Binary Aggregation Functions in Software Plagiarism Detection

Maciej Bartoszuk
Faculty of Mathematics and Information Science,
Warsaw University of Technology
ul. Koszykowa 75, 00-662 Warsaw, Poland
Email: maciej.bartoszuk@mini.pw.edu.pl

Marek Gagolewski
Systems Research Institute,
Polish Academy of Sciences
ul. Newelska 6, 01-447 Warsaw, Poland
Faculty of Mathematics and Information Science,
Warsaw University of Technology
ul. Koszykowa 75, 00-662 Warsaw, Poland

***Abstract*—Supervised learning is of key interest in data science. Even though there exist many approaches to solving, among others, classification as well as ordinal and standard regression tasks, most of them output models that do not possess useful formal properties, like nondecreasingness in each independent variable, idempotence, symmetry, etc. This makes them difficult to interpret and analyze. For instance, it might be impossible to determine the importances of individual features or to assess the effects of increasing the values of predictors on the behavior of a chosen response variable. Such properties are especially important in software plagiarism detection, where we are faced with the combination of degrees to which how much a code chunk A is similar to (or contained in) B as well as how much B is similar to A. Therefore, in this paper we consider a new method for fitting B-spline tensor product-based aggregation functions to empirical data. An empirical study indicates a highly competitive performance of the resulting models. Additionally, they possess an intuitive interpretation which is highly desirable for end-users.**

I. Introduction

Assume that we are given a data frame with m observations and $2n$ features, $\mathbf{X} = [\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}] \in [0,1]^{2n \times m}$, where $m \geq 2n$. Our setup models a situation in which we compare the similarities of two objects $A^{(j)}$ and $B^{(j)}$ by means of n different methods. The i-th method assesses how much A is similar to (or contained in) B, denoted $x_i^{(j)}$, and how much B is similar to A, denoted $x_{n+i}^{(j)}$. Moreover, along with $\mathbf{X}$ we also observe a (row) vector $\mathbf{Y} = [y^{(1)}, \ldots, y^{(m)}] \in [0,1]^{1 \times m}$ of desired outputs corresponding to each of the inputs.

For the sake of illustration, we shall use a dataset from our software plagiarism detection system for the R [1] environment for statistical computing and graphics, which we described in detail in [2], [3]. Here, $n = 4$ methods are used to assess the similarities of two R functions, namely:

- $x_1^{(j)}$ and $x_5^{(j)}$ are based on comparing Program Dependence Graphs (PDGs),
- $x_2^{(j)}$ and $x_6^{(j)}$ are based on the Levenshtein distances between two source codes,
- $x_3^{(j)}$ and $x_7^{(j)}$ are based on the longest common subsequence between string tokens,
- $x_4^{(j)}$ and $x_8^{(j)}$ compare the proportion of common calls to base R functions.

Each method has been implemented in such a way that it measures the degree to which one code chunk is contained in another R function's definition. It returns a single value in the unit interval, where 1 indicates a high level of similarity, while 0 means that they have nothing in common. It is worth noting that the quantities generated by different methods are not on the same scale: a similarity level of 0.66 as returned by one method does not "denote" the same similarity as 0.66 reported by another one.

Let us recall that a k-ary *aggregation function* $\mathsf{F} : [0,1]^k \to [0,1]$ is a mapping at least:

- nondecreasing in each variable, i.e., for all $\mathbf{x}, \mathbf{y} \in [0,1]^k$ with $\mathbf{x} \leq_k \mathbf{y}$ (componentwise), we have $\mathsf{F}(\mathbf{x}) \leq \mathsf{F}(\mathbf{y})$,
- fulfilling $\mathsf{F}(0, 0, \ldots, 0) = 0$,
- enjoying $\mathsf{F}(1, 1, \ldots, 1) = 1$,

see, e.g., [4], [5]. Moreover, we call F *idempotent* if $\mathsf{F}(x, x, \ldots, x) = x$ for all $x \in [0,1]$, and *symmetric* whenever $\mathsf{F}(x_1, x_2, \ldots, x_k) = \mathsf{F}(x_{\sigma(1)}, x_{\sigma(2)}, \ldots, x_{\sigma(k)})$ for any permutation σ of the set $\{1, 2, \ldots, k\}$. Aggregation functions are useful in numerous practical applications, like pattern recognition, multicriteria decision making, statistics, fuzzy logic, and so forth.

A most frequently used approach to modeling the dependent variable as a function of predictors considers a decision rule $\mathsf{D} : [0,1]^{2n} \to [0,1]$ such that:

$$\operatorname*{arg\,min}_{\mathsf{D} \in \mathcal{D}} \sum_{j=1}^{m} \left(\mathsf{D}\left(x_1^{(j)}, \ldots, x_{2n}^{(j)}\right) - y^{(j)} \right)^2,$$

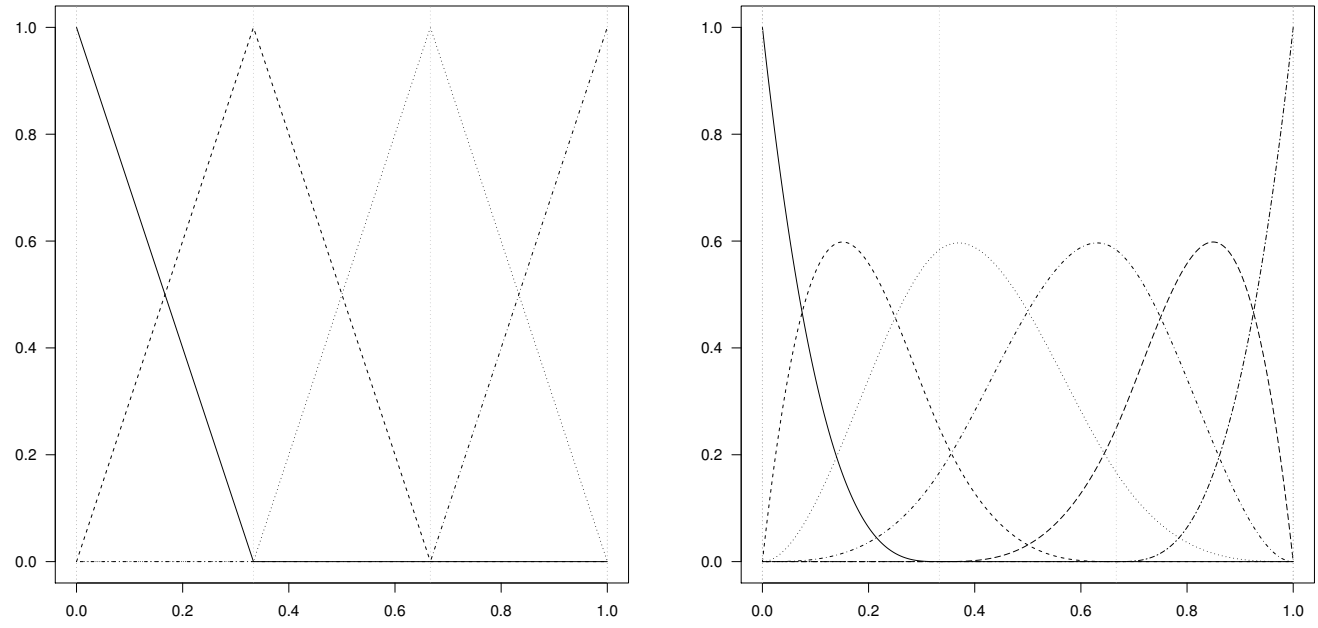
in the case of an ordinary regression task or, for some threshold $e \in [0,1]$:

$$\operatorname*{arg\,min}_{\mathsf{D} \in \mathcal{D}} \sum_{j=1}^{m} \left| \mathbf{1}\left(\mathsf{D}\left(x_1^{(j)}, \ldots, x_{2n}^{(j)}\right) \geq e \right) - \mathbf{1}\left(y^{(j)} \geq e \right) \right|,$$

in the case of a classification task. Here $\mathcal{D}$ denotes the set of admissible models, e.g., linear combinations of input variables or those which can be obtained by constructing decision trees.

Unfortunately, rarely $\mathcal{D}$ consists of decision rules that obey some useful practical properties (like nondecreasingness in each variable) or have an intuitive and appealing interpretation to end-users. The aim of this paper is to demonstrate that by

Please cite this paper as: Bartoszuk M., Gagolewski M., Binary aggregation functions in software plagiarism detection, In: Proc. FUZZ-IEEE'17, IEEE, 2017, pp. 1–6.
doi:10.1109/FUZZ-IEEE.2017.8015582



(a) $p = 1, k = 2$ (b) $p = 3, k = 2$

Fig. 1. Univariate B-spline basis functions $N^{\mathbf{t}}_{i-p,p}(x)$, $i = 1, ..., \eta$ for given p (polynomial degree) and k (number of internal knots).

modeling the set of admissible models based on the notion of aggregation functions, we might obtain human-readable and well-behaving decision rules. Moreover, we generate models of highly competitive predictive power.

We have organized the paper according to the following structure. In Sect. II we recall the notion of B-spline functions and the tensor product of B-splines, which we shall use to aggregate the non-symmetric pairwise similarity measures. In Sect. III we describe the proposed model and provide some implementation details. In Sect. IV we study the model's performance on the aforementioned benchmark dataset. We conclude the paper in Sect. V, where we also point out some areas for future studies.

II. B-spline Curves and Surfaces

Let us introduce the notion of univariate B-splines as well as their tensor product, which we shall use to model bivariate functions (spline surfaces).

Let $p \geq 1$ and $\mathbf{t} = (t_1, \ldots, t_k)$ be a *knot vector* of length $k \in \mathbb{N}$, with $0 < t_1 < \cdots < t_k < 1$. For brevity of notation, we assume that $t_i = 0$ for $i < 1$ and $t_i = 1$ whenever $i > k$. By the way, it is a common setting to rely on equidistant knots.

Assuming that $\cdot/0 = 0$, *B-spline basis functions* for $j = 0, \ldots, p$ and $x \in [0, 1]$ are defined recursively as:

$$
\begin{aligned}
N^{\mathbf{t}}_{i,j}(x) &= \begin{cases} 1 & \text{if } x \in [t_{i-1}, t_i[, \\ 0 & \text{otherwise,} \end{cases} && (j = 0) \\
N^{\mathbf{t}}_{i,j}(x) &= \frac{x - t_{i-1}}{t_{i+j-1} - t_{i-1}} N^{\mathbf{t}}_{i,j-1}(x) \\
&+ \frac{t_{i+j} - x}{t_{i+j} - t_i} N^{\mathbf{t}}_{i+1,j-1}(x). && (j > 0)
\end{aligned}
$$

Denoting $\eta = p + k + 1$, Fig. 1 illustrates some exemplary B-spline basis functions.

Given a vector of *control points* $\mathbf{v} \in [0, 1]^\eta$, we define a *nonperiodic B-spline of degree* p, see, e.g., [6], as a function $B^{\mathbf{t}}_{\mathbf{v}} : [0, 1] \to [0, 1]$ such that:

$$B^{\mathbf{t}}_{\mathbf{v}}(x) = \sum_{i=1}^{\eta} v_i N^{\mathbf{t}}_{i-p,p}(x).$$

In particular, for $p = 1$ we obtain a piecewise linear function interpolating $(0, v_1), (t_1, v_2), \ldots, (t_k, v_{\eta-1}), (1, v_\eta)$, while for $p = 3$ – a cubic B-spline.

Let us now introduce the notion of a B-spline surface. Given $p, q \geq 1$, two knot vectors, $\mathbf{u} = (u_1, \ldots, u_h)$ and $\mathbf{w} = (w_1, \ldots, w_l)$, and a matrix of control points $\mathbf{V} \in [0, 1]^{\eta_1 \times \eta_2}$, $\eta_1 = p + h + 1$, $\eta_2 = q + l + 1$, a *B-spline surface* is a tensor product of two B-splines, i.e., $B^{\mathbf{u},\mathbf{w}}_{\mathbf{V}} : [0, 1]^2 \to [0, 1]$ such that:

$$B^{\mathbf{u},\mathbf{w}}_{\mathbf{V}}(x, y) = \sum_{i=1}^{\eta_1} \sum_{j=1}^{\eta_2} v_{i,j} N^{\mathbf{u}}_{i-p,p}(x) N^{\mathbf{w}}_{j-q,q}(y).$$

III. Proposed Model

In this contribution we propose to aggregate the "non-symmetric" similarity measures $x_i^{(j)}$ and $x_{n+i}^{(j)}$ so as to obtain $\hat{x}_i^{(j)}$. This should be done independently for each $i = 1, \ldots, n$, as the "raw" similarity degrees are not calibrated relatively to each other. Then we may use a weighted mean (which is known to enjoy many practically useful properties) to combine each $\hat{x}_i^{(j)}$ into a single value.

Please cite this paper as: Bartoszuk M., Gagolewski M., Binary aggregation functions in software plagiarism detection, In: Proc. FUZZ-IEEE'17, IEEE, 2017, pp. 1–6.
doi:10.1109/FUZZ-IEEE.2017.8015582

TABLE I
THE BENCHMARK DATASET STRUCTURE

j	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**...**	**30,628**
$x_1^{(j)}$	0.28	0.41	0.64	0.28	0.36	0.10	0.10	0.36	0.23	0.23	1.00	0.18	0.69	0.23	0.28	0.88	...	0.67
$x_2^{(j)}$	0.20	0.21	0.32	0.25	0.23	0.24	0.23	0.26	0.30	0.18	0.62	0.20	0.53	0.23	0.25	0.58	...	0.35
$x_3^{(j)}$	0.00	0.30	0.33	0.00	0.06	0.08	0.15	0.08	0.08	0.00	0.64	0.06	0.55	0.06	0.00	0.41	...	0.17
$x_4^{(j)}$	0.24	0.41	0.56	0.34	0.32	0.29	0.22	0.44	0.46	0.22	0.82	0.15	0.66	0.17	0.29	0.75	...	0.54
$x_{4+1}^{(j)}$	0.41	0.37	0.69	0.39	0.56	0.14	0.44	0.31	0.20	0.38	1.00	0.70	0.90	0.69	0.42	0.74	...	0.59
$x_{4+2}^{(j)}$	0.17	0.21	0.33	0.20	0.20	0.23	0.22	0.23	0.23	0.15	0.54	0.22	0.70	0.22	0.20	0.55	...	0.36
$x_{4+3}^{(j)}$	0.00	0.20	0.26	0.00	0.06	0.08	0.38	0.05	0.06	0.00	0.80	0.20	0.72	0.16	0.00	0.37	...	0.13
$x_{4+4}^{(j)}$	0.25	0.27	0.47	0.39	0.37	0.27	0.60	0.25	0.37	0.25	1.00	0.46	0.87	0.44	0.36	0.65	...	0.44
$y^{(j)}$	0.00	0.25	0.25	0.25	0.25	0.00	0.00	0.25	0.25	0.00	1.00	0.00	0.50	0.00	0.25	0.75	...	0.25

We are interested in solving the optimization task:

$$\text{minimize} \sum_{j=1}^{m} \left(\sum_{i=1}^{n} w_i \hat{x}_i^{(j)} - y^{(j)} \right)^2$$

with respect to a weighting vector $\mathbf{w}$ under the constraints $\mathbf{1}^T\mathbf{w} = 1$ and $\mathbf{w} \geq \mathbf{0}$, see also, e.g., [7]. What is more, $\hat{x}_i^{(j)} = \varphi_i\left(x_i^{(j)}, x_{n+i}^{(j)}\right)$, where $\varphi_1, \ldots, \varphi_n : [0,1]^2 \to [0,1]$ are some automatically generated symmetric binary aggregation functions. Thus, our decision rule can be expressed as:

$$\mathsf{D}_{\boldsymbol{\varphi},\mathbf{w}}\left(\mathbf{x}^{(j)}\right) = \sum_{i=1}^{n} w_i \varphi_i\left(x_i^{(j)}, x_{n+i}^{(j)}\right).$$

It is easily seen that $\mathsf{D}_{\boldsymbol{\varphi},\mathbf{w}}$ is a $2n$-ary aggregation function. Moreover, it is an n-ary idempotent aggregation function of the $\hat{\mathbf{x}}^{(j)} = (\varphi_1(x_1^{(j)}, x_{n+1}^{(j)}), \ldots, \varphi_n(x_n^{(j)}, x_{n+n}^{(j)}))$ variable.

For the j-th input, the intermediate values $\hat{x}_i^{(j)}$ can be reported to the user together with corresponding weights, denoting the importances of the individual features.

Please note that in our previous studies [3], [8] we simply fixed $\varphi_1, \ldots, \varphi_n$ to be the product t-norms and we did not convey any discussion on the effects of choosing different mappings. In the current case, for the sake of computational convenience, we shall model each φ_i using a B-spline *surface*, for which it is straightforward to create constraints for nondecreasingness, symmetry, as well as induce the fulfillment of the two boundary conditions. On a side note, the use of *univariate* B-splines has already been proposed, e.g., in a quasi-arithmetic mean fitting task, see [9], [10], [11], [12].

In order to prevent our setting from being over-complicated, we shall rely on the tensor products of two B-splines having the same degree, p, and based on a single common knot vector $\mathbf{t}$ of length k. Each φ_i is thus given by the formula:

$$\varphi_i\left(x_i^{(j)}, x_{n+i}^{(j)}\right) = \sum_{\alpha=1}^{\eta}\sum_{\beta=1}^{\eta} v_{\alpha,\beta}^{(i)} N_{\alpha-p,p}^{\mathbf{t}}\left(x_i^{(j)}\right) N_{\beta-p,p}^{\mathbf{t}}\left(x_{n+i}^{(j)}\right)$$

for some control point matrix $\mathbf{V}^{(i)} \in [0,1]^{\eta\times\eta}$, $\eta = p+k+1$, fulfilling the following constraints:

- $v_{\alpha,\beta}^{(i)} \leq v_{\alpha+1,\beta}^{(i)}$ and $v_{\alpha,\beta}^{(i)} \leq v_{\alpha,\beta+1}^{(i)}$ for every α, β,
- $v_{1,1}^{(i)} = 0$ and $v_{\eta,\eta}^{(i)} = 1$,
- $v_{\alpha,\beta}^{(i)} = v_{\beta,\alpha}^{(i)}$ for every α, β,

which guarantee that φ_i is then a symmetric aggregation function.

Thus, we finally arrive at the optimization problem:

$$\arg\min_{\mathbf{w},\mathbf{V}^{(1)},\ldots,\mathbf{V}^{(n)}} \sum_{j=1}^{m} \left(\sum_{i=1}^{n} w_i \sum_{\alpha=1}^{\eta}\sum_{\beta=1}^{\eta} v_{\alpha,\beta}^{(i)} N_{\alpha-p,p}^{\mathbf{t}}\left(x_i^{(j)}\right) N_{\beta-p,p}^{\mathbf{t}}\left(x_{n+i}^{(j)}\right) - y^{(j)} \right)^2$$

under the constraints $\mathbf{w} \geq \mathbf{0}$, $\mathbf{1}^T\mathbf{w} = 1$, $v_{1,1}^{(i)} = 0$, $v_{\eta,\eta}^{(i)} = 1$, $v_{\alpha,\beta}^{(i)} = v_{\beta,\alpha}^{(i)}$, $v_{\alpha+1,\beta}^{(i)} - v_{\alpha,\beta}^{(i)} \geq 0$, $v_{\alpha,\beta+1}^{(i)} - v_{\alpha,\beta}^{(i)} \geq 0$, for all α, β, i.

This optimization task can be approached numerically as a bi-level minimization problem. In the inner-level part, we seek control point matrices assuming a fixed weighting vector $\mathbf{w}$. It is worth noting that we can compute the values of B-spline basis functions for every element in $\mathbf{X}$ once and then reuse them as needed with no performance overhead. Due to this, the current part is a standard quadratic programming task with linear constraints.

On the other hand, the outer-part optimizes with respect to $\mathbf{w}$: here a non-linear solver is needed. In our case we used the CMA-ES [13] algorithm. A logarithmic barrier function was used in order to guarantee the fulfillment of the constraints on $\mathbf{w}$.

IV. EXPERIMENTAL RESULTS

In this section we study the results of applying the proposed method on the aforementioned benchmark dataset consisting of pairs of R functions. Some of its excerpts are given in Tab. I. The values in $\mathbf{X}$ are continuous on an interval $[0,1]$. However, the corresponding desired outputs $\mathbf{Y}$ are in

TABLE II
THE DISTRIBUTION OF DESIRED SIMILARITY DEGREES

$y^{(i)}$	**0.00**	**0.25**	**0.50**	**0.75**	**1.00**
proportion	0.0818	0.8948	0.0089	0.0091	0.0054

TABLE III
PERFORMANCE OF REGRESSION MODELS (*mean absolute error* AND *root mean square error*). THE PROPOSED METHOD IS BASED ON (A) $w_1 = 0.30$, $w_2 = 0.03$, $w_3 = 0.13$, $w_4 = 0.54$, $p = 1$, $k = 5$ AND (B) $w_1 = 0.16$, $w_2 = 0.05$, $w_3 = 0.13$, $w_4 = 0.64$, $p = 1$, $k = 1$

method	**MAE**	**RMSE**
Proposed method (a)	**0.047**	0.082
Linear regression	0.063	**0.081**
Proposed method (b) – ordinal, MSE optimized	0.034	0.093
Proposed method (b) – ordinal, MAE optimized	0.033	0.093
Proportional odds model	**0.024**	**0.085**

TABLE IV
PERFORMANCE OF REGRESSION MODELS (SEE TAB. III FOR DETAILS) BASED ON TEST SAMPLE WITH TRUE $y^{(i)} \geq 0.5$ ONLY

method	**MAE**	**RMSE**
Proposed method (a)	**0.130**	**0.164**
Linear regression	0.169	0.205
Proposed method (b) – ordinal, MSE optimized	**0.137**	**0.198**
Proposed method (b) – ordinal, MAE optimized	0.144	0.206
Proportional odds model	0.244	0.307

TABLE V
PERFORMANCE OF OTHER REGRESSION MODELS FOR THE DATA ITEMS PRE-TRANSFORMED BY APPLYING $\varphi_1^*, \ldots, \varphi_4^*$

method	**MAE**	**RMSE**
Linear regression	0.047	0.082
Proportional odds model	0.036	0.097

$\{0.0, 0.25, 0.5, 0.75, 1.0\}$; their distribution is given in Tab. II. This is due to the fact that our experts were asked to assess the similarity degree of each code chunk pair based on the linguistic scale: "totally dissimilar", "dissimilar", "hard to say", "similar", or "very similar".

The number of observations equals to $m = 30{,}628$. The dataset was split into two parts: 80% of the data points constituted the training set, while the remaining 20% – the test set.

A. Scenarios

We decided to take the three following scenarios into account:

- a standard regression task, where we treat the $\mathbf{Y}$ variable as a continuous one; the considered error measures on test data are the *mean absolute error* (MAE) and the *root mean squared error* (RMSE),
- an ordinal regression task, where there are five *linearly ordered* classes: $\{0.0, 0.25, 0.5, 0.75, 1.0\}$ and again error measures are the MAE and RMSE,
- a (imbalanced) binary classification task, where class 1 (plagiarism, a suspiciously similar pair of functions) consists of observations such that $y^{(i)} \geq 0.5$, while the rest are treated as class 0 (not similar); accuracy, precision, recall, and the F-measure (the harmonic mean of precision and recall) on test data serve as quality measures.

We used equidistant knots in the B-spline basis and linear polynomials, $p = 1$. In each case, we fitted 5 models, each for a different $k = 1, \ldots, 5$. The results differed only slightly. In every below-mentioned table, we report the performance of the best among the considered models. Moreover, it is worth noting that we examined the influence of a ridge regression-like regularization coefficient on the inner-level of the optimization, but this did not improve the model quality.

B. Assessing the Model Quality

Error measures for the standard and ordinal regression tasks are given in Tab. III. In comparison with linear regression (`stats::lm()` in R), our method yields a much smaller mean absolute error and only slightly greater root mean squared error. Here, the greater the number of knots k, the better the model quality.

In the case of the ordinal regression task, we decided to introduce the third level of optimization: we searched for thresholds, which split the value of the resulting continuous y into 5 possible classes. Such an approach led to a smaller MAE, while the RMSE has slightly increased. Here, $k = 1$ yielded the best results, yet they were worse than those obtained with the Proportional Odds Model discussed in [14] (`MASS::polr()` in R).

Please note that the true outputs are highly imbalanced (compare Tab. II) and that in practice it is more important to obtain more reliable predictions in cases of true plagiarisms than when we deal with dissimilar code chunks (such data can be filtered out manually in a post-processing step). Therefore, we also decided to inspect "recall-like" error measures on a subset of the test set with $y^{(i)} \geq 0.5$. The results are reported in Tab. IV. This time we observe that the proposed model performs significantly better than the other ones: there seems to be an overfit to the most frequently occuring output class, i.e., 0.25.

Table VI gives the performance of the following binary classifiers (with default parameters):

- random forest (`randomForest::randomForest()` in R),
- support vector machines (`e1071::svm()` in R),
- decision tree (`rpart::rpart()` in R),
- naïve Bayes (`e1071::naiveBayes()` in R),
- logistic regression (`stats::glm(family=binomial)` in R).

Our method yielded the best F-measure and overall accuracy.

TABLE VI
QUALITY OF THE FITTED MODELS (ACCURACY, PRECISION, RECALL, F-MEASURE). THE PROPOSED METHOD IS BASED ON $w_1 = 0.32$, $w_2 = 0.04$, $w_3 = 0.16$, $w_4 = 0.48$, $p = 1$, $k = 3$

method	**accuracy**	**precision**	**recall**	F
Proposed method	**0.998**	0.963	0.970	**0.967**
Random forest	**0.998**	**0.986**	0.945	0.965
Logistic regression	0.997	0.948	0.977	0.962
SVM	0.997	0.965	0.945	0.954
Decision tree	0.996	0.955	0.924	0.939
Naïve Bayes	0.992	0.818	**0.988**	0.895

TABLE VII
PERFORMANCE OF THE FITTED MODELS (ACCURACY, PRECISION, RECALL, F-MEASURE) FOR THE DATA ITEMS PRE-TRANSFORMED BY APPLYING $\varphi_1^*, \ldots, \varphi_4^*$

method	**accuracy**	**precision**	**recall**	F
Random forest	**0.998**	**0.968**	0.968	**0.968**
Logistic regression	0.997	0.950	0.972	0.961
SVM	**0.998**	0.959	0.972	0.966
Decision tree	0.997	0.952	0.965	0.959
Naïve Bayes	0.994	0.844	**1.000**	0.915

C. Effects of Pre-transforming of Data Items

We also decided to test the performance of different methods, like support vector machines (SVM) or linear regression, on data pre-transformed by applying the optimal $\varphi_1^*, \ldots, \varphi_4^*$ ($p = 1, k = 5$ as in case (a) in Tab. III, see Fig. 2), so as to only 4 features $\hat{x}_1, \ldots, \hat{x}_4$ are used in the training step. The results are given in Tab. V (classic and ordinal regression) and Tab. VII (binary classification). Interestingly, the performance of most of the considered classifiers has increased: due to a proper feature engineering scheme – i.e., the aggregation of the non-symmetric similarity measures – we are able to make better predictions.

Please note that in [8] we studied a setting in which each φ_i was set to some monotone transformations of the product t-norm. In particular, in the case of the random forest algorithm, the F-measure was equal to 0.941 and the logistic regression yielded the F-measure of 0.921. We observe that by choosing more suitable φ-functions we were able to improve the classification quality significantly. On the other hand, we did not obtain better error measures in the case of standard and ordinal regression tasks.

V. CONCLUSION

In this work we have introduced a method of fitting a particular hierarchy of aggregation functions into an empirical data sample. We have used the approach known from supervised learning tasks, such as standard and ordinal regression as well as binary classification.

The described model handles a situation in which given inputs consist of pairs of connected features, each on a different measurement (yet ordinal) scale, so without their proper aggregation and calibration they can neither be compared with each other nor aggregated using an idempotent function. The model was tested on a real-world dataset which was based on our plagiarism detection system for R functions. Of course, the presented approach can be applied on other datasets of similar structure.

The described model possesses a very nice and intuitive interpretation. In particular, one can easily assess the importance of each pair of features and analyze how a change in one variable affects the output. Moreover, it turns that the binary aggregation functions φ_i, which are used to "symmetrize" the similarity measures, transform the pairs of features in such a way that also the accuracy of an external classifier may increase.

A more detailed inspection of the fitted φ_i functions is left for further research. For instance, one may ask whether each φ_i can be expressed as $\psi_i \circ T_i$, for some increasing ψ_i and a T-norm (or a uninorm, nullnorm, t-conorm) T_i which could provide an even more intuitive insight and interpretability. Another direction of future research is to use fuzzy membership functions to represent a more gradual behavior in the outcome class and then in the predictive model.

ACKNOWLEDGMENTS

This study was partially supported by the National Science Center, Poland, research project 2014/13/D/HS4/01700.

REFERENCES

[1] R Development Core Team, *R: A language and environment for statistical computing*, R Foundation for Statistical Computing, Vienna, Austria, 2017, http://www.R-project.org.

[2] M. Bartoszuk and M. Gagolewski, "A fuzzy R code similarity detection algorithm," in *Information Processing and Management of Uncertainty in Knowledge-Based Systems, Part III*, A. Laurent *et al.*, Eds., vol. 444. Springer, 2014, pp. 21–30.

[3] ——, "Detecting similarity of R functions via a fusion of multiple heuristic methods," in *Proc. IFSA/Eusflat 2015*, J. Alonso, H. Bustince, and M. Reformat, Eds. Atlantis Press, 2015, pp. 419–426.

[4] G. Beliakov, H. Bustince, and T. Calvo, *A Practical Guide to Averaging Functions*. Springer, 2016.

[5] M. Grabisch, J.-L. Marichal, R. Mesiar, and E. Pap, *Aggregation Functions*. Cambridge University Press, 2009.

[6] L. Schumaker, *Spline Functions: Basic Theory*. Cambridge University Press, 2007.

[7] G. Beliakov, "How to build aggregation operators from data," *International Journal of Intelligent Systems*, vol. 18, pp. 903–923, 2003.

[8] M. Bartoszuk, G. Beliakov, M. Gagolewski, and S. James, "Fitting aggregation functions to data: Part II – Idempotization," in *Information Processing and Management of Uncertainty in Knowledge-Based Systems, Part II*, ser. Communications in Computer and Information Science, J. Carvalho *et al.*, Eds. Springer, 2016, vol. 611, pp. 780–789.

[9] G. Beliakov, "Monotone approximation of aggregation operators using least squares splines," *International Journal of Uncertainty, Fuzziness and Knowledge-based Systems*, vol. 10, pp. 659–676, 2002.

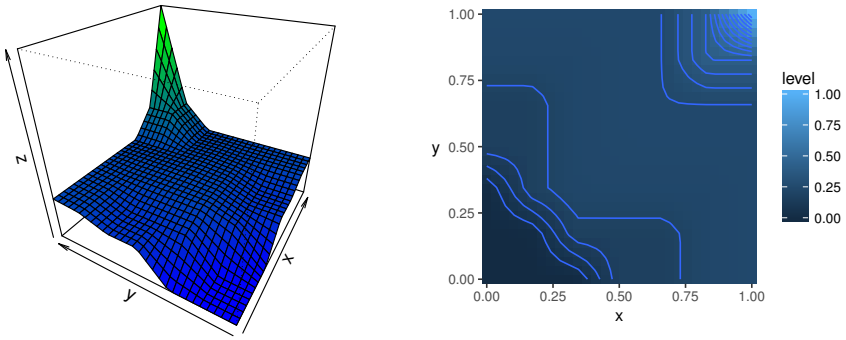
[10] ——, "Learning weights in the generalized OWA operators," *Fuzzy Optimization and Decision Making*, vol. 4, pp. 119–130, 2005.

[11] G. Beliakov, A. Pradera, and T. Calvo, *Aggregation Functions: A Guide for Practitioners*. Springer-Verlag, 2007.

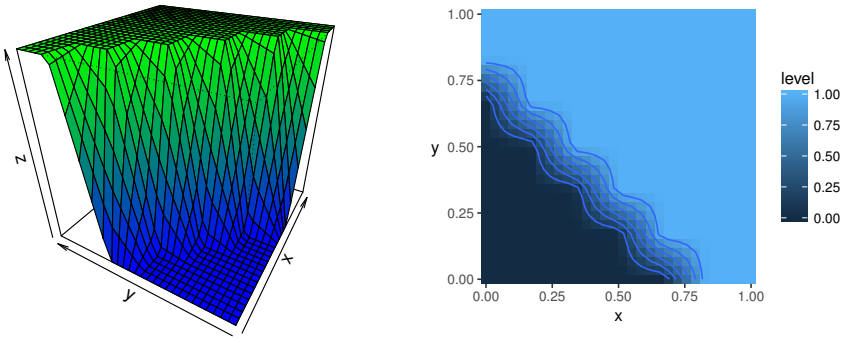
[12] G. Beliakov and J. Warren, "Appropriate choice of aggregation operators in fuzzy decision support systems," *IEEE Transactions on fuzzy systems*, vol. 9, no. 6, pp. 773–784, 2001.

[13] N. Hansen, "The CMA evolution strategy: A comparing review," in *Towards a new evolutionary computation. Advances in estimation of distribution algorithms*, J. Lozano, P. Larranga, I. Inza, and E. Bengoetxea, Eds. Springer, 2006, pp. 75–102.

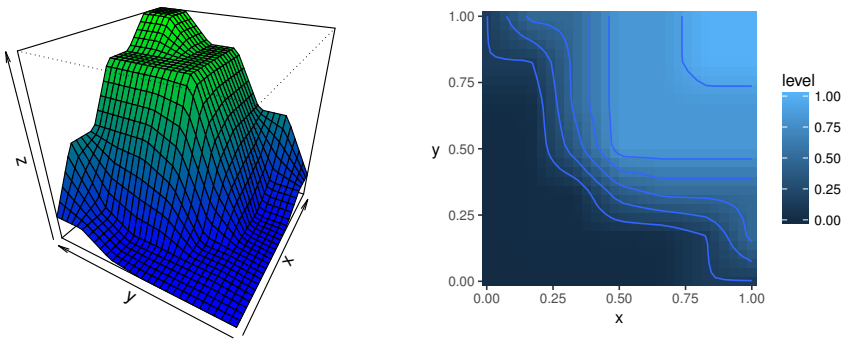
[14] P. McCullagh, "Regression models for ordinal data," *Journal of the Royal Statistical Society*, vol. 42, no. 2, pp. 109–142, 1980.



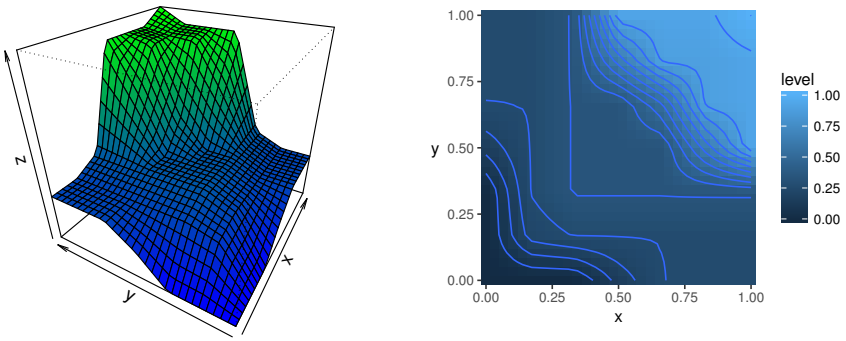
(a) x_1 and x_5



(b) x_2 and x_6



(c) x_3 and x_7



(d) x_4 and x_8

Fig. 2. Bivariate aggregation functions $\varphi_1^*, \ldots, \varphi_4^*$.