heidelberg (2007)
9. Fotheringham, A.S., Brunsdon, C., Charlton, M.: Geographically Weighted Regression: The Analysis of Spatially Varying Relationships. Wiley, Chichester (2002)
10. Gallagher, B., Tong, H., Eliassi-Rad, T., Faloutsos, C.: Using ghost edges for classification in sparsely labeled networks. In: Proc. 14th ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining, pp. 256–264 (2008)
11. Góra, G., Wojna, A.: RIONA: A classifier combining rule induction and k-NN method with automated selection of optimal neighbourhood. In: Elomaa, T., Mannila, H., Toivonen, H. (eds.) ECML 2002. LNCS (LNAI), vol. 2430, pp. 111–123. Springer, Heidelberg (2002)
12. Jensen, D., Neville, J., Gallagher, B.: Why collective inference improves relational classification. In: Proc. 10th Intl. Conf. on Knowledge Discovery and Data Mining, pp. 593–598 (2004)
13. Legendre, P.: Spatial autocorrelation: Trouble or new paradigm? Ecology 74(6), 1659–1673 (1993)
14. Macskassy, S., Provost, F.: Classification in networked data: a toolkit and a univariate case study. Machine Learning 8, 935–983 (2007)
15. Macskassy, S.A.: Improving learning in networked data by combining explicit and mined links. In: Proc. 22th Intl. Conf. on Artificial Intelligence, pp. 590–595 (2007)
16. McPherson, M., Smith-Lovin, L., Cook, J.: Birds of a feather: Homophily in social networks. Annual Review of Sociology 27, 415–444 (2001)
17. Mehta, M., Agrawal, R., Rissanen, J.: Sliq: A fast scalable classifier for data mining. In: Apers, P.M.G., Bouzeghoub, M., Gardarin, G. (eds.) EDBT 1996. LNCS, vol. 1057, pp. 18–32. Springer, Heidelberg (1996)
18. Michalski, R.S., Stepp, R.: Machine Learning: An Artificial Intelligence Approach. In: Learning from Observation: Conceptual Clustering, Tioga, pp. 331–363 (2003)
19. Neville, J., Jensen, D.: Relational dependency networks. Journal of Machine Learning Research 8, 653–692 (2007)
20. Neville, J., Simsek, O., Jensen, D.: Autocorrelation and relational learning: Challenges and opportunities. In: Wshp. Statistical Relational Learning (2004)
21. Orkin, M., Drogin, R.: Vital Statistics. McGraw Hill, New York (1990)
22. Pace, P., Barry, R.: Quick computation of regression with a spatially autoregressive dependent variable. Geographical Analysis 29(3), 232–247 (1997)
23. Randic, M.: On characterization of molecular attributes. Journal of American Chemical Society (1975)
24. Sen, P., Namata, G., Bilgic, M., Getoor, L., Gallagher, B., Eliassi-Rad, T.: Collective classification in network data. AI Magazine 29(3), 93–106 (2008)
25. Vapnik, V.: Statistical Learning Theory. Wiley, New York (1998)
26. Zhu, X., Ghahramani, Z., Lafferty, J.D.: Semi-supervised learning using gaussian fields and harmonic functions. In: Proc. 20th Intl. Conf. on Machine Learning, pp. 912–919 (2003)