Value-transformation for Monotone Prediction by Approximating Fuzzy Membership Functions

Tomáš Horváth*§, Alan Eckhardt†, Krisztián Buza‡, Peter Vojtáš†, Lars Schmidt-Thieme*
*Information Systems and Machine Learning Lab, University of Hildesheim, Germany
Email: {horvath, schmidt-thieme}@ismll.uni-hildesheim.de
†Department of Software Engineering, Charles University in Prague. Czech Republic
Email: {Peter.Vojtas, Alan.Eckhardt}@mff.cuni.cz
‡Department of Computer Science and Information Theory, Budapest University of Technology and Economics, Hungary
Email: buza@cs.bme.hu
§Institute of Computer Science, Pavol Jozef Šafárik University in Košice, Slovakia
Email: Tomas.Horvath@upjs.sk

***Abstract*—Monotone prediction problems, in which the target variable is non-decreasing given an increase of the explanatory variables, have became more popular nowadays in many problem settings which fulfill the so-called monotonicity constraint, namely, if an object is better in all attributes as another one then it should not be classified lower. Recent approaches to monotone prediction consider linear ordering on attribute domains, thus the meaning of being better in an attribute is limited to having a larger or a lower value in that attribute. However, this limitation restricts the use of recent approaches in cases where middle or marginal values of an attribute are better, what is natural in many real-world scenarios. We present a simple attribute value-transformation approach in this paper. The idea is to map attribute domains to real values where the mapped values express how the given value of an attribute contributes to higher classification of objects. Thus, we are searching for an data-specific approximation of a fuzzy membership function on the domain of each (numerical) attribute. Then, instead of the original attribute values we use their mapped values to mitigate the violation of monotonicity constraints in the data. Our approach is quite simple and is not limited to numerical attributes only. The described approach was tested and evaluated on benchmark datasets from the UCI machine learning repository.**

I. Introduction

Ordinal classification is a common phenomenon in our everyday life, however we often know it as rating or scoring. We usually rate student performances by grades $(A, B, \dots)$, hotels by stars $(\star, \star\star, \star\star\star, \dots)$, bond of countries and institutions by levels $(AAA, AAB, \dots)$, and so on. Moreover, it seems to be natural – and easier for humans – to classify objects in a natural language, e.g. 'good', 'bad', 'worst' or 'big', 'medium', 'small', etc. In all these cases there is a presence of linear ordering between the classes the objects belong to. Ordinal classification belongs to the set of monotone prediction problems, which are characterized by two concepts, namely, the comparability of objects and the monotonicity induced by a labeling function on data.

We introduce the monotone prediction task in the next chapter since this paper deals with a pre-processing step for this problem. Then, the so-called monotonic noise is discussed, followed by the description of our approach for attribute value-transformation. Finally, the experiments and the corresponding discussion will conclude the paper.

II. Monotone Prediction

First, the comparability of objects is defined. Assume a space $\mathcal{X} = \mathcal{X}_1 \times \ldots \times \mathcal{X}_k$ with a total ordering $\leq_j$ defined on each $\mathcal{X}_j$ for $1 \leq j \leq k$. An element of $\mathcal{X}$ is called object denoted as $\mathbf{x} = (x_1, \ldots, x_k)$. A partial ordering $\preceq$ on $\mathcal{X}$ is defined as

$$\mathbf{x} \preceq \mathbf{y} \iff (\forall j \in \{1, \ldots, k\}) \quad x_j \leq_j y_j \tag{1}$$

We call two objects $\mathbf{x}, \mathbf{y} \in \mathcal{X}$ *comparable* if either $\mathbf{x} \preceq \mathbf{y}$ or $\mathbf{y} \preceq \mathbf{x}$, otherwise $\mathbf{x}$ and $\mathbf{y}$ are incomparable. Identical objects will be denoted as $\mathbf{x} = \mathbf{y}$, non-identical as $\mathbf{x} \neq \mathbf{y}$.

Second, the labeling function and its interpretation from the monotonicity point of view is discussed. We can define a labeling function as $l : \mathcal{X} \to \mathcal{L}$. If $\mathcal{L} \subset \mathbb{N}$ we are talking about classification function and if $\mathcal{L} \subseteq \mathbb{R}$ then we are talking about regression function. If $\mathfrak{L}$ denotes the set of labeling functions then we can define a function $MP : \mathcal{X} \times \mathcal{X} \times \mathfrak{L} \to \{0, 1\}$ as

$$MP(\mathbf{x}, \mathbf{y}, l) = \begin{cases} 1 & \text{if} \quad (\mathbf{x} \preceq \mathbf{y} \wedge \mathbf{x} \neq \mathbf{y}) \Rightarrow l(\mathbf{x}) \leq l(\mathbf{y}) \\ & \text{or} \quad \mathbf{x} = \mathbf{y} \Rightarrow l(\mathbf{x}) = l(\mathbf{y}) \\ 0 & \text{otherwise} \end{cases}$$

We call the pair of objects $(\mathbf{x}, \mathbf{y})$ monotone under the labeling l if $MP(\mathbf{x}, \mathbf{y}, l) = 1$. We call the labeling l *monotone* if all the pairs of objects are monotone under l, i.e. the following holds

$$(\forall \mathbf{x}, \mathbf{y} \in \mathcal{X}) \quad MP(\mathbf{x}, \mathbf{y}, l) = 1 \tag{2}$$

Such a monotone labeling isn't rare in real life. For example, let the results from several courses represent students attributes and let's have a given student with an aggregated rating B. If any other student has at least as good results from all the courses, she should be rated at least B, i.e. B or A.

As usual, we don't know l exactly, instead we have a training sample $\mathcal{T} = \{(\mathbf{x}^i, l_{\mathcal{T}}(\mathbf{x}^i))\}$ providing some information about l, where $1 \leq i \leq n$, $\mathbf{x}^i \in \mathcal{X}_{\mathcal{T}} \subset \mathcal{X}$ and $l_{\mathcal{T}}(\mathbf{x}^i) \in \mathcal{L}$. We call the sample $\mathcal{T}$ monotone, if the following holds:

$$(\forall \mathbf{x}, \mathbf{y} \in \mathcal{X}_{\mathcal{T}}) \quad MP(\mathbf{x}, \mathbf{y}, l_{\mathcal{T}}) = 1 \tag{3}$$

The goal of prediction is to approximate l by $\hat{l}$ as closely as possible, by learning $\hat{l}$ from the given sample $\mathcal{T}$. In *monotone prediction* problems, l is assumed to be monotone and $\hat{l}$ should also fulfill the monotonicity constraint, i.e.

$$(\forall \mathbf{x}, \mathbf{y} \in \mathcal{X}) \quad MP(\mathbf{x}, \mathbf{y}, \hat{l}) = 1 \tag{4}$$

After the *monotone predictor* $\hat{l}$ is built, the estimation of its prediction performance on a test set $\mathcal{S} = \{(\mathbf{x}^i, l_{\mathcal{S}}(\mathbf{x}^i))\}$ is the following important step, where $1 \leq i \leq m$, $\mathbf{x}^i \in \mathcal{X}_{\mathcal{S}} \subset \mathcal{X}, \mathcal{X}_{\mathcal{S}} \cap \mathcal{X}_{\mathcal{T}} = \emptyset$ and $l_{\mathcal{S}}(\mathbf{x}^i) \in \mathcal{L}$. The usually used prediction error measure for regression problems is the mean-squared error $MSE(\hat{l}, \mathcal{S}) = \sum_{\mathbf{x} \in \mathcal{X}_{\mathcal{S}}} (l_{\mathcal{S}}(\mathbf{x}) - \hat{l}(\mathbf{x}))^2$. For classification problems the misclassification error $MCE(\hat{l}, \mathcal{S}) = \sum_{\mathbf{x} \in \mathcal{X}_{\mathcal{S}}} (1 - b(l_{\mathcal{S}}(\mathbf{x}) = \hat{l}(\mathbf{x})))$ is usually used, where $b(\cdot) = 1$ if the condition $(\cdot)$ in its parameter holds, otherwise $b(\cdot) = 0$. Cross-validation [16] is used in general to get a model with good generalization capabilities and to avoid the so-called overfitting. In cross-validation the sample $\mathcal{T}$ is divided into training set and validation set, the model learned on the training set is validated on the validation set, and finally, the model with least prediction error is chosen.

Monotone prediction problems have in recent years became attractive in data mining and machine learning communities. Monotone approaches have been developed in a wide variety of machine learning and data mining models as regression [19], neural networks [8], support vector machines [7], bayesian networks [1], nearest neighbor classification [10] and decision trees [2], [5], [14], [21]. We will not discuss the mentioned algorithms in details, since this paper does not concern a monotone prediction algorithm, but, a data pre-processing approach for such algorithms. A deep discussion and a taxonomy of the above mentioned algorithms, as well as their experimental comparison can be found in [4].

III. Monotonic Noise

Usually, the prediction algorithms mentioned here assume datasets with monotone labeling (monotone datasets) with numerical attributes on input. An algorithm described in [14] is able to handle all attribute types in data; however, the resulting rules may not preserve monotonicity constraints. On the other hand, algorithms introduced in [3], [6] can classify monotonically even from non-monotone datasets but assume only ordinal attributes on input. An experimental comparison of classical and monotone prediction techniques on real benchmark datasets [4] has concluded that there is no significant improve on performance if monotonicity, as an additional information, is utilized. This conclusion follows from the observation of the presence of a so-called *monotonic noise* in real datasets considered as an amount of violation of monotonicity constraints in data. It is probably caused by the fact that real datasets are usually noisy and this noise causes some violation of the mentioned monotonicity constraints. Such a noisy dataset will probably not be – totally – monotone (with no violation of monotonicity constraints, i.e. the monotonicity degree δ introduced in the equation 5 is 1) even if it represents a monotone problem setting. Moreover, human ratings or scoring of objects often produce some pairwise inconsistencies, too.

Such a monotonic noise in an arbitrary data sample $\mathcal{D} = \{(\mathbf{x}^i, l_{\mathcal{D}}(\mathbf{x}^i))\}$, with $\mathbf{x}^i \in \mathcal{X}_{\mathcal{D}} \subset \mathcal{X}, l_{\mathcal{D}}(\mathbf{x}^i) \in \mathcal{L}$, can be measured by the *degree of monotonicity* δ, which is the ratio of monotone pairs to all comparable pairs in $\mathcal{D}$. Formally, it can be computed as

$$\delta(\mathcal{D}) = \frac{\sum_{\mathbf{x},\mathbf{y} \in \mathcal{X}_{\mathcal{D}}} MP(\mathbf{x}, \mathbf{y}, l_{\mathcal{D}})}{\sum_{\mathbf{x},\mathbf{y} \in \mathcal{X}_{\mathcal{D}}} b(\mathbf{x} \preceq \mathbf{y})} \tag{5}$$

where $b(\cdot)$ is the former introduced indicator of truth of the condition in its parameter. The higher the degree of the monotonicity, the less is the monotonic noise in the dataset.

Forty datasets from the UCI Machine Learning repository [23] were used in this paper. The goal is to test the monotonicity degrees of these datasets before a pre-processing, after a baseline pre-processing and finally, by examining the pre-processing method described in this paper. There have been some modifications made on these datasets, as removing object identification attributes since these have no impact on the monotonization, and, transforming non-numerical ordinal attributes to numerical ones (e.g. the values low, $middle$, $high$ were transformed to 1, 2, 3, respectively).

The fifth column of the table I contains the monotonicity degrees δ_{orig} of the datasets used in our experiments. δ_{orig} was computed according to the equation 5 by taking into account only the numerical attributes of instances, i.e. all the nominal attributes were removed from the datasets. "NaN" values mean division by zero, i.e. there were no comparable pairs in the dataset, while "–" means that the monotonicity degree could not be computed since after the removal of nominal attributes as well as the object identifier attribute(s) there have not been any numerical attributes left in the dataset and thus, computation of the monotonicity degree would make no sense. As we can see, there are only 5 datasets out of 40, which are totally monotone when considering only the numerical attributes.

An usual data pre-processing approach for monotone prediction, after removing all the nominal attributes from the dataset, is to measure the direction of influence of each (numerical) attribute on the target attribute, i.e. measuring the correlation between each attribute and the target attribute over all of the objects. If the correlation is negative for an attribute, all the values of this attribute are multiplied with -1. Similar transformation was used in [13] in the so-called data preparation step. The monotonicity degree on a dataset being pre-processed in this way is denoted δ_{corr} in this paper and its values for the datasets are shown in the sixth column of the table I. As can be seen, such a pre-processing improves the monotonicity degrees of the datasets, however in most of the cases δ_{corr} is still less than 1. In such cases, a further monotonization of data is needed. Here it is important to note that the datasets[1] with low monotonicity degrees may probably

[1]e.g. communities, forestfires, servo, agaricus-lepiota, house-votes-84, nursery, poker-hand-train, tae and tic-tac-toe.

TABLE I
DATASETS USED IN THE EXPERIMENTS: # OBJ AND #NUM (#NOM) REFERS TO THE NUMBER OF OBJECTS AND NUMERICAL (NOMINAL) ATTRIBUTES, RESPECTIVELY.

Name	#Obj	#Num	#Nom	δ_{orig}	δ_{corr}	δ_{fuzzy}
auto-mpg	398	7	2	0.204	0.977	**0.979**
breast-canc.	699	10	1	1	1	1
communities	1994	123	5	NaN	NaN	**1**
concrete	1030	9	0	0.953	**0.986**	0.984
forestfir.	517	11	2	0.733	0.719	**0.775**
machine	209	7	3	0.918	0.954	**0.958**
servo	167	3	2	0.333	0.704	**0.85**
slump	103	8	3	NaN	1	1
wdbc	569	31	1	1	1	1
wine	178	13	1	0.333	1	1
winequal.-r	1599	12	0	0.82	0.965	**0.966**
winequal.-w	4898	12	0	0.786	**0.919**	0.899
wpbc	198	33	2	0.962	1	1
abalone	4177	8	1	0.829	0.829	**0.854**
adult	32561	7	8	0.966	0.965	**0.988**
agaricus	8124	1	22	–	–	**1**
austral.	690	15	0	0.977	0.991	0.991
bands	540	21	19	1	0.957	1
car	1728	7	0	0.902	1	1
cmc	1473	8	2	0.8	0.818	**0.845**
crx	690	7	9	0.923	0.963	**0.991**
diagnosis	120	2	6	0.677	0.813	**1**
haberman	306	4	0	**0.893**	0.878	0.874
heart	270	14	0	0.988	**0.996**	0.995
hepatitis	155	7	13	0.98	0.993	**0.996**
horse-colic	300	17	11	1	1	1
house-v.-84	435	1	16	–	–	**0.986**
ionosphere	351	35	0	1	1	1
magic04	19020	11	0	0.446	0.979	**0.985**
mammogr.	961	4	2	0.944	0.944	**0.969**
nursery	12960	2	7	0.694	0.694	**1**
parkinsons	195	23	1	NaN	0.995	**0.997**
pima-ind.	768	9	0	0.978	**0.978**	0.977
poker-hand	25010	11	0	0.693	**0.735**	0.734
post-oper.	90	9	0	0.772	0.825	**0.857**
shuttle-l.	15	7	0	0.833	0.833	**1**
spambase	4601	58	0	0.977	**0.998**	0.987
tae	151	2	4	0.644	0.677	**0.882**
tic-tac-toe	958	1	9	–	–	**0.961**
transfusion	748	5	0	0.807	**0.94**	0.936

not belong to a monotone problem setting, and thus, using monotone prediction as well as further monotonization would be baseless.

To our best knowledge, there is only one approach to dataset monotonization in recent literature. The basic idea of this approach is to find inconsistent objects (violating the monotonicity constraints) in the train set and relabel them, e.g. as is done in [10]. A greedy relabeling algorithm is presented in [9] where the number of non-monotone pairs of objects are reduced by relabeling one object in each step. The problem of relabeling is viewed as a problem of finding a maximum independent set in the monotonicity violation graph in [22]. Another graph-based approach is presented in [13], where the problem of relabeling is converted to an optimal flow network problem in comparability graphs what can be solved in polynomial time. The relabeling of the dataset $\mathcal{S}$ results in the dataset $\mathcal{S}'$ with $\delta(\mathcal{S}') = 1$. The effectiveness of these algorithms is measured in the minimal number of relabeled objects necessary to reach the monotonicity of the dataset. Relabeling was tested on 5 real datasets in [13] where the average ratio of label changes to the number of objects was about 10% (the lowest ratio was below 2%, the highest one was above 18%).

A few methods for generating synthetic monotone datasets were also developed. In [18] monotone data are generated via the Markov Chain Monte Carlo Method. Two algorithms for generating unstructured and structured monotone datasets using directed graphs are presented in [20].

All of the recent approaches consider only linear ordering on attribute domains, i.e. when the higher or the lower values of an attribute have positive influence to the class value. In contrast, we consider partial ordering on attribute domains, allowing the middle or the marginal values to have positive influence to the class value. The presented methods for learning such an ordering for each attribute domain (separately) are a slightly refined versions of those we introduced in [11], [17] for learning user preferences where we assumed that users' preferences on attributes of objects define some partial ordering on the domains of these attributes, i.e. a user have a "most favorite area" in the space of objects implied by the "most preferred values" on attributes of these objects. The same idea forms the basis for our approach, namely that the certain parts of attribute domains influence the classification of objects positively. Revealing these partitions enables us to transform the original dataset to the dataset with the highest degree of monotonicity. If the monotonicity degree is smaller than 1 data re-labeling [9], [10], [22], [13] can be further used to monotonize the dataset.

IV. LEARNING ORDERINGS ON ATTRIBUTE DOMAIN

Let us assume that the dataset (sample) $\mathcal{D} = \{(\mathbf{x}^i, l_{\mathcal{D}}(\mathbf{x}^i))\}, 1 \leq i \leq n$, defined in the previous section belongs to a monotone setting and may contain monotonic noise higher than a certain threshold ϵ, i.e. $\delta(\mathcal{D}) < 1 - \epsilon$. Moreover, not all of the attributes in $\mathcal{D}$ need to be numerical. In case that the monotonic noise is lower than the threshold, i.e $\delta(\mathcal{D}) \geq 1 - \epsilon$, we expect to use the relabeling approach for data monotonization.

A transformation of the dataset $\mathcal{D}$ to a dataset $\mathcal{D}_T \subset \mathbb{R}^k$ is supposed, such that all the attributes in $\mathcal{D}_T$ are numerical, and $\delta(\mathcal{D}_T) \geq 1 - \epsilon$. We assume that there exist mappings $f_j : \mathcal{X}_j \to \mathbb{R}$ for each attribute j, $1 \leq j \leq k$ such that the monotonicity degree of the transformed dataset $\mathcal{D}_T = \{(f(\mathbf{x}^i), l_{\mathcal{D}}(\mathbf{x}^i))\}$ is at least $1 - \epsilon$, where $f(\mathbf{x}^i) = (f_1(x_1^i), \ldots, f_k(x_k^i))$ is a transformed object $\mathbf{x}^i$ for which we keep its original label.

The main difference between our and other approaches is that by using the mappings f_j we are not limited only to linear orderings ($\leq$) on attribute domains. The comparison of two objects as is defined in the equation 1 can thus be generalized as

$$\mathbf{x} \preceq \mathbf{y} \iff (\forall j \in \{1, \ldots, k\}) \quad f_j(x_j) \leq f_j(y_j) \quad (6)$$

Our goal is to approximate mappings f_j by $\hat{f}_j$ from the dataset $\mathcal{D}$. The learning of a given $\hat{f}_j$ is done in an univariate

manner, i.e. from a "reduced" dataset $\mathcal{D}_j = \{(x_j^i, l_{\mathcal{D}}(\mathbf{x}^i))\}$, where $\mathbf{x}^i \in \mathcal{X}_{\mathcal{D}}$ and $x_j^i \in \mathcal{X}_j$ denotes the value of the j-th attribute of an object $\mathbf{x}^i$. Then, $\hat{f}_j$ is induced using simple statistical methods according to the type (numerical or nominal) of the j-th attribute.

A. Numerical Attributes

Four main types of orderings on attribute domains are considered in this work called *lower-best*, *higher-best*, *middle-best* and *marginal-best* according to which values of a certain attribute have more influence to the rating of objects. These types of orderings are illustrated by solid lines on the figure 1, where the "best" value(s) of an attribute are labeled by b.

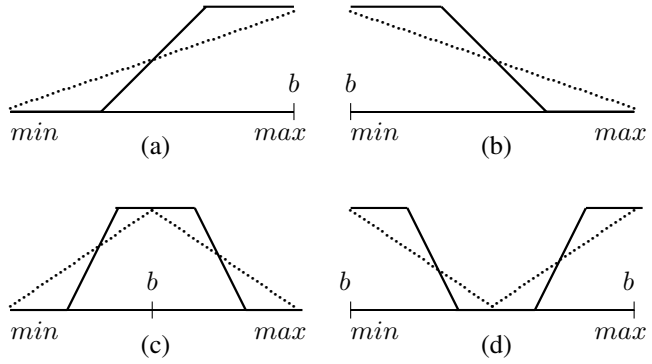

Fig. 1. Four types of orderings on numerical attribute domains: (a) *higher-best*, (b) *lower-best*, (c) *middle-best*, (d) *marginal-best*.

These orderings are common in human thinking – one can prefer cars with low fuel consumption, notebooks with high-speed processor, middle sized flat situated either close or far from the work when she prefers walking and/or driving, etc.

Our approach learns the most simplest approximations of these orderings, i.e. we focus on partially linear functions $\hat{f}_j$ as illustrated by dotted lines on the figure 1.

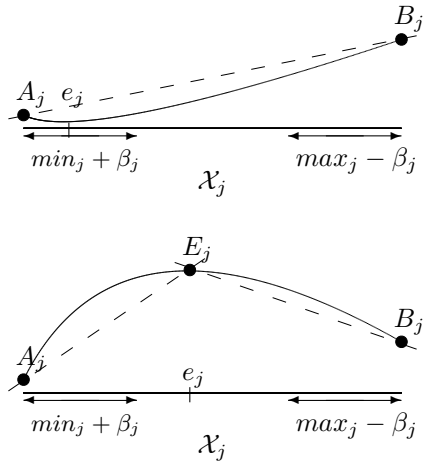

Fig. 2. Illustration of learning $\hat{f}_j$ for numerical attributes.

The main idea of our approach is illustrated on the figure 2[2]. The input is the dataset $\mathcal{D}_j$ for a given attribute j.

We fit $\mathcal{D}_j$ with a parabola $p_j(X_j) = \alpha X_j^2 + \beta X_j + \gamma$, where X_j is a variable. In other words, we estimate the class values of objects from their values in the attribute $\mathcal{X}_j$. Then, there are two possibilities:

First, we can directly use this parabola as a basis function for X_j in a monotone prediction algorithm. Thus, $\hat{f}_j = p_j$, and instead of dealing with real values $x_j \in \mathcal{X}_j$ of objects (from the train set) we feed the prediction algorithm with transformed values $\hat{f}_j(x_j)$. Similarly, we also need to use this transformation for the test objects before predicting their class values.

Second, sometimes we need to interpret the prediction model. A well interpretable model is a monotone decision tree [5], [14], [21] which represents a set of IF-THEN rules with a conjunction of conditions in their bodies. Each of these conditions can be of one of the following forms

$$X_j = v, \quad X_j \geq v, \quad X_j \leq v \tag{7}$$

Since we learn the prediction model from a transformed dataset, the values v in these conditions relate to transformed values. Such rules are hardly interpretable and thus, we would need to have inverse mappings $\hat{f}_j^{-1} : \mathbb{R} \to \mathcal{X}_j$ such that $\hat{f}^{-1}(v) = x$, where $x \in \mathcal{X}_j$. Unfortunately, p_j is not an invertible mapping and we have to (partially) linearize it in the following way, as illustrated on the figure 2:
First, we need to find the global extreme e_j of p_j. Then, we compute the value of p_j in the minimal and maximal value of the domain[3] and in e_j, illustrated on the figure 2 by points $A_j = p_j(min_j), B_j = p_j(max_j)$ and $E_j = p_j(e_j)$.
Finally, to define $\hat{f}_j$ six possible cases are considered, depending if e_j lies in the middle or in the – left or right – border of the domain and if the parabola p_j has a global minimum or maximum in e_j. The borders of the domain are expressed by β_j which is an external parameter of this procedure. If e_j lies on the border of the domain or outside the domain then $\hat{f}_j$ is a line defined by the points A_j and B_j. If e_j lies in the middle of the domain then $\hat{f}_j$ consists of two lines $\hat{f}_{j_{AE}}, \hat{f}_{j_{EB}}$ defined by the points A_j, E_j and E_j, B_j respectively. $\hat{f}_j$ is illustrated with dashed lines on the figure 2.

Such mappings $\hat{f}_j$ are invertible and the conditions introduced in (7) can be converted to conditions with real attribute values according to the type of $\hat{f}_j$ in the following way

- If $\hat{f}_j$ is of the type *higher-best* then the conditions in (7) will be
$$X_j = \hat{f}_j^{-1}(v), \quad X_j \geq \hat{f}_j^{-1}(v), \quad X_j \leq \hat{f}_j^{-1}(v)$$
- If $\hat{f}_j$ is of the type *lower-best* then the conditions in (7) will be
$$X_j = \hat{f}_j^{-1}(v), \quad X_j \leq \hat{f}_j^{-1}(v), \quad X_j \geq \hat{f}_j^{-1}(v)$$

[2]We illustrate only two of the six possible cases, here. The other four cases can be analogously shown.

[3]If we don't have any domain knowledge about the possible values of the domain, we compute the minimal and maximal value from the known values which are present in the dataset.

- If $\hat{f}_j$ is of the type *middle-best* then the conditions in (7) will be

$$X_j = \hat{f}^{-1}_{jAE}(v) \wedge X_j = \hat{f}^{-1}_{jEB}(v),$$
$$X_j \geq \hat{f}^{-1}_{jAE}(v) \wedge X_j \leq \hat{f}^{-1}_{jEB}(v),$$
$$X_j \leq \hat{f}^{-1}_{jAE}(v) \wedge X_j \geq \hat{f}^{-1}_{jEB}(v)$$

- If $\hat{f}_j$ is of the type *marginal-best* then the conditions in (7) will be

$$X_j = \hat{f}^{-1}_{jAE}(v) \wedge X_j = \hat{f}^{-1}_{jEB}(v),$$
$$X_j \leq \hat{f}^{-1}_{jAE}(v) \wedge X_j \geq \hat{f}^{-1}_{jEB}(v),$$
$$X_j \geq \hat{f}^{-1}_{jAE}(v) \wedge X_j \leq \hat{f}^{-1}_{jEB}(v)$$

B. Nominal Attributes

In case of nominal attributes, we just simply compute for every value of the domain the average of class values of objects having a given value in j-th attribute

$$\hat{f}_j(x) = \frac{\sum\limits_{\{\mathbf{x}^i \in \mathcal{X}_D | x^i_j = x\}} l_{\mathcal{D}}(\mathbf{x}^i)}{|\{\mathbf{x}^i \in \mathcal{X}_D | x^i_j = x\}|} \tag{8}$$

where $x \in \mathcal{X}_j$.

Since $\hat{f}_j$ is defined in an explicit form and stored in an appropriate data structure, the computation of the inverse mappings $\hat{f}^{-1}j$ can be made directly from the data structures used. Then the conditions in (7) will transformed to ones with original values as

$$X_j \in \{x \in \mathcal{X}_j | \hat{f}_j(x) = v\},$$
$$X_j \in \{x \in \mathcal{X}_j | \hat{f}_j(x) \geq v\},$$
$$X_j \in \{x \in \mathcal{X}_j | \hat{f}_j(x) \leq v\}$$

The presented approach uses the most simple technique to approximate attribute value orderings in both cases of numerical as well as nominal attributes. The presented techniques are easy to implement, their complexity is linear, which makes them as good baselines for other, more sophisticated approaches to data pre-processing for monotone prediction.

V. Experiments

In our experiments we used 40 datasets shown in Table I. We performed two experiments. In the first one, we transformed the datasets with our approach. we measured the monotonicity degrees and ratios of comparable pairs to all pairs of objects before and after the transformation. In the second experiment we investigated how one of the state-of-the art monotone prediction algorithms performs on the transformed datasets.

A. Settings

Two baselines were used: (i) a simple pre-processing by removing the nominal attributes and (ii) a correlation based preprocessing using only the numerical attributes. The monotonicity degrees of datasets transformed in these ways are denoted as δ_{base} and δ_{corr} in the table I, and were already discussed before.

The parameter β (related to borders of numerical attributes considered in our approach) was set as follows: for the datasets with smaller number of numerical attributes all the combinations of $\beta_j \in 10\%, 30\%, 50\%$ were tried out. Thus $3^{k'}$ cases were tested for each dataset, where k' refers to number of ordinal attributes. Since the small amount of datasets and the number of ordinal attributes in these datasets, this experiment was manageable in feasible time. The communities, wdbc and wpbc datasets were first tested with fixed $\beta = 10\%$, $\beta = 30\%$ and $\beta = 50\%$ for each ordinal attribute. Since, these tests resulted in a totally monotone datasets we did not have to try all combinations (which would be very expensive).

In the second experiment, we evaluated our approach in context of one of the state-of-the art monotone prediction algorithms, concretely the one proposed by Duivesteijn and Feelders [10]. We compared the performance of this monotone prediction algorithm on the transformed datasets. We used the implementation provided by the authors of [10]. The accuracy of the algorithm was tested using 10-fold cross validation with fixed $k = 3$.

B. Results

Table I contains the best monotonicity degrees δ_{fuzz} for each dataset reached across the various combinations on β_j. The best values found for β_j were almost always 30%. As we see, our approach performs quite well, 14 datasets were transformed to totally monotone datasets. In case of 22 datasets δ_{fuzzy} is higher than both δ_{corr} and δ_{base} and in other 10 cases δ_{fuzzy} is at least as high as one of the δ_{corr} or δ_{base}. Here, we also have to mention that our approach deals with all the attributes, not only with the numerical ones.

Besides the monotonicity degrees the percentages of comparable pairs of objects (1) to all pairs of objects $(n(n-1)/2)$ in the transformed datasets were measured, denoted as $\%CP_{base}$, $\%CP_{corr}$ and $\%CP_{fuzzy}$, respectively. The average results over all datasets are the following: $\%CP_{base} = 27.74$, $\%CP_{corr} = 29.90$ and $\%CP_{fuzzy} = 15.04$. This is caused by the fact that the dimensionality in the case of our approach is usually higher that in the case of the two baselines since we keep also the nominal attributes. However, we have found out that there is no strong correlation between the ratio of $\%CP$ and δ in the transformed datasets in all of the three cases. The concept of monotonicity of the dataset doesn't deal with the issue of the number of comparable pairs, it depends only on the number of pairs in the dataset violating the monotonicity constraints. Although, this issue should be further investigated since it can be interesting in further development of monotone prediction algorithms.

As mentioned above, in the second experiment we evaluated our approach in the context of the monotone prediction algorithm proposed by Duivesteijn and Feelders [10], i.e., we compared the accuracy of classification on the datasets transformed by various methods, that were our approach (fuzzy), the correlation based approach (corr) and the basic approach

(base). The algorithm presented in [10] is a nearest-neighbor algorithm performing a relabeling data monotonization step before the computation. Out of the 40 datasets of Table I, we could only use 27 in this experiment because the implementation of the used monotone prediction algorithm did not terminate in reasonable time for the other datasets (auto-mpg, abalone, adult, agaricus-lepiota, cmc, horse-colic, house-votes-84, magic04, nursery, poker-hand, shuttle-landing, tae, tic-tac-toe). In case of 17 and 14 datasets, our approach outperformed the corr and the base approaches, respectively. In some cases, our approach performed significantly better (e.g. diagnosis and bands datasets).

VI. Conclusions

A data pre-processing (value-transformation) approach for monotone prediction problems was proposed in this paper. The main idea, similar to one for user preference learning presented in [17], is to approximate certain types of partial orderings on the domains of attributes of objects. We have chosen the most simple types of orderings which are quite common in human thinking and can be represented by fuzzy membership functions (here, we have to note that we did not normalize the mapped values to the $[0, 1]$ unit interval since it is not necessary because we are interested in orderings).

The presented approach is quite simple, easy to implement, runs in linear time and is not limited to numerical attributes only.

We have used 40 datasets with different characteristics from the UCI machine learning repository [23] in our experiments, where we have tested monotonicity degrees of the transformed datasets as well as how a monotone prediction algorithm from [10] performs on these transformed datasets.

In further investigation of our approach we would like to focus on the following issues: Since outliers have impact on regression, we would like to develop more robust algorithms capable to deal with outliers. As we have mentioned before, an interesting issue would be to investigate how the dimensionality of a dataset influences its monotonicity degree and the data pre-processing approaches.

In comparison to two baselines, our approach was quite successful what makes the plans for further research promising.

Acknowledgements

Tomáš Horváth was supported by the grant VEGA 1/0131/09 and the centre of excellence in computer science and knowledge systems (CaKS - ITMS 26220120007). Krisztián Buza is supported by the grant TÁMOP-4.2.2.B-10/1–2010-0009. We would like to thank to Alexandros Nanopoulos for his comments and suggestions. Finally, our special thanks go to Wouter Duivesteijn and Ad Feelders for providing us their implementation of their algorithm introduced in [10].

References

[1] E. Altendorf, A. Restificar and T. Dietterich, Learning from Sparse Data by Exploiting Monotonicity Constraints, in: *21st Annual Conference on Uncertainty in Artificial Intelligence*, AUAI Press, Arlington, Virginia, USA, 2005, pp. 18–26.

[2] A. Ben-David, Monotonicity Maintenance in Information-Theoretic Machine Learning Algorithms, *Machine Learning* 19 (1995), 29–43.

[3] A. Ben-David, L. Sterling and Y. Pao, Learning and classification of monotonic ordinal concepts, *Computational Intelligence* 5, 1 (1989), 45–49.

[4] A. Ben-David, L. Sterling and T. Tran, Adding monotonicity to learning algorithms may impair their accuracy, *Expert Systems with Applications* 36 (2009), 6627–6634.

[5] J. C. Bioch and V. Popova, Induction of Ordinal Decision Trees: An MCDA Approach, in: *ERIM Report Series* Reference No. ERS-2003-008-LIS (2003).

[6] K. Cao-Van, *Supervised Ranking, from semantics to algorithms*, Ph.D. dissertation, Faculty of Science, University of Ghent, Belgium (2003).

[7] W. Chu and S.S. Keerthi, New approaches to support vector ordinal regression, in: *22nd International Conference on Machine Learning*, ACM, New York, USA, 2005, pp. 145–152.

[8] H. Daniels and B. Kamp, Application of MLP networks to bond rating and house pricing, *Neural Commuting & Applications* 8 (1999), 226–234.

[9] H. Daniels and M. Velikova, Derivation of monotone decision models from non-monotone data, *Discussion Paper* 30, Tilburg University, Center for Economic Research, 2003.

[10] W. Duivesteijn and A. Feelders, Nearest Neighbour Classification with Monotonicity Constraints, in: *European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*, LNAI vol. 5211, Springer-Verlag, Berlin Heidelberg, 2008, pp. 301–316.

[11] A. Eckhardt, T. Horváth and P. Vojtáš, PHASES: A User Profile Learning Approach for Web Search, in: *IEEE/WIC/ACM International Conference on Web Intelligence*, IEEE Computer Society, 2007, pp. 780–783.

[12] R. Fagin, A. Lotem and M. Naor, Optimal Aggregation Algorithms for Middleware, in: *20th ACM Symposium on Principles of Database Systems*, ACM, 2001, pp. 102–113.

[13] A. J. Feelders, M. Velikova and H. Daniels, Two polynomial algorithms for relabeling non-monotone data, *Technical Report* UU-CS-2006-046, Utrecht University, Niederland, 2006, pp: 12.

[14] E. Frank and E. Hall, A Simple Approach to Ordinal Classificaiton, in: *12th European Conference on Machine Learning*, LNCS vol. 2167, Springer-Verlag London, UK (2001), pp. 145–156.

[15] J. Gama and P. Brazdil, Characterization of Classification Algorithms, in: *7th Portuguese Conference on Artificial intelligence*, Lecture Notes In Computer Science, vol. 990, Springer-Verlag, 1995, pp. 189–200.

[16] T. Hastie, R. Tibshirani and J. Friedman, *The Elements of Statistical Learning (2nd edition)*, Springer-Verlag, 2008.

[17] T. Horváth, A Model of User Preference Learning for Content-Based Recommender Systems, *COMPUTING AND INFORMATICS* Vol. 28, No. 4 (2009), 453–881.

[18] K. De Loof, B. De Baets and H. De Meyer, On the random generation of monotone data sets, *Information Processing Letters* 107 (2008), 216–220.

[19] P. McCullagh, Regression models for ordinal data, *Journal of Royal Statistical Society*, 42(2) (1980), 109–142.

[20] R. Potharst, A. Ben-David and M. van Wezel, Two algorithms for generating structured and unstructured monotone ordinal data sets, *Engineering Applications of Artificial Intelligence* 22 (2009), 491–496.

[21] R. Potharst and A. J. Feelders, Classification trees for problems with monotonicity constraints, *SIGKDD Explorations Newsletter* 4, 1 (2002), 1–10.

[22] M. Rademaker, B. De Baets and H. De Meyer, Loss optimal monotone relabeling of noisy multi-criteria data sets, *Information Sciences* 179, 24 (2009), 4089–4096.

[23] A. Asuncion and D.J. Newman, UCI Machine Learning Repository. http://www.ics.uci.edu/~mlearn/MLRepository.html, University of California, School of Information and Computer Science, Irvine, CA, 2007.

[24] M. Velikova, *Monotone Models for Prediction in Data Mining*, PhD Thesis, Tilburg University, Netherland, 2006.

[25] M. Velikova and H. Daniels, On Testing Monotonicity of Datasets, in: *Workshop on Learning Monotone Models from Data at European Conference on Machine Learning and Principles of Knowledge Discovery in Databases*, Bled, Slovenia, 2009, pp. 11–22.