European Journal of Operational Research 305 (2023) 781–805

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Decision Support

Deep preference learning for multiple criteria decision analysis

Krzysztof Martyn*, Miłosz Kadziński

Institute of Computing Science, Poznań University of Technology, Piotrowo 2, Poznań 60-965, Poland

ARTICLE INFO

Article history:
Received 7 September 2021
Accepted 24 June 2022
Available online 30 June 2022

Keywords:
Multiple criteria decision aiding
Preference learning
Artificial neural networks
Multiple criteria sorting
Preference disaggregation

ABSTRACT

We propose preference learning algorithms for inferring the parameters of a threshold-based sorting model from large sets of assignment examples. The introduced framework is adjusted to different scores originally used in Multiple Criteria Decision Analysis (MCDA). They include Ordered Weighted Average, an additive value function, the Choquet integral, a distance from the ideal and anti-ideal alternatives, and Net Flow Scores built on the results of outranking-based pairwise comparisons. As a concrete application of these models, we use Artificial Neural Networks with up to five hidden layers. Their components and architecture are designed to ensure high interpretability, which supports the models' acceptance by domain experts. To learn the most favorable values of all parameters at once, we use a variant of a gradient descent optimization algorithm called AdamW. In this way, we make the MCDA methods suitable for handling vast, inconsistent information. The extensive experiments on various benchmark problems indicate that the introduced algorithms are competitive in predictive accuracy quantified in terms of Area Under Curve and the 0/1 loss. In this regard, some approaches outperform the state-of-the-art algorithms, including generalizations of logistic regression, mathematical programming, rule ensemble and tree induction algorithms, or dedicated heuristics.

1. Introduction

The need to process data to conclusions or arguments that support more informed and better decision-making is growing each year (Liu, Kadziński, Liao, & Mao, 2021). Consequently, one of the main trends in today's information technology is developing intelligent decision support systems. Their successful application depends on the quality of being believable or trustworthy (Linkov, Galaitsi, Trump, Keisler, & Kott, 2020). The need to explain the decisions made by computer systems (Doshi-Velez & Kim, 2017) is reflected in the legal regulations of the European Union (Goodman & Flaxman, 2017).

Multiple Criteria Decision Aiding (MCDA) and Machine Learning (ML) belong to the most important and fastest developing disciplines within Artificial Intelligence (AI) (Corrente, Greco, Kadziński, & Słowiński, 2013; Doumpos & Zopounidis, 2011). They offer methods that support humans in decision-making processes. Within the scope of this paper, we focus on multiple criteria sorting (Alvarez, Ishizaka, & Martínez, 2021) or instance ranking (Fürnkranz & Hüllermeier, 2011) problems. They aim at assigning a set of alternatives to preference ordered classes, labels, or degrees in the presence of multiple attributes with pre-defined preference directions. Moreover, we limit our interest to learning ordered classification models from decision examples. In MCDA, they are treated as the DM's indirect preference information in the form of assignment examples (Liu, Liao, Kadziński, & Słowiński, 2019; Zopounidis & Doumpos, 2000), whereas in ML – they form a training set in the task of supervised learning (Doumpos & Zopounidis, 2011). The goal is to find the model for classifying all alternatives, including the ones that have not been judged directly by the Decision Maker (DM) nor considered in the reference set (Doumpos & Zopounidis, 2018).

Even though the paradigm of learning by example is handled by both MCDA and ML, there are notable differences between these two disciplines (Corrente et al., 2013; Doumpos & Zopounidis, 2011; Waegeman, De Baets, & Boullart, 2009). On the one hand, MCDA is user-oriented. It exploits Decision Makers' knowledge or expertise and aims at the DMs to learn about their preferences and the problem at hand. On the contrary, ML is model-oriented, being focussed on data analysis, information extraction, and preference discovery. These various aims are, in turn, reflected in different forms of incorporated models, the amount of processed information, techniques used for arriving at a final result, and the role of users.

The preference models used in MCDA are highly interpretable and explainable (Corrente et al., 2013). Their primary role is to en-

* Corresponding author.
E-mail addresses: krzysztof.martyn@cs.put.poznan.pl (K. Martyn), milosz.kadzinski@cs.put.poznan.pl (M. Kadziński).

https://doi.org/10.1016/j.ejor.2022.06.053
0377-2217/

courage the involvement of the DMs (Roy, 2010) through gaining insights on the role of different criteria, the character of alternatives, and the influence of particular performances on the decision. On the contrary, ML has mainly focused on the development of non-linear models, offering higher predicting ability and the possibility of capturing complex interdependencies (Corrente et al., 2013). However, this results in limited ability to determine which data influences a decision and, consequently, less confidence in the model's employment by the users who need to interpret and understand the underlying process (Waegeman et al., 2009).

The traditional MCDA methods have been designed for learning from a small set of decision examples for a subset of reference alternatives (Doumpos & Zopounidis, 2018). Typically, the translation of assignment examples into compatible values of an assumed preference model has been conducted with mathematical programming techniques that aim at reconstructing the DM's judgments as faithfully as possible. However, when the DM's preference information is rich and highly inconsistent, most approaches cannot deal efficiently with preference disaggregation (Liu et al., 2019). Some exceptions that have in-built mechanisms for dealing efficiently with large sets of inconsistent preferences include variants of the Dominance-based Rough Set Approach (DRSA) (Greco, Matarazzo, & Słowiński, 2001) and UTADIS (Zopounidis & Doumpos, 2000). On the contrary, ML has always been focused on dealing with large, inconsistent sets of training data (Doumpos & Zopounidis, 2011). These are usually composed of historical data, preferences collected over time, or observations of past decisions. In ML, some advanced statistical models or optimization algorithms are used to exploit the parameters space in search of the values that minimize some classification error.

Over the years, MCDA and ML have been developing separately while fostering their interests mentioned above. Nonetheless, the availability of large data resources as well as the need for both explainable models and interpretable decision-making processes have motivated the cross-fertilization of the two disciplines. Individuals, companies, organizations, and governments have accumulated a vast quantity of data, and its analysis has exceeded the reach of human processing capacity. However, it needs to be exploited in a way that allows verifying whether a model focuses on the relevant aspects, offers arguments and knowledge for decision-making, and involves the DM to take part in the process actively. Consequently, one has developed the algorithms that scale up well with an increasing number of assignment examples, at the same time incorporating the intuitive models originally proposed in MCDA (Cinelli, Kadziński, Miebs, Gonzalez, & Słowiński, 2022).

The research at the crossroads of MCDA and ML is called preference learning (Fürnkranz & Hüllermeier, 2011). Within this field, some MCDA methods have been adjusted to deal with large data, leading to the elaboration of intuitive classification methods. In what follows, we list the representative algorithms aimed at multiple criteria sorting and instance ranking problems. In particular, Chandrasekaran, Ryu, Jacob, & Hong (2005) proposed linear programming models based on isotonic separation, and Kotłowski & Słowiński (2013) introduced a family of classifiers exploiting the class of all monotonic functions, not making any additional assumptions about the model apart from the monotonicity constraints. Then, Tehrani, Cheng, Dembczyński, & Hüllermeier (2012) generalized logistic regression to learn the parameters of the Choquet integral, Liu et al. (2021) formulated optimization models for learning additive value functions augmented with components for handling the interactions between criteria, whereas Kadziński & Szczepański, (2022) proposed a variety of methods for learning the parameters of a sorting model with characteristic class profiles. Furthermore, Dembczyński, Kotłowski, & Słowiński (2009) introduced an algorithm based on the variant of DRSA for generating a monotonic rule ensemble and Dembczyński, Kotłowski, & Słowiński (2006) extended DRSA by considering an additive function model resulting from rough approximations. Also, a few approaches have been proposed to learn the parameters of an outranking-based sorting model used in the ELECTRE TRI-B method or its simplified variant called MR-Sort. They include an evolutionary algorithm (Doumpos, Marinakis, Marinaki, & Zopounidis, 2009) or linear programming models combined with simulated annealing (Olteanu & Meyer, 2014) or a dedicated metaheuristic (Sobrie, Mousseau, & Pirlot, 2019).

This paper proposes to use Artificial Neural Networks (ANNs) for preference learning in the context of highly interpretable MCDA models. ANNs are versatile learners that can be applied to nearly any learning task, where input and output data are well-understood, yet the process that relates the input to the output is highly complex. Over the last years, ANNs have been successfully applied in the context of data analysis, control systems, speech and pattern recognition, and computer games. This is mainly due to the development of Deep Learning (DL) (i.e., efficient learning algorithms for ANNs with multiple hidden layers) that has revolutionalized the field of AI and its applicability in the context of big data (Deng & Yu, 2014). However, the employment of ANNs in MCDA has been scarce. In particular, Malakooti & Zhou (1994) used an Adaptive Feedforward Adaptive Feedforward ANN to learn the utility function based on a set of training patterns in the form of alternatives with their associated evaluations by the DM and then applied it to rank a discrete set of alternatives. Moreover, Hu (2009) proposed a single-layer perceptron for multiple criteria classification problems based on pairwise comparisons among alternatives conducted in the spirit of an ELECTRE-based outranking relation. Furthermore, Hanne (1997) suggested the use of ANNs as a part of an MCDA network, in which they can be applied to standardize and aggregate performances from different criteria or even to choose the most relevant method from a pre-defined pool of a few approaches. Finally, Guo, Zhang, Liao, Chen, & Zeng (2021) proposed the NN-MCDA method that combines an additive value model with potentially non-monotonic marginal functions and a fully connected deep neural network.

We introduce the preference learning algorithms that use ANNs to infer parameters of the threshold-based sorting procedure from large sets of assignment examples. In this procedure, following UTADIS (Zopounidis & Doumpos, 2000), the frontiers between classes are delimited by the thresholds on a scale of a comprehensive score that reflects the quality of each alternative from all relevant viewpoints considered jointly. We adjust the introduced framework to different types of scores. In particular, we consider aggregation of the performances on various criteria using OrderedWeighted Average (OWA) operator (Yager, 1988), an additive value function initially employed in UTADIS (Zopounidis & Doumpos, 2000), and the Choquet integral (Angilella, Corrente, Greco, & Słowiński, 2013). These scores are able to capture different compensation levels or interactions between criteria. Moreover, we account for a model postulated in TOPSIS that builds on the distances of a given alternative from the ideal and anti-ideal options (Hwang & Yoon, 1981). Also, we consider the Net Flow Score (NFS) procedures that aggregate the results of pairwise comparisons between all alternatives. The comparisons are conducted in the spirit of the PROMETHEE (Brans & De Smet, 2016) and ELECTRE (Figueira, Greco, Roy, & Słowiński, 2013) methods, exploiting either preference degrees or the outcomes of concordance and discordance tests.

The ANNs have been originally designed to capture complex transformations of inputs (in our case, performances on all criteria) to outputs (in our case, class assignments). We have designed their architecture and adjusted the characteristics of individual units to derive sorting models that are flexible enough to fit the learn-

ing data and sufficiently interpretable due to being inspired by the MCDA methods. This is in line with the recent trend in ML, which postulates making prediction models and their decisions interpretable (Molnar, 2020).

When learning the sorting models, we minimize the loss function defined as an average of regrets for all reference alternatives. The choice of ANNs as a computation technique for conducting preference disaggregation allowed us to use a variety of tools supporting the optimization. In particular, to learn the most favorable value parameters, we employ a variant of a gradient descent optimization algorithm called Adam (Kingma & Ba, 2014). The optimization is enhanced with techniques such as data augmentation to increase the noise resistance, regularization to prevent model overfitting, and batch optimization to reduce the impact of the information processing order on the attained results. The networks deriving the parameters of the OWA-, Choquet-, and distance-based models are shallow. However, the ANNs proposed for UTADIS, PROMETHEE, and ELECTRE can be classified as deep learning models (Deng & Yu, 2014) due to many hidden layers and considering different levels related to the data processing (e.g., criteria, alternatives, pairs of alternatives, and assignments). Hidden layers are required to learn complex models inspired by the value- and outranking-based MCDA methods. However, the raw weight values of multiple layers, some of which conduct non-linear transformations of data, are hardly interpretable for the users. Therefore, we ensure that users are exhibited only with the final models of ANN-UTADIS, ANN-PROMETHEE, and ANN-ELECTRE. These models summarize the comprehensive contribution of individual criteria, resulting from the transformations conducted by various layers, activation performed with non-linear activations functions, and normalization to an easily interpretable range of alternatives' scores.

We conduct a thorough experimental verification of the proposed algorithms on a set of benchmark sorting problems. Its results are quantified in terms of two quality measures for different proportions between the sizes of the training and testing sets. The multiplicity of proposed methods allows indicating which model is most appropriate for a given problem. We also compare the obtained results with the performance of the existing preference learning approaches. These include the Choquistic (Tehrani et al., 2012) and logistic (Hosmer, Lemeshow, & Sturdivant, 2000) regression, Kernel Logistic Regression (KLR) with polynomial and Gaussian kernels, rule ensemble (MORE) (Dembczyński et al., 2009) and tree induction (LMT) (Landwehr, Hall, & Frank, 2003) algorithms, value-based UTADIS model (Zopounidis & Doumpos, 2000), and outranking-based methods incorporating mathematical programming (MIP) (Leroy, Mousseau, & Pirlot, 2011) or a dedicated metaheuristic (META) (Sobrie et al., 2019).

The remainder of the paper is organized in the following way. Section 2 reminds a threshold-based sorting procedure. In Section 3, we discuss the novel preference learning algorithms that incorporate different scores for judging a comprehensive quality of alternatives. Section 4 provides details of the employed optimization techniques. In Section 5, we illustrate the use of the proposed methods on a selected multiple criteria sorting problem for which a large set of assignment examples is available. Section 6 discusses the results of computational experiments, comparing the predictive capabilities of our ANN-based approaches and the state-of-the-art methods. The last section concludes and provides avenues for future research.

2. Threshold-based score-driven multiple criteria sorting

The following notation is used in the paper:

- $A = \{a_1, a_2, \ldots, a_i, \ldots, a_n\}$ – a finite set of n alternatives;
- $A^R = \{a_1^*, a_2^*, \ldots, a_i^*, \ldots\} \subseteq A$ – a finite set of reference alternatives, which the DM accepts to critically judge in a holistic way;
- $G = \{g_1, g_2, \ldots, g_j, \ldots, g_m\}$ – a finite set of m evaluation criteria, $g_j : A \to \mathbb{R}$ for all $j \in J = \{1, \ldots, m\}$;
- $X_j = \{x_j \in \mathbb{R} : g_j(a_i) = x_j, a_i \in A\}$ – a set of all different performances on g_j, $j \in J$; as typical in the field of preference learning, we assume that all performances on g_j, $j = 1, \ldots, m$, are scaled to the [0,1] interval;
- $x_j^1, x_j^2, \ldots, x_j^{n_j(A)}$ – increasingly ordered values of X_j, $x_j^k < x_j^{k+1}$, $k = 1, 2, \ldots, n_j(A) - 1$, where $n_j(A) = |X_j|$ and $n_j(A) \leq n$;
- $C_1, C_2, \ldots, C_p$ – p pre-defined, preference ordered classes, where C_{h+1} is preferred to C_h, $h = 1, \ldots, p-1$ ($H = \{1, \ldots, p\}$).

We consider the problem of sorting imposed by the use of function $f : R^m \to H$ that maps alternative $a_i \in A$ evaluated in terms of m criteria to one of the decision classes C_h, $h = \{1, \ldots, p\}$. To aggregate performances on multiple criteria, we use a function assigning a comprehensive score $Sc(a_i)$ to $a_i \in A$. The maximal score is assigned to an ideal alternative a^+ with the most preferred performances on all criteria, whereas the minimal score is associated with an anti-ideal alternative a^-. The range $[Sc(a^-), Sc(a^+)]$ may differ depending on the applied method. Moreover, the scale of a comprehensive score is divided by a set of class thresholds t_h, $h = 1, \ldots, p-1$, which delimit the intervals implying an assignment to particular decision classes (Köksalan & Özpeynirci, 2009):

$$\begin{aligned} &Sc(a_i) < t_1 \Rightarrow a_i \in C_1, \\ &t_{h-1} \leq Sc(a_i) < t_h \Rightarrow a_i \in C_h, \quad \text{for} \quad h = 2, \ldots, p-1, \\ &Sc(a_i) \geq t_{p-1} \Rightarrow a_i \in C_p. \end{aligned} \tag{1}$$

To avoid direct specification of the parameter values, we assume indirect preference information is available or specified by the DM. It has the form of desired class assignments $C_{DM}(a_i^*)$ for reference alternatives $a_i^* \in A^R$. When constructing or training the sorting model, we will disaggregate holistic preferences to respect the reference assignments in the following way (Doumpos & Zopounidis, 2004):

$$\left.\begin{array}{rl} & \text{for all } a_i^* \in A^R: \\ Sc(a_i^*) \geq t_{C_{DM}(a_i^*)-1}, & \text{if } C_{DM}(a_i^*) > 1, \\ Sc(a_i^*) + \varepsilon \leq t_{C_{DM}(a_i^*)}, & \text{if } C_{DM}(a_i^*) < p, \end{array}\right\} \tag{2}$$

where ϵ is an arbitrarily small positive value. When numerous assignment examples are considered, they might not be reproduced simultaneously. Therefore, in the optimization phase, we will consider the following loss function defined as an average of regrets for all reference alternatives:

$$Minimize: loss = \frac{1}{|A^R|} \sum_{a_i^* \in A^R} regret(a_i^*), \tag{3}$$

where *regret* is equal to the distance from thresholds delimiting the desired class in case an alternative is misclassified or to zero, otherwise:

$$regret(a_i^*) = \max\{t_{C_{DM}(a_i^*)-1} - Sc(a_i^*),\ Sc(a_i^*) - t_{C_{DM}(a_i^*)},\ 0\}. \tag{4}$$

In the following section, we discuss a variety of scoring procedures that will be incorporated in the ANN-based preference learning algorithms. For each of them, the scoring function Sc is defined differently.

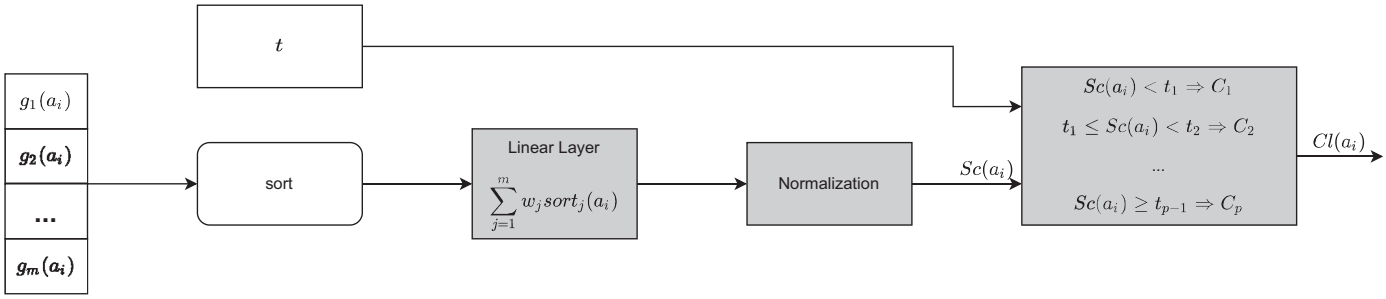


Fig. 1. The architecture of the neural network employed by the ANN-OWA method.

3. Preference learning with artificial neural networks and MCDA-inspired preference models

In this section, we present the MCDA-inspired approaches that learn parameters of the sorting models from large sets of assignment examples. For this purpose, they apply Artificial Neural Networks. We will discuss a variety of methods that implement different strategies for deriving the comprehensive scores of alternatives. Nonetheless, for all of them, the derived model remains easily interpretable, and the sorting results are explainable for a human DM.

3.1. ANN-OWA: preference learning with ordered weighted average and ANN

OWA is an aggregation function generalizing other operators such as min, max, average, median, or sum (Yager, 1988). It aggregates performances using a revised weighted sum:

$$OWA(a_i) = \sum_{j=1}^{m} w_j sort_j(a_i), \tag{5}$$

where $sort_j(a_i)$ is the j-th largest performance of alternative a_i on any criterion and w_j is the weight linked with the j-th position in sorted performance vector of a_i. We assume that $w_j \in \mathbb{R}_{\geq 0}$.

ANN-OWA starts with sorting the performances of each alternative in a non-increasing order (see Fig. 1). Then, a single linear layer aggregates the performances using the OWA operator with non-negative weights. Since the value of OWA can be, in general, arbitrarily large, to increase interpretability of the results, we apply normalization to the [0,1] range by dividing the scores by the sum of weights w_j:

$$Sc_{ANN-OWA}(a_i) = \frac{\sum_{j=1}^{m} w_j sort_j(a_i)}{\sum_{j=1}^{m} w_j}. \tag{6}$$

Such a score is compared against the thresholds $t = [t_1, t_2, \ldots, t_{p-1}]$ to determine the class assignments using Eq. (1) and calculate the regret that is considered when optimizing the network parameters, i.e., weights w_j and thresholds t.

The last component of the ANN responsible for the comparison of a comprehensive score with class thresholds to derive the assignment is the same for all methods presented in the following subsections. Thus, we will not mention it when describing these approaches, instead focussing on the computation of scores in line with the assumptions of different methods. Nevertheless, the thresholds and the underlying sorting procedure will always be depicted in the figures representing the architectures of neural networks.

3.2. ANN-Ch: preference learning with the Choquet integral and ANN

The Choquet integral model is an additive aggregation method, dealing with interactions between criteria (Angilella et al., 2013). It takes the form of a weighted sum over all subsets of criteria $T \subseteq G$, where the performance for T is the minimum over the performances on criteria contained in T:

$$Ch_\mu(a_i) = \sum_{T \subseteq G} w_T \cdot \min_{j \in T} g_j(a_i), \tag{7}$$

where $\sum_{T \subseteq G} w_T = 1$. We limit the considered interactions to pairs of criteria by referring to the 2-additive Möbius transform (Tehrani et al., 2012):

$$Ch_{\mu,2}(a_i) = \sum_{j=1}^{m} w_j g_j(a_i) + \sum_{\{j,l\} \subseteq G} w_{\{j,l\}} \min(g_j(a_i), g_l(a_i)). \tag{8}$$

To respect the pre-defined preference directions for all criteria, we assume that the weights are non-negative:

$$w_j \geq 0, \forall j \in \{1, \ldots, m\}. \tag{9}$$

Moreover, we consider the positive and negative interactions, though limiting their impact on the attained scores in the following way:

$$w_{\{j,l\}} + w_j \geq 0, \quad \forall j \in \{1, \ldots, m\}, \forall l \in \{1, \ldots, m\} \backslash \{j\}. \tag{10}$$

The variant of the method respecting such constraints will be denoted as **ANN-Ch-Constr.** In the pre-processing phase, we perform the Möbius transform of a 2-order additive measure of the input data (see Fig. 2). Then, two linear layers are responsible for aggregating pre-criteria performances using non-negative weights respecting Eq. (9) and interaction components using weights associated with pairs of criteria that respect Eq. (10). Their outputs are summed and normalized to the [0,1] range as follows:

$$Sc_{ANN-Ch-Constr.}(a_i) = \frac{Ch_{\mu,2}(a_i)}{\sum_{j=1}^{m} w_j + \sum_{\{j,l\} \subseteq G} w_{\{j,l\}}}. \tag{11}$$

The parameters optimized by ANN are weights of both linear layers (w_j and $w_{j,l}$) and class thresholds t.

The other two variants of the Choquet integral-based method incorporate different assumptions. The first one, called **ANN-Ch-Pos.**, considers only positive interactions, hence limiting the weights for individual criteria and pairs to non-negative values. The other variant, called **ANN-Ch-Uncons.**, does not impose any constraints on the weights. Moreover, both variants apply normalization of scores with the sigmoid function as proposed in Tehrani et al. (2012):

$$Sc_{ANN-Ch-Sig}(a_i) = sigmoid(Ch_{\mu,2}(a_i) + bias). \tag{12}$$

A diagram showing the network operations for these variants is presented in Fig. 3. First, we perform the Möbius transform. Since there are no constraints involving different weights, per-criteria performances and interaction components can be aggregated using a single linear layer. It performs the calculations defined by Eq. (8) and adds a bias value as defined by Eq. (12). The bias allows the sigmoid function to be shifted, and the lack of restrictions on the sum of weights allows for an arbitrary adjustment of

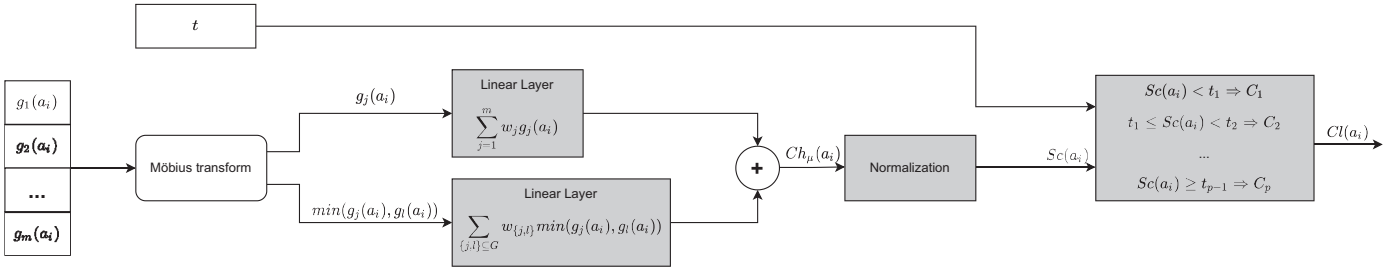


Fig. 2. The architecture of the neural network employed by the ANN-Ch-Constr. method.

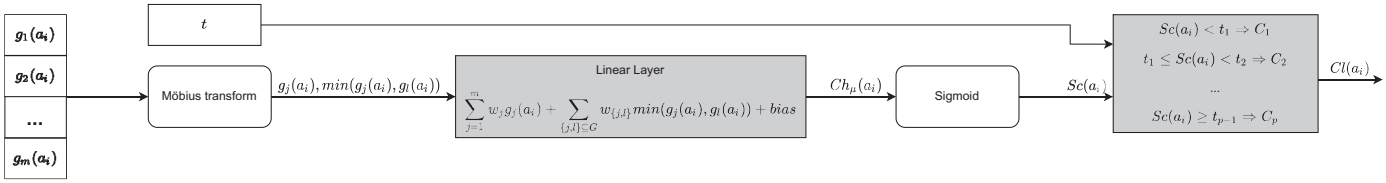


Fig. 3. The architecture of the neural network employed by the ANN-Ch-Pos. and ANN-Ch-Uncons. methods.

the sigmoid function's argument scale. In **ANN-Ch-Pos.**, all weights need to be non-negative. The result from the linear layer is processed by a sigmoid activation function. It ensures that the score for each alternative is in the [0,1] range.

3.3. ANN-TOPSIS: preference learning with TOPSIS and ANN

Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS) considers the ideal a^+ and anti-ideal a^- alternatives with the following performances on each criterion $g_j \in G$ (Hwang & Yoon, 1981):

$$g_j(a^+) = max_{a_i \in A}(g_j(a_i)) \text{ and } g_j(a^-) = min_{a_i \in A}(g_j(a_i)). \quad (13)$$

The closer alternative $a_i \in A$ is to a^+ and the further it is from a^-, the more preferred it is. The respective distances can be computed as follows:

$$d^+(a_i) = \left(\sum_{j=1}^{n} w'_j y_j^+(a_i)\right)^{\frac{1}{z}} \text{ and } d^-(a_i) = \left(\sum_{j=1}^{n} w'_j y_j^-(a_i)\right)^{\frac{1}{z}}, \quad (14)$$

where $w'_j = |w_j|^z$, $w_j \in \mathbb{R}_{\geq 0}$ is the weight associated with criterion $g_j \in G$, $y_j^+(a_i) = |g_j(a^+) - g_j(a_i)|^z$ and $y_j^-(a_i) = |g_j(a_i) - g_j(a^-)|^z$, for $j = 1, \ldots, m$. Overall, the comprehensive score for a_i is computed in the following way:

$$R(a_i) = \frac{d^-(a_i)}{d^-(a_i) + d^+(a_i)}. \quad (15)$$

In this paper, we assume $z = 1$. Thus w'_j can be interpreted as the weight of criterion g_j without any additional transformations.

The architecture of the neural network performing the respective calculations for **ANN-TOPSIS** is presented in Fig. 4. In the preprocessing stage, we compute $y_j^+(a_i)$ and $y_j^-(a_i)$ values for each alternative $a_i \in A$. The linear layer calculates the distances from the ideal and anti-ideal alternatives while using non-negative weights w'_j. It is followed by aggregation according to Eq. (15). The parameters subject to optimization are weights w'_j and class thresholds t. The neural networks for ANN-OWA, all variants of ANN-Ch, and ANN-TOPSIS share the same number of layers, including one input layer, one hidden layer, and one output layer responsible for sorting.

3.4. Modelling monotonic functions with ANNs

To construct ANNs suitable for conducting calculations of more complex MCDA methods, it is necessary to define a monotonic function. It can be seen as transforming per-criteria performances or performance differences, maintaining the pre-defined preference directions. We consider two monotonic functions: non-decreasing and non-increasing for gain- and cost-type criteria, respectively. The transformation of a function from non-decreasing to non-increasing is conducted by negating the function. We define a non-decreasing function as a neural network with a single hidden layer and a continuous sigmoidal activation function with positive weights. According to Cybenko (1989), for an arbitrary continuous sigmoid function σ, function $u(\boldsymbol{x})$ of vector $\boldsymbol{x} \in \mathbb{R}^N$:

$$u(\boldsymbol{x}) = \sum_{k=1}^{L} \alpha_k \sigma(y_k^T \boldsymbol{x} + \theta_k), \quad (16)$$

where $\alpha_k, \theta_k \in \mathbb{R}$ and $y_k \in \mathbb{R}^N$, can approximate any N-dimensional continuous function with precision depending on the number of components L. Also, $u(\boldsymbol{x})$ is equivalent to a neural network with a single hidden layer (Cybenko, 1989).

In what follows, we build on the following two observations. On the one hand, if F is a family of monotonic functions, then $\sum_{f(x)\in F} f(x)$ is also a monotonic function. On the other hand, the linear transformation $\alpha f(x) + \beta$ of a monotonic function f, where $\alpha \in \mathbb{R}_{\geq 0}$ and $\beta \in \mathbb{R}$, is a monotonic function. Assuming $\alpha_k \in \mathbb{R}_{\geq 0}$, $y_k \in \mathbb{R}^N_{\geq 0}$, $\theta_k \in \mathbb{R}$, and σ is a monotonic continuous sigmoidal function, then $u(\boldsymbol{x})$ is also a monotonic function. The values of α_k, y_k, and θ_k will be optimized using an algorithm described in Section 4 by iteratively refining parameter values with function gradients. The major monotonic continuous sigmoidal functions are sigmoid and hard sigmoid functions. However, to avoid a problem of gradient vanishing, in the learning process, we will consider the non-decreasing monotonic function *LeakyHardSigmoid* (see Fig. 5):

$$Leaky\ Hard\ Sigmoid(x) = \begin{cases} \delta x, & if\ x < 0, \\ x, & if\ 0 \leq x \leq 1, \\ \delta(x-1)+1, & if\ x > 1, \end{cases} \quad (17)$$

where δ is a slope factor, being a very small value in the range [0,1). The above function is not a continuous sigmoidal function and cannot be used to approximate any non-decreasing monotonic function. For example, it cannot represent the level segments. However, it is possible to decrease the value of a slope during training to zero. Then, *LeakyHardSigmoid* will be equal to hard sigmoid function. We will consider a one-dimensional space of x and

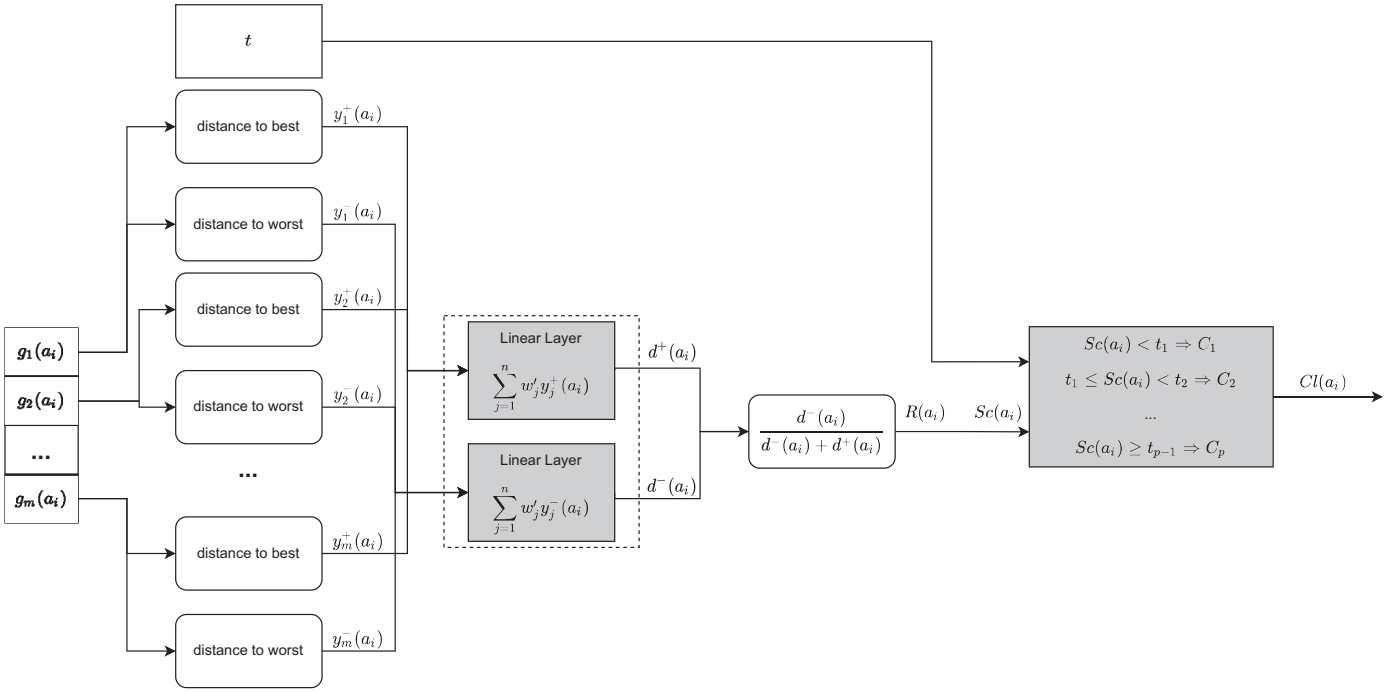


Fig. 4. The architecture of the neural network employed by the ANN-TOPSIS method.

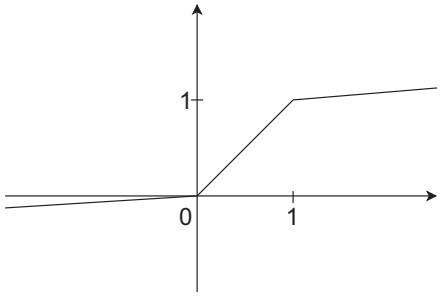


Fig. 5. The *LeakyHardSigmoid* function.

y. Thus, for the sake of simplicity, we assume that:

$$u(x) = \sum_{k=1}^{L} \alpha_k \sigma(y_k x + \theta_k), \tag{18}$$

where σ is *LeakyHardSigmoid* with a slope 0.01, L is the number of components of function u, and $y_k, \alpha_k \in \mathbb{R}_{\geq 0}$. Function $u(x)$ can be considered as a line segment function with ends designated by the individual components. It changes slope only at the characteristic points resulting from the applied σ function. Each component has two characteristic points: $(\frac{-\theta_k}{y_k}, 0)$ and $(\frac{-\theta_k+1}{y_k}, \alpha_k)$, which are projected onto $u(x)$. Such a projection from the component functions on the output model for a single argument x is presented in Fig. 6. Function $u(x)$ is marked with a solid line resulting from the combination of three components marked with dashed lines. The transformation conducted by *Monotonic Block* is general, not imposing the limits on the ranges of its output values. This means that, in particular, $u_j(0) \in \mathbb{R}$ and $u_j(1) \in \mathbb{R}_{\geq 0}$. To ensure that the results are interpretable, subsequent normalization to the desired range, e.g., $[0, 1]$, is needed.

Function $u(x)$, defined by Eq. (18), can be presented as a neural network with a single input value x. This value is copied L times and passed as the input to the linear layers, where it is scaled by weights y_k and shifted by bias θ_k. Then, the output from the input layer is transformed by the *LeakyHardSigmoid* function and passed to the next linear layer. It must be ensured that the weights in all layers are greater than zero to maintain the function's monotonicity. The weights α_k are initialized with positive values. If during training some value falls below ε being an arbitrarily small positive value, it is set to ε. In what follows, we will refer to the network representing function $u(x)$ as *Monotonic Block* (see Fig. 7). It will be used as a component of the three preference learning methods that are presented in the following subsections.

3.5. ANN-UTADIS: preference learning with UTADIS and ANN

UTADIS is a preference disaggregation method that quantifies a comprehensive quality of each alternative using an additive value function (Zopounidis & Doumpos, 2000):

$$U(a_i) = \sum_{j=1}^{m} w_j u_j(g_j(a_i)), \tag{19}$$

where $u_j \in [0, 1]$ is a marginal value function and w_j is a weight associated with criterion g_j. Function $U(a_i)$ takes values in the [0,1] range, delimited by $U(a^-) = 0$ and $U(a^+) = 1$ for anti-ideal and ideal alternatives, respectively. In UTADIS, u_j is piecewise linear with $n_j(A)$ pre-defined characteristic points x_j^k such that:

$$u_j(x_j^k) \leq u_j(x_j^{k+1}), \ \forall k \in \{1, \ldots, n_j(A) - 1\}, \text{ and } \forall j \in \{1, \ldots, m\}. \tag{20}$$

The marginal values between these points are computed using linear interpolation. In UTADIS, the marginal values $u_j(x_j^k)$ in the characteristic points and weights w_j are determined using mathematical programming based on a set of assignment examples (Zopounidis & Doumpos, 2000). In turn, we will employ ANN for deriving weights and the shape of marginal value functions without having to specify characteristic points. In this way, the method offers greater flexibility in fitting the learning data.

The neural network used by **ANN-UTADIS** is shown in Fig. 8. The performance on each criterion is transformed using *Monotonic*

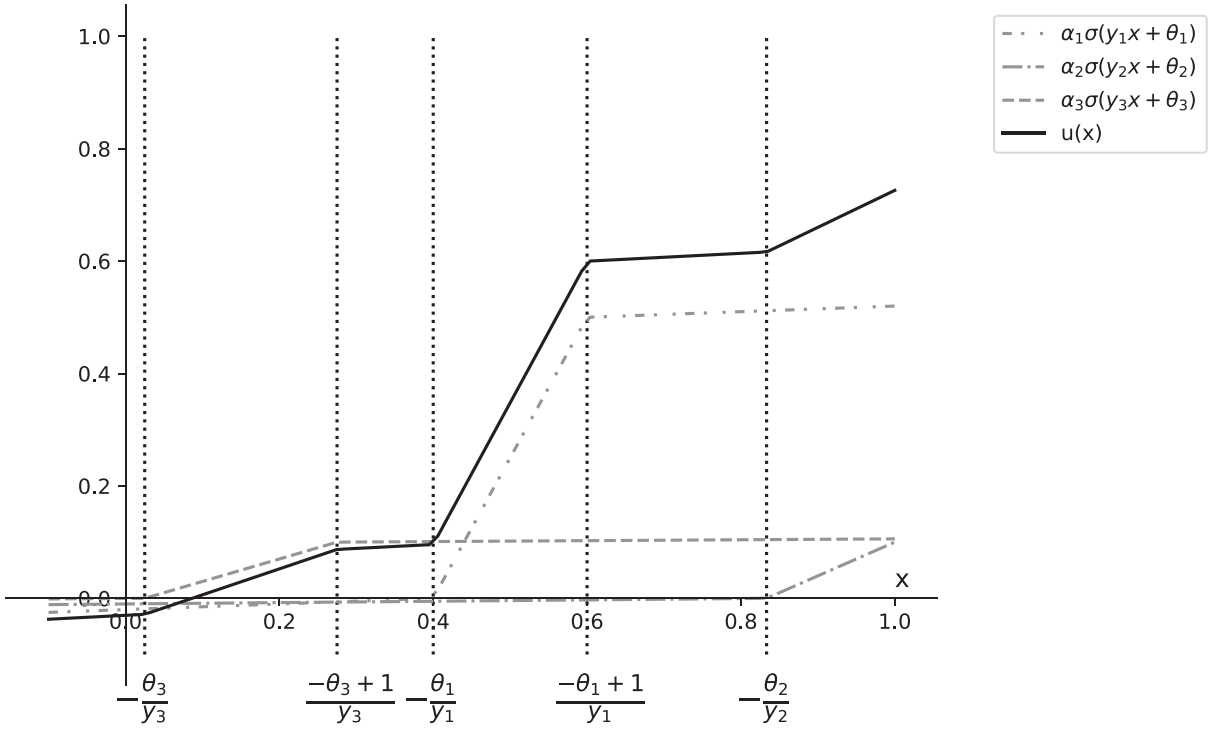


Fig. 6. Function $u(x)$ representing the transformation conducted by the *Monotonic Block* with three ($L = 3$) components.

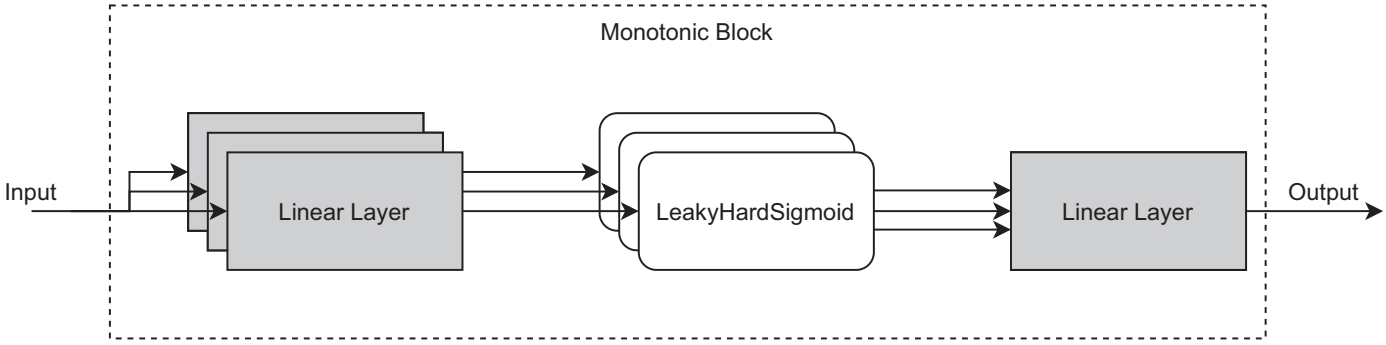


Fig. 7. The *Monotonic Block* used in the preference learning algorithms based on ANNs.

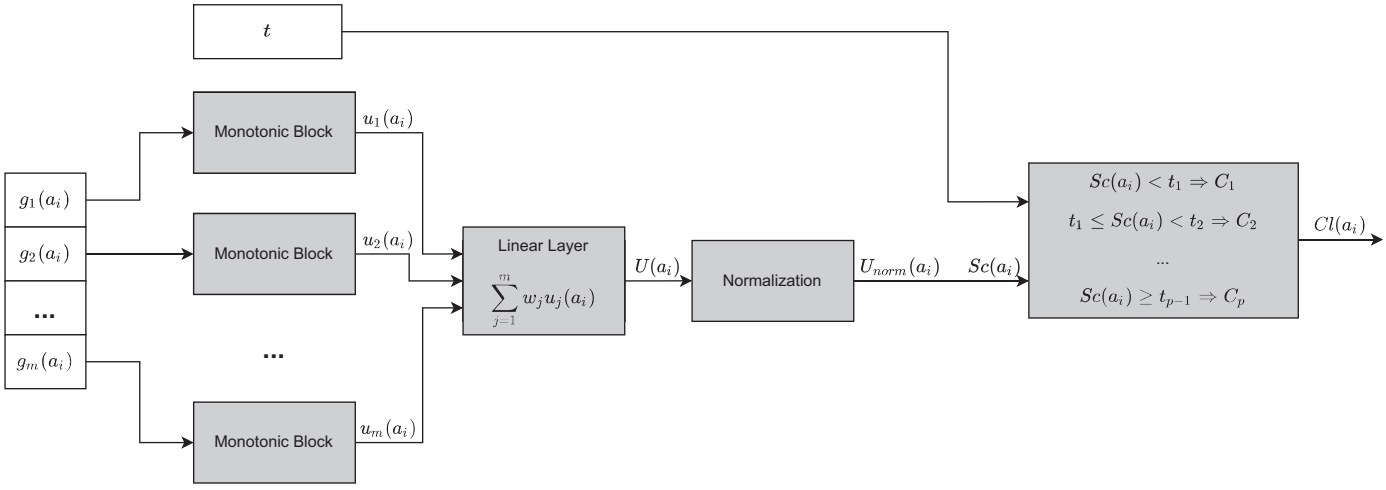


Fig. 8. The architecture of the neural network employed by the ANN-UTADIS method.

Blocks according to Eq. (18), adhering to the monotonicity constraint. To use each *Monotonic Block*, it is required to provide the number L of components. It constrains the maximum number of breakpoints for the resulting function (note that some components may become inactive during optimization, i.e., when $\alpha_k = 0$). Then, the per-criterion marginal values are aggregated into a comprehensive value in line with Eq. (19) by a linear layer. Its weights w_j are constrained to positive values to preserve the pre-defined preference directions. For the sake of normalization, we apply the min-max scaling of comprehensive scores:

$$Sc_{\text{ANN-UTADIS}}(a_i) = \frac{U(a_i) - U(a^-)}{U(a^+) - U(a^-)}. \tag{21}$$

The neural network used by ANN-UTADIS optimizes weights w_j, class thresholds t, and parameters incorporated in the *Monotonic Blocks*. In general, a marginal value function for each criterion may be modeled with a different number L of components. However, we will use the same value of L for all criteria, which allows for a more straightforward parallelization of calculations in Eq. (18) by operations on tenors rather than on individual scalars. Overall, the network for ANN-UTADIS involves one input layer, three hidden layers, and one output layer.

3.6. ANN-PROMETHEE: preference learning with PROMETHEE and ANN

The PROMETHEE method aggregates the results of pairwise comparisons of each alternative against all remaining ones into a comprehensive measure of desirability (Brans & De Smet, 2016). For each pair $(a_i, a_k) \in A \times A$ and each criterion $g_j \in G$, the marginal preference degree is computed as follows:

$$F_j(a_i, a_k) = P_j\big(d_j(a_i, a_k)\big), \tag{22}$$

where P_j is a marginal preference function and $d_j(a_i, a_k) = g_j(a_i) - g_j(a_k)$ is the performance difference. In PROMETHEE, six pre-defined types of P_j are considered. However, the most commonly used is the following:

$$F_j(a_i, a_k) = \begin{cases} 0, & \text{if } d_j(a_i, a_k) \le q_j, \\ \frac{d_j(a_i, a_k) - q_j}{p_j - q_j}, & \text{if } 0 < d_j(a_i, a_k) \le p_j, \\ 1, & \text{if } d_j(a_i, a_k) > p_j, \end{cases} \tag{23}$$

where q_j is an indifference threshold defining the maximal performance difference that is negligible and p_j is a preference threshold specifying the minimal performance difference justifying a strict preference. All preference functions in PROMETHEE are non-decreasing. Also, they are normalized so that $F_j(a_i, a_k) = 0$ for $d_j(a_i, a_k) \le 0$ and their largest value is one. The function type and the respective parameter values for each criterion need to be provided by the DM. The outcomes from the individual criteria are aggregated into a comprehensive preference index $\pi(a_i, a_k)$ using a weighted sum:

$$\pi(a_i, a_k) = \sum_{j=1}^{m} w_j F_j(a_i, a_k), \tag{24}$$

where $w_j \ge 0$ is a weight associated with criterion g_j and $\sum_{j=1}^{m} w_j = 1$. As a result, $\pi(a_i, a_i) = 0$, $a_i \in A$ and $\pi(a^+, a^-) = 1$, where a^+ and a^- are the ideal and anti-ideal alternatives. Such preference degrees are further aggregated into the positive $NFS^+(a_i)$ and negative $NFS^-(a_i)$ flows, using the NFS procedure:

$$NFS^+(a_i) = \frac{1}{n-1} \sum_{k=1}^{n} \pi(a_i, a_k) \text{ and}$$

$$NFS^-(a_i) = \frac{1}{n-1} \sum_{k=1}^{n} \pi(a_k, a_i). \tag{25}$$

The arguments in favour and against alternative a_i are finally aggregated into a comprehensive flow:

$$NFS(a_i) = NFS^+(a_i) - NFS^-(a_i). \tag{26}$$

In the proposed **ANN-PROMETHEE** method, we use monotonic marginal preference functions that are automatically adjusted to the training data, not requiring the specification of type, weights, or comparison thresholds. The architecture of the underlying neural network is presented in Fig. 9. Following the assumptions of PROMETHEE, we first compute the performance differences $d_j(a_i, a_k)$ on each criterion. The negative differences are clipped to zero via the ReLU function:

$$ReLU(x) = \max(x, 0). \tag{27}$$

In this way, the non-positive performance differences will be assigned the same value of the preference index. The values of marginal preference functions F_j are computed using the *Monotonic Block* which ensures both monotonicity and flexibility of shape adjustment:

$$F_j(a_i, a_k) = u_j(\max(d_j(a_i, a_k), 0)). \tag{28}$$

The marginal preference degrees are aggregated into a comprehensive preference index using a linear layer with non-negative weights. Since weights and parameters of the *Monotonic Block* are not constrained from the top, we normalize the comprehensive indices as follows:

$$\pi_{norm}(a_i, a_k) = \frac{\pi(a_i, a_k) - \pi(a^-, a^-)}{\pi(a^+, a^-) - \pi(a^-, a^-)}. \tag{29}$$

Then, the outcomes of pairwise comparisons are aggregated over all alternatives into positive, negative, and comprehensive flows using the Net Flow Score procedure:

$$\begin{aligned} Sc_{\text{ANN-PROMETHEE}}(a_i) &= NFS^+(a_i) - NFS^-(a_i) \\ &= \frac{1}{n-1} \left[\sum_{k=1}^{n} \pi_{norm}(a_i, a_k) - \pi_{norm}(a_k, a_i) \right]. \end{aligned} \tag{30}$$

The use of NFS implies that the preference degrees for all pairs of alternatives need to be computed in a batch. Moreover, similar to the ANN-UTADIS, ANN-PROMETHEE requires specification of the number of components for each *Monotonic Block*. However, it is recommended to use the same number L for all such blocks. Overall, the network for ANN-PROMETHEE involves one input layer, four hidden layers, and one output layer.

3.7. ANN-ELECTRE: preference learning with ELECTRE and ANN

The ELECTRE method compares the alternatives pairwise through an outranking relation (Figueira et al., 2013). In what follows, we discuss its adaptation for scoring the alternatives based on aggregating the sufficiently great outranking credibilities using the NFS procedure. We will consider two tests to compute the credibility for pair $(a_i, a_k) \in A \times A$. The concordance test quantifies the arguments in favor of a_i being at least as good as a_k. The marginal concordance index for criterion g_j is computed as follows:

$$c_j(a_i, a_k) = \begin{cases} 1, & if \;\; g_j(a_i) \ge g_j(a_k) - q_j, \\ \frac{g_j(a_i) + p_j - g_j(a_k)}{p_j - q_j}, & if \;\; g_j(a_i) < g_j(a_k) - q_j \\ & \text{and} \;\; g_j(a_i) \ge g_j(a_k) - p_j, \\ 0, & if \;\; g_j(a_i) < g_j(a_k) - p_j, \end{cases} \tag{31}$$

where q_j and p_j are, respectively, indifference and preference thresholds. Whichever the threshold values, $c_j(a_i, a_k) = 1$ for $g(a_i) \ge g(a_k)$. Moreover, $c_j(a_i, a_k)$ is a monotonic and piecewise

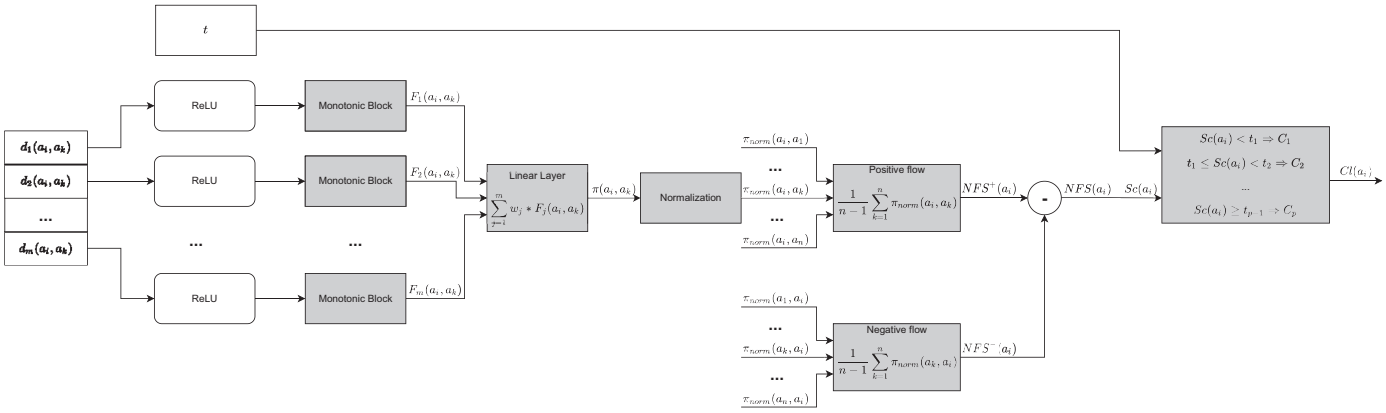


Fig. 9. The architecture of the neural network employed by the ANN-PROMETHEE method.

linear function. The per-criteria results are aggregated into a comprehensive concordance index $C(a_i, a_k)$ using a weighted sum:

$$C(a_i, a_k) = \sum_{j=1}^{m} w_j c_j(a_i, a_k), \tag{32}$$

where w_j is a weight associated with g_j and $\sum_{j=1}^{m} w_j = 1$. Index $C(a_i, a_k)$ is interpreted as the strength of the coalition of criteria supporting the outranking. In turn, the discordance test verifies the strength of arguments against the outranking. In particular, a marginal discordance index is defined as follows:

$$D_j(a_i, a_k) = \begin{cases} 1, & if \;\; g_j(a_i) \le g_j(a_k) - v_j, \\ \frac{g_j(a_k) - p_j - g_j(a_i)}{v_j - p_j}, & if \;\; g_j(a_i) > g_j(a_k) - v_j \\ & \text{and} \;\; g_j(a_i) \le g_j(a_k) - p_j, \\ 0, & if \;\; g_j(a_i) > g_j(a_k) - p_j, \end{cases} \tag{33}$$

where v_j is a veto threshold interpreted as the minimal performance difference implying a complete discordance. The thresholds need to respect the following constraints: $0 \le q_j \le p_j < v_j$. Note that the discordance effect does not to be considered for all $g_j \in G$ because the power to veto against the outranking is usually attributed only to the most important criteria. We consider the aggregation of partial discordances into a comprehensive one using the following function (Mousseau & Dias, 2004):

$$D(a_i, a_k) = 1 - \max_{j=1,\dots,m} D_j(a_i, a_k). \tag{34}$$

Hence the maximal partial discordance over all criteria decides upon the comprehensive strength of arguments against the hypothesis that a_i outranks a_k. Finally, the credibility degree is computed by multiplying the comprehensive concordance and discordance:

$$\sigma(a_i, a_k) = C(a_i, a_k) \cdot D(a_i, a_k). \tag{35}$$

Thus the greater the arguments in favor and the lesser the arguments against the outranking, the greater the credibility. To compute the score for each alternative, we will consider only sufficiently great credibilities to avoid compensation between marginal arguments in favor or against a_i being a favorable alternative. Specifically, we will consider only $\sigma(a_i, a_k)$ which are at least as good as cutting level λ such that $0.5 \le \lambda \le 1$. Finally, similar to the PROMETHEE method, we compute the Net Flow Score for each alternative $a_i \in A$:

$$\begin{aligned} NFS(a_i) &= NFS^{+}(a_i) - NFS^{-}(a_i) \\ &= \frac{1}{n-1}\left[\sum_{k=1}^{n} \sigma_{NFS}(a_i, a_k) - \sigma_{NFS}(a_k, a_i)\right], \end{aligned} \tag{36}$$

where $\sigma_{NFS}(a_i, a_k) = \sigma(a_i, a_k) - \lambda$ if $\sigma(a_i, a_k) \ge \lambda$ and $\sigma_{NFS}(a_i, a_k) = 0$, otherwise. Note that other realizations of $\sigma_{NFS}(a_i, a_k)$ would also be possible. However, we opted for a variant that keeps the spirit of ELECTRE while being intuitively useful in computing comprehensive scores of alternatives via NFS.

In the proposed **ANN-ELECTRE**, we avoid direct specification of thresholds (q_j, p_j and v_j), weights w_j, and cutting level λ. In turn, the parameters of an outranking-based sorting model are inferred indirectly using the neural network whose architecture is presented in Fig. 10. In the preprocessing phase, ANN computes the performance differences. Then, the calculations are split into two parts responsible for conducting the concordance and discordance tests. These parts share the value of preference thresholds p_j, $j = 1, \dots, m$, to prevent the simultaneous occurrence of concordance and discordance. These thresholds are optimized when training the ANN while ensuring that $p_j \in [0, 1]$.

In part responsible for the concordance test, the performance differences are truncated to positive values by the ReLU function (see Eq. (27)), and their order is reversed by subtracting them from one. Since the performances on individual criteria are normalized in the [0,1] range, after the above transformation, we will get one (corresponding to the maximal value of the concordance index) if $g_j(a_i) \ge g_j(a_k)$, or a value in the [0,1] range, otherwise. The obtained value is processed by the marginal concordance function u_j^c implemented by *Monotonic Block*, allowing for its monotonic and flexible transformation as depicted in Fig. 11(a). The marginal concordance should be zero if the performance difference exceeds the preference threshold p_j. This can be attained by subtracting the value of u_j^c attained for $1 - p_j$, i.e., $u_j^c(1 - p_j)$ from $u_j^c(1 - ReLU(g_j(a_k) - g_j(a_i)))$. The resulting difference should be truncated to positive values, e.g., using the ReLU function. However, the lack of a gradient for the negative arguments of this function makes it difficult to optimize values of preference thresholds p_j, $j = 1, \dots, m$. For this reason, we use the LeakyReLU function instead which has a non-zero gradient for negative values equal to δ:

$$LeakyReLU(x) = max(x, \delta x), \tag{37}$$

where δ is a slope angle for the negative part of the function. It should take a small value and can be minimized to zero during optimization. The result of these operations is shown in Fig. 11(b). Overall, the marginal concordance index $c_j(a_i, a_k)$ is computed as follows:

$$\begin{aligned} c_j(a_i, a_k) = LeakyReLU_p\big(u_j^c\big(1 - ReLU(g_j(a_k) - g_j(a_i))\big) \\ - u_j^c(1 - p_j)\big). \end{aligned} \tag{38}$$

Comprehensive concordance index $C(a_i, a_k)$ is calculated using Eq. (32) by a linear layer that incorporates criteria weights $w_j \ge 0$.

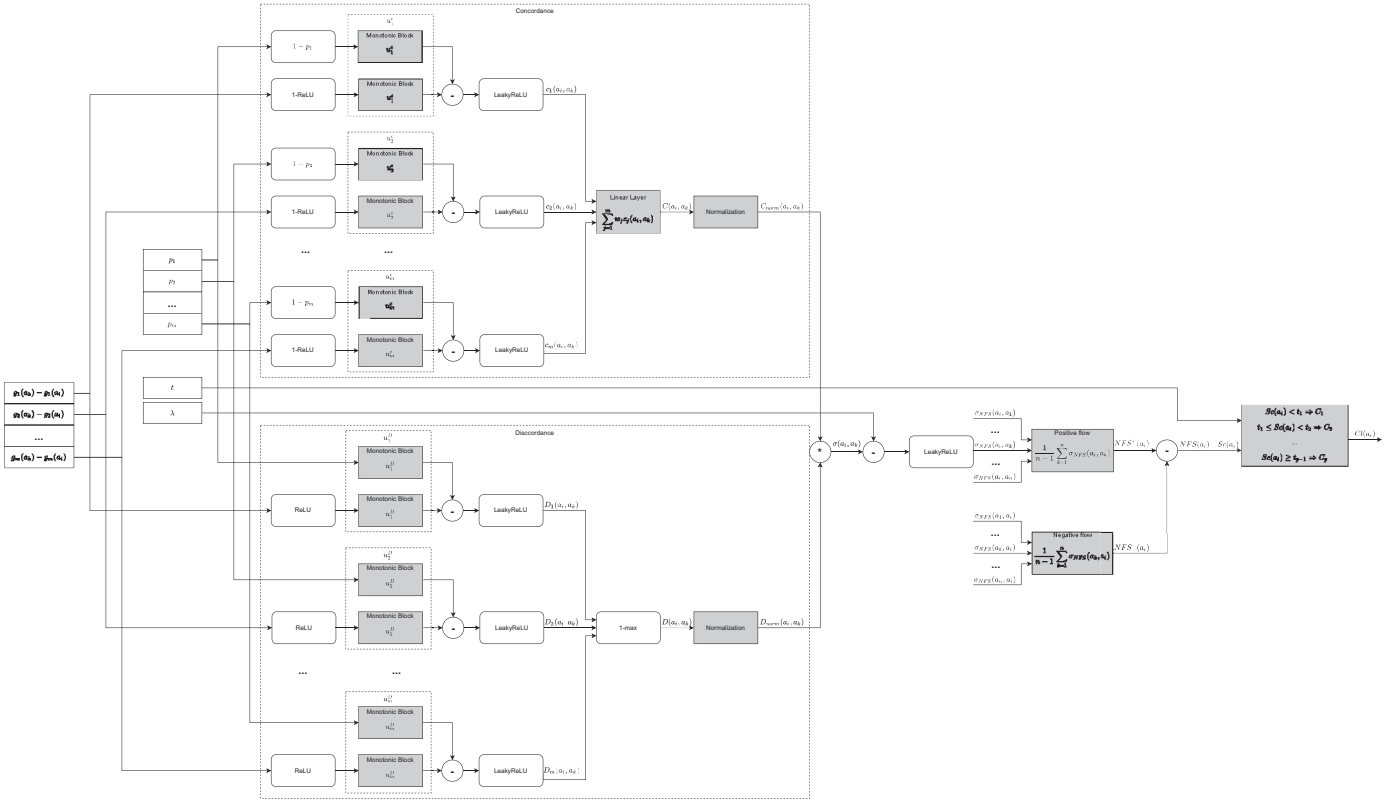
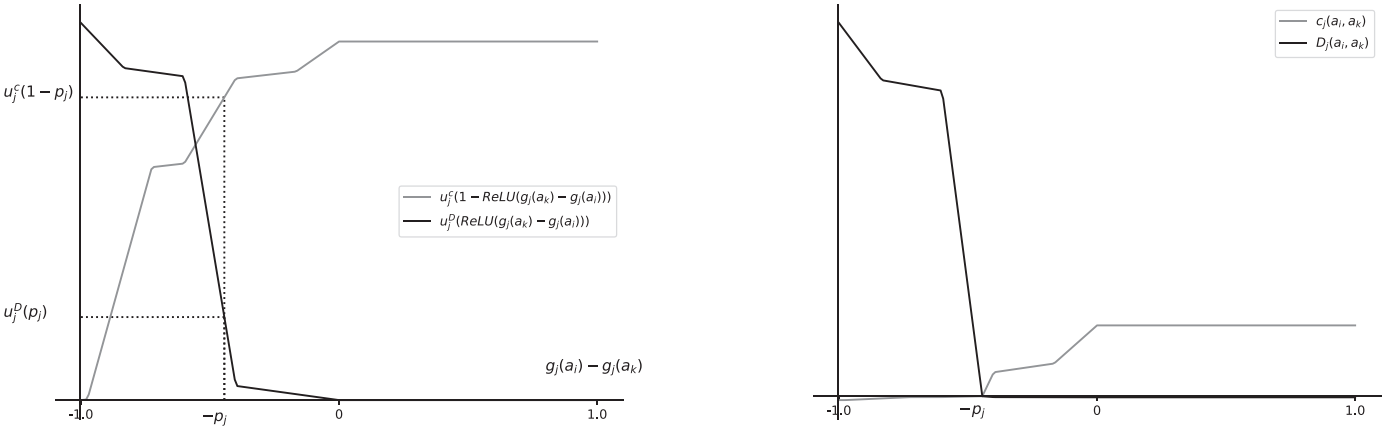


Fig. 10. The architecture of the neural network employed by the ANN-ELECTRE method.



(a) intermediate functions (b) final functions

Fig. 11. The marginal concordance and discordance functions for the ANN-ELECTRE method before (a) and after (b) subtracting the value attained for preference threshold p_j and after transformation by LeakyReLU with $\delta = 0.01$.

Finally, values of $C(a_i, a_k)$ are normalized to the [0,1] range, using the min-max scaling:

$$C_{norm}(a_i, a_k) = \frac{C(a_i, a_k) - C(a^-, a^+)}{C(a^+, a^-) - C(a^-, a^+)}. \tag{39}$$

The other part of the ANN-ELECTRE network is responsible for conducting the discordance test. It first truncates the performance differences to positive values, i.e., these for which $g_j(a_k) \geq g_j(a_i)$. Then, the result of such an operation is processed by function u_j^D modeled by the *Monotonic Block* to obtain marginal discordance index (see Fig. 11a). To account for the preference threshold p_j and reduce the discordances to zero for performance differences below this threshold, we subtract the value of u_j^D attained for p_j, i.e., $u_j^D(p_j)$, from $u_j^D(ReLU(g_j(a_k) - g_j(a_i)))$. Finally, the resulting difference is processed using the LeakyReLU function (see Fig. 11b) in the following way:

$$D_j(a_i, a_k) = LeakyReLU\left(u_j^D(ReLU(g_j(a_k) - g_j(a_i))) - u_j^D(p_j)\right). \tag{40}$$

Comprehensive discordance index $D(a_i, a_k)$ is computed in line with Eq. (34) and normalized to the [0,1] range:

$$D_{norm}(a_i, a_k) = \frac{D(a_i, a_k) - D(a^+, a^-)}{D(a^-, a^+) - D(a^+, a^-)}. \quad (41)$$

Overall, the largest value of the marginal discordance is one, which allows the method to adjust the test in such a way that the discordance is not necessarily modeled on all criteria.

The results from the two parts of ANN responsible for the concordance and discordance tests are combined into the outranking credibility $\sigma(a_i, a_k)$ using Eq. (35) in the form of a multiplication layer. To consider only sufficiently great credibilities, we should decrease them by cutting level λ and transform the resulting negative values to zero. However, since cutting level λ is a parameter learned during training, to allow for its more efficient optimization, we decided to transform the negative credibilities to values close to zero using the LeakyReLU function with a very small δ equal to 0.001:

$$\sigma_{NFS}(a_i, a_k) = LeakyReLU(\sigma(a_i, a_k) - \lambda). \quad (42)$$

The positive and negative flows as well as comprehensive scores, denoted by $Sc_{ANN-ELECTRE}(a_i)$, for all alternatives $a_i \in A$ are computed in line with Eq. (36).

The hyperparameters of ANN-ELECTRE are the slope values δ for the LeakyReLU function and the number L of components for *Monotonic Blocks*. Similar to the previously discussed methods, to speed up the optimization process, we use the same value of L for all criteria in the concordance and discordance parts of the network. Overall, the network for ANN-ELECTRE involves one input layer, five hidden layers, and one output layer. Hence its architecture involves the greatest number of layers and units among all introduced methods.

4. Optimization

In this section, we discuss the process of determining parameter values for the presented sorting models along with all the supporting techniques that accelerate this process. The role of optimization is to determine an optimal model highly consistent with the supplied/available assignment examples. Due to nonlinear transformations, numerous relationships between values of different parameters, and a large number of objects to be scored (particularly for methods based on pairwise comparisons), the use of contemporary mathematical programming solvers is excluded because of their insufficient efficiency. Therefore, to determine the values of model parameters, we use the iterative optimization methods based on Gradient Descent (GD). There are many different techniques, called optimizers, used in ANN that are based on GD. In this paper, we employ AdamW, which is the Adam optimizer (Kingma & Ba, 2014) with decoupled weight decay regularization (Loshchilov & Hutter, 2018).

The AdamW optimizer employs the following hyperparameters having a significant impact on the training process, speed, and quality of an identified solution:

- α – a learning rate that affects the size of the parameter correction in an optimization step. Too low values imply slow learning and the possibility of getting stuck in the local optimum too early, while too high values make it possible to omit the optimum and prevent the optimization from converging.
- β_1 and β_2 – momentum factors determining the impact of historical improvement of parameters on the current step. Momentum is used to speed up and improve the optimization process by drawing conclusions from previous steps to determine a more stable optimization direction and less dynamic response to perturbations during training.
- ϵ – a small value added to the denominator to stabilize the calculations.
- w_τ – a weight decay factor.

The entire optimization process is presented as Algorithm 1.

Algorithm 1 Optimization algorithm using AdamW (adapted after Loshchilov & Hutter, 2018).

1: **given** $\alpha \in (0, 1)$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, $w_\tau = 0.01$, $\xi \in (0, 1)$
2: **initialize** epoch number $\tau \leftarrow 0$, parameter vector $\mathbf{x}_{\tau=0} \in \mathbb{R}^n$, first moment vector $\mathbf{m}_{\tau=0} \leftarrow 0$, second moment vector $\mathbf{v}_{\tau=0} \leftarrow 0$
3: $evaluations \leftarrow g(A^R)$
4: $input \leftarrow Preprocesing(evaluations)$
5: **repeat**
6: $\quad \tau \leftarrow \tau + 1$
7: $\quad input_{noised} \leftarrow input + \mathcal{N}(0, \xi)$
8: $\quad \mathbf{g}_\tau \leftarrow \nabla Loss(Sc(\mathbf{x}_{\tau-1}, input_{noised}))$
9: $\quad \mathbf{m}_\tau \leftarrow \beta_1 \mathbf{m}_{\tau-1} + (1 - \beta_1)\mathbf{g}_\tau$
10: $\quad \mathbf{v}_\tau \leftarrow \beta_2 \mathbf{v}_{\tau-1} + (1 - \beta_2)\mathbf{g}_\tau^2$
11: $\quad \hat{\mathbf{m}}_\tau \leftarrow \mathbf{m}_\tau / (1 - \beta_1^\tau)$
12: $\quad \hat{\mathbf{v}}_\tau \leftarrow \mathbf{v}_\tau / (1 - \beta_2^\tau)$
13: $\quad \mathbf{x}_\tau \leftarrow \mathbf{x}_{\tau-1} - (\alpha \hat{\mathbf{m}}_\tau / (\sqrt{\hat{\mathbf{v}}_\tau} + \epsilon) + w_\tau \mathbf{x}_{\tau-1})$
14: **until** stopping criterion is met

First, all parameter values $\mathbf{x}_{\tau=0}$ are initialized randomly according to the constraints imposed on specific parameter types. These parameters can be, e.g., weights w_j, interaction coefficients $w_{\{j,l\}}$ and class thresholds t for the ANN-Ch methods, whereas for ANN-ELECTRE – these are α_k, y_k, θ_k from each *Monotonic Block*, weights w_j, preference thresholds p_j, cutting level λ, and class thresholds t. At the same time, all auxiliary variables for the optimization process, including an epoch number and moment vectors, are initialized (see line 2).

We used two optimization techniques aimed at accelerating optimization. The first one is Batch Gradient Descent (BGD), which calculates loss, gradient, and modifications of network parameter values at once after processing all alternatives in A^R (see line 3). It speeds up the entire optimization process and makes the final model independent from the order of processing the alternatives. If it is impossible to use BGD, it is recommended to employ Mini Batch Gradient Descent (Ruder, 2016). This technique divides the training set into subsets in each epoch and trains this subset at once. In this case, the order of processing alternatives may affect the final result, but this impact will be negligible with sufficiently large batches.

The other method for reducing processing time is to prepare the input data in the preprocessing stage so that only operations using network parameters are performed in each epoch (see line 4). For example, one assumes that the entry gets alternatives with performances converted to the 0–1 range via min-max scaling.

After the input data preprocessing stage, the actual optimization process takes place. It consists of the iterative improvement of the model parameters to minimize a comprehensive classification error. To increase the noise resistance, robustness of the model, and its generalization capabilities, we used data augmentation (Zheng, Song, Leung, & Goodfellow, 2016). It is a technique mainly used to reduce overfitting (Shorten & Khoshgoftaar, 2019). It is about creating new training objects from the transformations of the original objects. The basic change is to add noise, e.g., in the form of Gaussian noise $\mathcal{N}(0, \xi)$, where ξ is the standard deviation, being an additional hyperparameter of the optimization process. Its application implies a slight change in alternatives performances, different in each epoch (see line 7).

Table 1
Values of criteria weights obtained for the ANN-based methods for the illustrative example concerning the ERA dataset.

Method	w_1	w_2	w_3	w_4
ANN-OWA	0.4257	0.0055	0.2225	0.3464
ANN-Ch-Constr.	0.0693	0.0433	0.0000	0.0255
ANN-Ch-Uncons.	0.0030	0.0039	0.0018	−0.0029
ANN-Ch-Pos.	0.0060	0.0048	0.0021	0.0003
ANN-TOPSIS	0.5799	0.7987	0.6676	0.5701
ANN-UTADIS	0.3251	0.1663	0.4217	0.0869
ANN-PROMETHEE	0.2126	0.4573	0.1591	0.1709
ANN-ELECTRE (concordance)	0.5139	0.1955	0.2726	0.0180
ANN-ELECTRE (discordance)	0.3029	0.0000	1.0000	0.2110

By propagating the input with noise through the successive layers of the network in iteration τ with the current parameter values $x_{\tau-1}$, scores Sc are computed for all reference alternatives A^R. The resulting class assignments are compared with the desired ones, and the respective loss is computed. Then, the loss is backpropagated across all network layers, leading to gradient vectors g_τ (see line 8). Subsequently, gradient transformation is performed for each parameter to improve the optimization process (see lines 9–12). The AdamW algorithm employs an adaptive learning rate for each method parameter, using squared gradients to scale the learning rate and moving momentum average.

Finally, a new parameter value x_τ is computed by combining the current value with the identified correction. For this purpose, AdamW considers the previously prepared auxiliary variables, learning rate, and weight decay (see line 13). The latter parameter controls the model's regularization, imposing an additional optimization goal that prevents the construction of accurate, though incorrect, solutions with poor generalization capabilities. This may occur in the case of overfitting the model for the training data or assigning parameter values that are hard to interpret (in the case of ANNs, these are usually prohibitively large values). The weight decay mechanism adds a penalty, controlled by w_τ, for the value of the parameters in each optimization step.

Processing all alternatives and modifying the parameter values is called an epoch. Such a process is performed multiple times until the stopping condition is met. In our case, it occurs after 200 training epochs. The final parameter values are those for which the model obtained the lowest error for the validation set during optimization.

5. Illustration of preference models inferred with neural networks

In this section, we illustrate the preference models inferred with the proposed ANN-based methods. We consider a two-class problem called ERA (Employee Rejection / Acceptance) (Hall et al., 2009). It concerns a student survey regarding the willingness to hire an employee based on four features of a candidate, such as, e.g., experience and verbal skills. All criteria are of gain type and have been pre-processed as described in Section 4. The models were obtained by training the methods on 80% randomly chosen alternatives. The criteria weights obtained for all methods are presented in Table 1, whereas the interaction coefficients for the ANN-Ch algorithms are given in Table 2.

ANN-OWA By applying the ANN-OWA method, we obtained a model parameterized with the weights shown in Table 1. They reflect the impact of each position in the sorted performance vector on the comprehensive score and assignment of each alternative. The highest performance on any criterion has the greatest impact on the results (almost 43%), and the lowest performance is the second most important factor (almost 35%). In contrast, the second-best performance has a negligible impact on the recommended assignment (below 1%). The two classes considered in the ERA problem are separated by threshold $t_1 = 0.4114$ with OWA taking values between 0 and 1.

In what follows, we provide the models derived with different variants of the Choquet integral-based algorithms. Unlike in the ANN-OWA method, the weights from the linear layer correspond to the weights of individual criteria and interaction coefficient for pairs of criteria.

ANN-Ch-Constr Let us first consider the variant in which the criteria weights need to be positive, interactions can be either positive or negative, but the negative interaction coefficients cannot be greater than the weights of the criteria involved in a given pair. The analysis of weights (see Table 1) indicates that the greatest impact is attributed to the first criterion, whereas the third criterion has the least influence on the attained score. The values of the interaction coefficients are given in Table 2. All coefficients but $w_{\{1,4\}}$ are positive. The greatest synergy effect is observed for g_1 and g_2 as well as g_3 and g_4. This means that the simultaneous presence of highly preferred performances on these criteria pairs gives the alternative a bonus. Note that the weights retain the required dependencies and fulfill the constrain defined by Eq. (10) (e.g., $w_1 + w_{\{1,4\}} = 0.0693 + -0.0255 \geq 0$).

The actual significance of criterion g_i in the Choquet integral can be represented by the Shapley value $\varphi(i)$ defined as follows (Angilella et al., 2013):

$$\varphi(i) = w_i + \sum_{\{i,l\} \subseteq G} \frac{w_{\{i,l\}}}{2}. \tag{43}$$

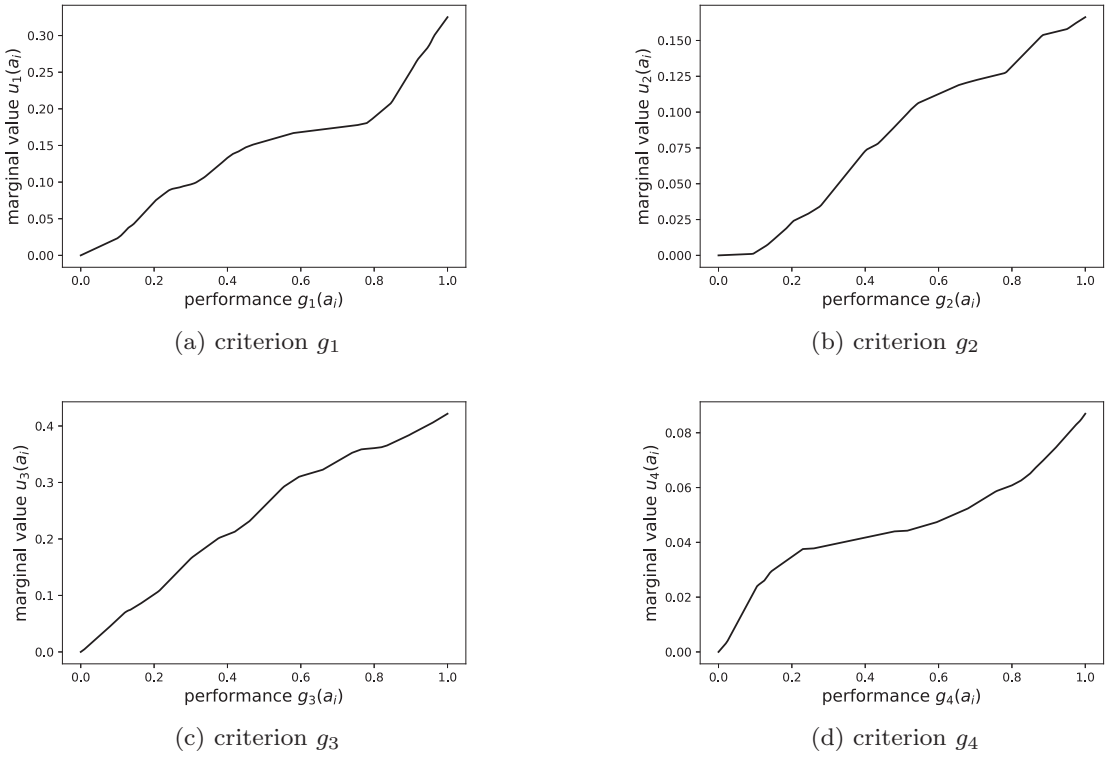
For the considered model, we obtained the following coefficients: $\varphi(1) = 0.2454$, $\varphi(2) = 0.3409$, $\varphi(3) = 0.2148$, and $\varphi(4) = 0.1989$. They indicate that g_1 and g_4 are the most and the least important criteria, respectively. In addition, g_3 has a relatively high significance level $\varphi(3)$ despite its zero weight w_3. However, it is involved in multiple interacting pairs of criteria. Finally, the separating class threshold is $t_1 = 0.6117$.

ANN-Ch-Uncons For the variant of ANN-Ch that considers both positive and negative interactions, while allowing for a change in the direction of preference for a given criterion, the results are quite different. Based on the inferred weights (see Table 1), we conclude that g_2 and g_3 have, respectively, the greatest and the least individual impacts on the attained scores. Moreover, g_4 is assigned a negative weight, meaning that preference learning led to the inversion of preference direction from gain to cost for this criterion. This may indicate possible inconsistencies in the data or suggest the need for incorporating additional constraints in the model. The interaction coefficients for all pairs of criteria are shown in Table 2. Pair $\{g_1, g_2\}$ has the greatest positive impact on the attained score, giving a great bonus to alternatives with high performances on both g_1 and g_2. On the other extreme, $w_{\{1,3\}}$ is very low, implying that the benefit from the coexistence of high values on g_1 and g_3 is marginal. Furthermore, negative interactions can be observed for $\{g_1, g_4\}$ and $\{g_2, g_4\}$. This suggests that it is beneficial for alternatives to have a low value on at least one criterion in these two pairs, which most likely relates to g_4, whose individual weight was already negative. The value of bias is 0.4486, serving to shift the sigmoid function and having no direct interpretation. In this case, the value of a separating class threshold is $t_1 = 0.6117$.

ANN-Ch-Pos The last variant of ANN-Ch assumed that both individual weights and interaction coefficients need to be positive. This excludes, e.g., negating the preference direction of g_4, as suggested by the previous model. The analysis of weights (see Table 1) indicates the g_1 and g_2 are the most important criteria, whereas the impact of g_4 is negligible. The crucial role of the first two criteria is emphasized by the highest value of the interaction coefficient for this pair. On the other extreme, g_3 and g_4 are not interacting,

Table 2
Values of criteria interaction coefficients obtained for the ANN-based methods using the Choquet integral for the illustrative example concerning the ERA dataset.

Method	$w_{\{1,2\}}$	$w_{\{1,3\}}$	$w_{\{1,4\}}$	$w_{\{2,3\}}$	$w_{\{2,4\}}$	$w_{\{3,4\}}$
ANN-Ch-Constr.	0.2859	0.0919	−0.0255	0.1374	0.1720	0.2003
ANN-Ch-Uncons.	0.0042	0.0002	−0.0007	0.0022	−0.0021	0.0006
ANN-Ch-Pos.	0.0043	0.0012	0.0008	0.0030	0.0006	0.0000



(a) criterion g_1 (b) criterion g_2

(c) criterion g_3 (d) criterion g_4

Fig. 12. Marginal value functions scaled by criteria weights constructed by ANN-UTADIS for the ERA dataset.

meaning that the coexistence of high or low values on these criteria has no impact on the attained score. The precise value of interaction coefficients are shown in Table 2. All above weights translate into the following normalized Shapley values: $\varphi(1) = 0.3962$, $\varphi(2) = 0.3794$, $\varphi(3) = 0.1819$, and $\varphi(4) = 0.0425$. They confirm that g_1 and g_2 are the most influential criteria, whereas the role of g_4 is negligible. The threshold separating the two considered classes on a scale of the Choquet integral is $t_1 = 0.6173$.

ANN-TOPSIS TOPSIS investigates the distance of each alternative from the ideal and anti-ideal alternatives. For the considered problem, the performances of these alternatives are as follows: $a^+ = [1, 1, 1, 1]$ and $a^- = [0, 0, 0, 0]$. Criterion g_2 has the greatest impact on the distances, whereas the influence of g_4 is the least (see Table 1). However, the ratios between the criteria weights are much lesser than in the case of the Choquet integral-based models, meaning that in TOPSIS, the importances of all criteria are more balanced. The threshold separating the less and more preferred classes on the considered distance scale from 0 to 1 is $t_1 = 0.4601$.

ANN-UTADIS The value-based model inferred by ANN-UTADIS consists of marginal value functions for all criteria. Their shapes can be visualized based on the characteristic points of the *Monotonic Blocks*, weights for the linear layer aggregating marginal values, and the normalization constraint for the weights. We used 20 component functions (neurons in the hidden layer) in each of the *Monotonic Blocks*. Thus the constructed functions can have up to 40 characteristic points. The plots can be reconstructed by querying relevant parts of the ANN for the value assigned to artificially generated input data.

The marginal value functions are shown in Fig. 12. The greatest maximal share in the comprehensive value is assigned to g_3, whereas the lowest maximal share corresponds to g_4 (see Table 1). The marginal function for g_1 reveals minor differences for the performances ranging from 0.6 to 0.8. In contrast, above 0.8, there is a rapid increase in marginal values, indicating a high preference for alternatives with the most preferred values on g_1. For g_2, the marginal values assigned to performances lesser than 0.1 are close to zero. Above this level, the function's shape, similar to the function corresponding to g_3, is nearly linear. In turn, for g_4, the differences between the marginal values are significant for very low or very high performances, whereas the slope is less steep in the mid-range. The threshold separating the two classes on a scale of a comprehensive value from 0 to 1 is $t_1 = 0.4909$.

ANN-PROMETHEE In the PROMETHEE-based method, the parameter values of the network refer to pairwise comparisons of alternatives, providing evidence on how much one of them is preferred to the other. In this case, we used 20 component functions in each of the *Monotonic Blocks* and reconstructed the marginal preference functions similarly as for ANN-UTADIS. The plots in Fig. 13 are already scaled by the criteria weights.

The weight of g_2 is the greatest, whereas the importance coefficients of g_3 and g_4 are much lesser (see Table 1). For all criteria and the non-positive performance differences, preference degrees are zero. Moreover, a small advantage of one alternative over another does not imply the preference or the preference degree is very marginal. For example, for g_4 – the preference functions starts to increase for difference greater than 0.12. Hence this value can be

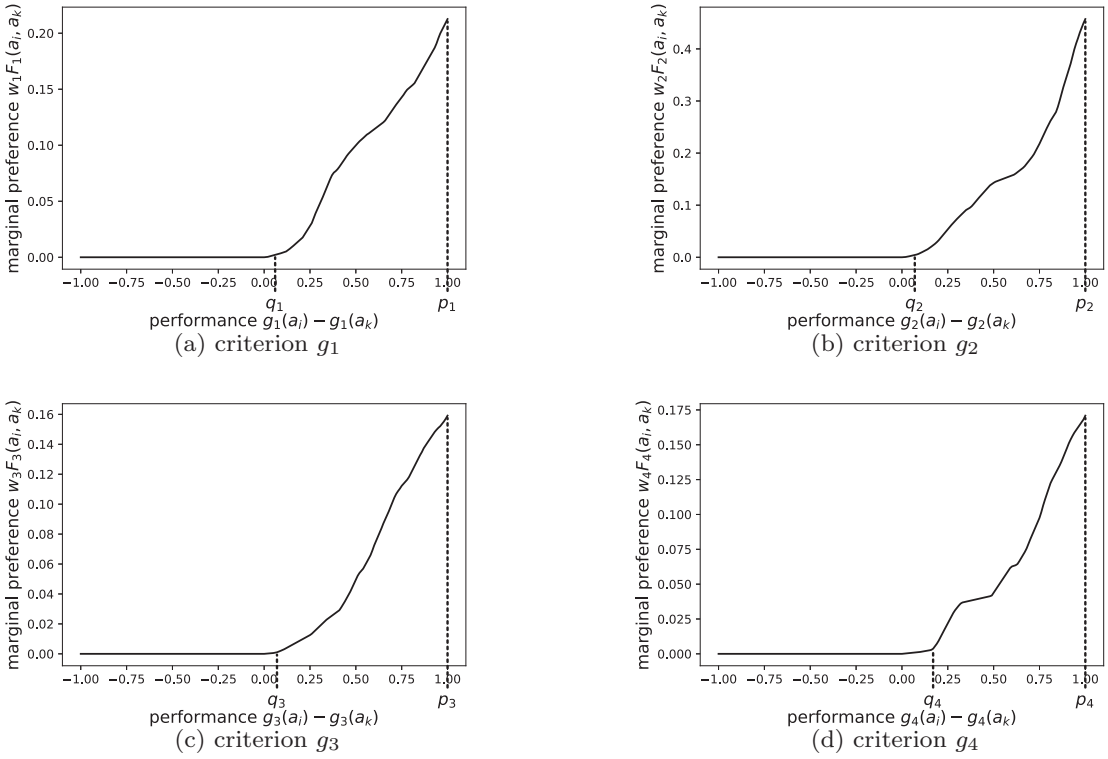


Fig. 13. Marginal preference functions scaled by criteria weights constructed by ANN-PROMETHEE for the ERA dataset.

interpreted as an indifference threshold. Furthermore, we do not observe any level (constant) part above a certain value. Once the function starts to increase, this trend is maintained till the very end. Hence the preference threshold for all criteria is equal to one. The plots show that the greatest increase in the preference degree occurs for the largest differences (> 0.75), but for g_1 and g_4, such a steep slope is also observed for differences between 0.2 to 0.3. The threshold separating the two classes on a Net Flow Score scale from -1 to 1 is $t_1 = 0$.

ANN-ELECTRE For the ELECTRE-like method, we analyze the concordance and discordance functions for each criterion. In this approach, 30 components were used in each of the *Monotonic Blocks*. However, as can be seen in Fig. 14, most of them were deactivated during training. This led to easily interpretable functions with clearly distinguished thresholds for the performance difference, implying the maximal value of either concordance or discordance.

The marginal concordance and discordance functions presented in Fig. 14 were already normalized. Moreover, the concordance functions were scaled by the weights (see Table 1). For ANN-ELECTRE, there is no univocal information on the importance of different criteria because the methods assigned different weights to the arguments in favor and against the outranking deriving from the same criterion. On the one hand, g_1 has the greatest impact in terms of supporting the truth of outranking, whereas the concordance weight of g_4 is the least. On the other hand, g_3 may have a very negative impact by strongly supporting discordance in case of large performance differences against the outranking. The maximal discordance on g_3 is one, hence zeroing the outranking credibility in case one alternative is vastly worse than another on this criterion. Furthermore, the discordance does not occur for g_2, which can be interpreted as the lack of power of this criterion to veto against the outranking.

When it comes to the marginal functions, for performance differences greater or equal to zero, the concordance indices take the maximal value of one (if the plot is unscaled) or concordance weight assigned to a given criterion (when considering a scaled plot as depicted in Fig. 14). For all criteria, an indifference threshold is close to zero. It is also possible to distinguish the preference and veto thresholds. When the performance difference exceeds the negated preference threshold, the concordance becomes positive, whereas if the performance difference is lesser than this threshold, the discordance occurs (when veto is admitted for a given criterion). For g_3, this threshold has a value of 0.2236. In turn, for g_1, there is a large zone with no or very marginal concordance and discordance. The concordance becomes positive for marginally negative performance differences, whereas the discordance is above zero only when one alternative is worse than another by at least $p_1 = 0.5940$. The preference thresholds directly optimized by the ANN for g_2 and g_4 are 0.3329 and 0.3354. Finally, when the performance difference is greater than the veto threshold, the maximal discordance on a given criterion occurs. The values of this threshold for g_1, g_3, and g_4 are, respectively, around 0.93, 0.38, and 0.61.

An important parameter inferred by ANN-ELECTRE is the cutting level λ. It was assigned a very high value of 0.95. This means that the arguments supporting the outranking need to be very strong, and the arguments against the outranking need to be none or negligible to support the inclusion of credibility in the Net Flow Score computations performed by the method. The threshold separating the rejection and acceptance classes on the scale between -1 and 1 is $t_1 = 0$.

6. Computational experiments

To investigate the performance of the proposed methods, they were applied to a set of binary sorting problems (see Table 3). The datasets come from the UCI repository (http://archive.ics.uci.edu/ml/) and the WEKA software (Hall et al., 2009). The number of criteria is between four and eight, whereas the number of alternatives ranges from several dozen to several hundred. In Table 3,

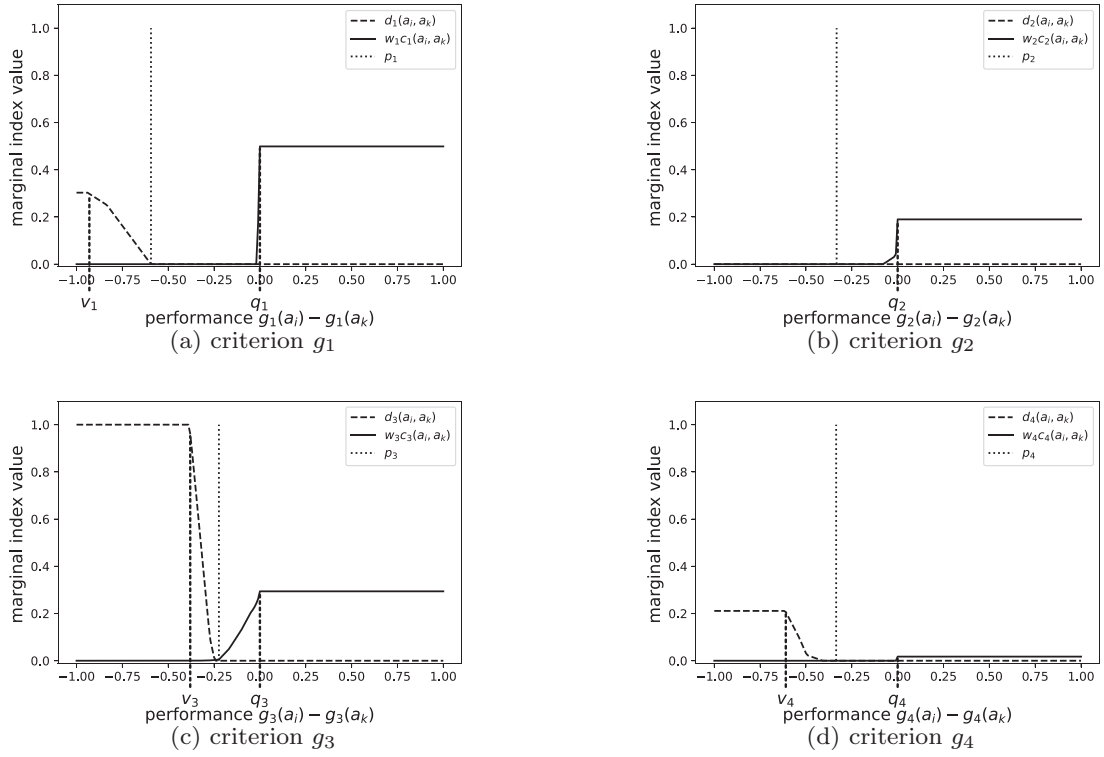


Fig. 14. Marginal concordance and discordance functions constructed by ANN-ELECTRE for the ERA dataset.

Table 3
Datasets considered in the experimental evaluation.

Name	Code	No. of alternatives	No. of criteria	No. of pairwise comparisons
Den Bosch	DBS	120	8	14,280
Computer Processing Units	CPU	209	6	43,472
Breast Cancer	BCC	286	7	81,510
Auto MPG	MPG	392	7	153,272
Employee Selection	ESL	488	4	237,656
Mammographic	MMG	961	5	922,560
Employee Rejection/Acceptance	ERA	1000	4	999,000
Lecturers Evaluation	LEV	1000	4	999,000
Car Evaluation	CEV	1728	6	2,984,256

we include the information on the number of pairwise comparisons that appear as input in outranking-based approaches such as ANN-PROMETHEE and ANN-ELECTRE.

The same set of problems was considered in Tehrani et al. (2012) and Sobrie et al. (2019). For a detailed description of each set, see Tehrani et al. (2012). Some of them (MPG, MMG, and BCC) involve nominal attributes that have been transformed into monotonic criteria according to Tehrani et al. (2012). This increases the difficulty of the preference learning task for such problems as the methods respecting the pre-defined preference directions need to deal with an arbitrarily imposed order which reduces their flexibility in fitting the model.

To quantify the algorithms' performance, we use two classification quality measures. The first one is a standard misclassification error (0/1 loss), referring to the number of alternatives in $A^C \subseteq A$ that the inferred model classifies incorrectly:

$$0/1 \;\; loss = \frac{1}{|A_C|} \sum_{a_i \in A_C} \text{CL error}(a_i), \tag{44}$$

where:

$$\text{CL error}(a_i) = \begin{cases} 1, & \text{if } \; Sc(a_i) < t_{C_{DM}(a_i)}, \;\; \textit{or} \;\; Sc(a_i) \geq t_{C_{DM}(a_i)+1}, \\ 0, & \text{otherwise.} \end{cases} \tag{45}$$

The other measure is AUC, which – for a binary classification involving classes C_1 and C_2 – takes the following form:

$$AUC = \frac{\sum_{a_i \in A_{C_1}} \sum_{a_j \in A_{C_2}} \mathbb{1}[Sc(a_i) < Sc(a_j)]}{|A_{C_1}||A_{C_2}|}, \tag{46}$$

where:

$$\mathbb{1}[Sc(a_i) < Sc(a_j)] = \begin{cases} 1, & \text{iff } \; Sc(a_i) < Sc(a_j), \\ 0, & \text{else.} \end{cases} \tag{47}$$

AUC builds on the number of pairs of alternatives from different classes for which the order of classes is reflected in the respective scores, i.e., a comprehensive score of a_i from the less preferred class than the class of a_j is lesser than $Sc(a_j)$. The measure is normalized by the number of all pairs of alternatives from different classes. Thus AUC indicates how many changes in the ranking imposed by the comprehensive scores are needed to obtain an entirely consistent outcome.

In the following subsection, we report the experimental results for eight algorithms proposed in this paper. We compare them against the following state-of-the-art preference learning methods:

- logistic regression (LR), which is a well-established statistical classification method, using the linear model of the input attributes (Hosmer et al., 2000); while estimating the

parameters of the weighted sum model, it optimizes the log-likelihood function capturing the probability of observing the desired classification for alternatives given the input data and the model;
- Choquistic regression (CR), i.e., a generalization of LR in which the Choquet integral is used as the preference model (Tehrani et al., 2012); when estimating values of its parameter, the algorithm also optimizes the log-likelihood function using a sequential quadratic programming approach implemented in Matlab;
- kernel logistic regression with the polynomial kernel (KLR-ply) and a degree equal to two so that it models low-level interactions of criteria (Tehrani et al., 2012);
- kernel logistic regression with the Gaussian kernel (KLR-rbf) able to capture interactions of higher-order; note that KLR methods are extensions of LR that are flexible but not necessarily monotonic in the sense of preserving pre-defined preference directions (Tehrani et al., 2012);
- the MORE algorithm that learns rule ensembles, adhering to monotonicity constraints, in which a single rule is treated as a subsidiary base classifier (Dembczyński et al., 2009); rule induction is performed by minimizing the sigmoid 0–1 loss function;
- the LMT algorithm that induces tree-structured models containing logistic regression functions at the leaves (Landwehr et al., 2003), while accounting for the least squared misclassification error;
- the UTADIS method, which employs linear programming provided by the IBM ILOG CPLEX solver (Sobrie et al., 2019) to infer a threshold-based value-driven sorting model using piecewise linear marginal functions with three segments (Zopounidis & Doumpos, 2000); it optimizes a misclassification error defined as an average distance of alternatives' comprehensive values from the value ranges delimited by the thresholds associated with their desired classes; the proposed ANN-based algorithms minimize the same objective function;
- the Mixed-Integer Program (MIP) for learning the parameters of MR-Sort, which is a simplified variant of ELECTRE TRI-B, using a majority rule and boundary class profiles (Leroy et al., 2011); the model parameters are selected by minimizing the 0/1 loss using the IBM ILOG CPLEX solver;
- the metaheuristic (META) for learning the parameters of MR-Sort (Sobrie et al., 2019) which uses evolutionary algorithms and mathematical programming to select parameter values minimizing the 0/1 loss.
- UTADIS-G, i.e., UTADIS employing general marginal value functions with the characteristic points corresponding to all unique performances (Greco, Mousseau, & Słowiński, 2010); the optimized objective is the same as for the standard UTADIS; the method has been implemented by the authors of this paper using the GLPK solver.

6.1. Estimation of hyperparameter values

In Section 4, we discussed the process of optimizing parameter values taking into account hyperparameters. This section is devoted to estimating the values of these hyperparameters as well as other hyperparameters involved in the operations of the proposed preference learning algorithms that are needed to train the models successfully.

To find the optimal values, we performed a grid search to verify the classification quality for different values. Specifically, we tested three hyperparameters:

- learning rate $\alpha \in \{0.001, 0.002, 0.005, 0.01, 0.02, 0.05\}$ (in addition, for ANN-OWA, all variants of ANN-Ch, and ANN-TOPSIS, we considered $\{0.1, 0.2, 0.5\}$);
- the number $L \in \{10, 20, 30\}$ of components used by *Monotonic Block* for ANN-UTADIS, ANN-PROMETHEE, and ANN-ELECTRE – it is the only parameter whose value needs to be provided before training for these methods;
- standard deviation $\xi \in \{0, 0.01, 0.02, 0.05\}$ of Gaussian noise used in date augmentation, where 0 means there is no additional noise added to the input data in each optimization step.

The range of a learning rate for ANN-OWA, ANN-Ch, and ANN-TOPSIS was extended due to the existing trend in the preliminary tests. They indicated that better results could be obtained for higher values of α. However, the extended tests revealed that this trend was valid only for a specific range of values, and after exceeding a certain threshold, the classification outcomes deteriorated.

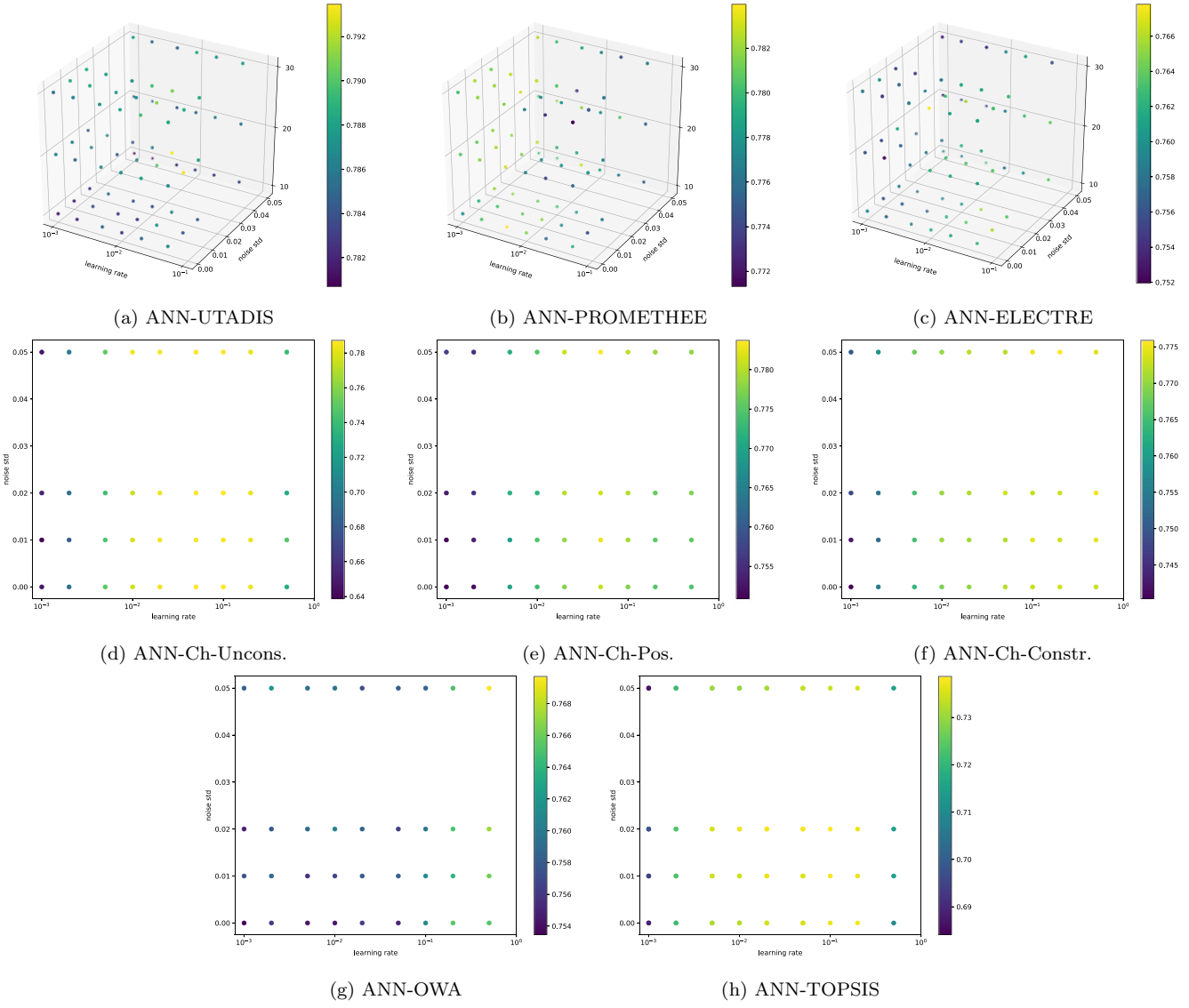
In a single test, we considered precise values for each of the above hyperparameters. The test was repeated 100 times for three sizes of the training and test sets. They correspond to the scenarios where (i) the training set is small compared to the test set (20% vs. 80%), (ii) both sets are equal in size (50% vs. 50%), and (iii) the training set contains a significant number of alternatives compared to the test set (80% vs. 20%), which is the most common setting. In each run, the allocation of alternatives to the training and test sets was performed randomly and independently. The selected values of hyperparameters are the ones for which the best average value of the performance measure was obtained for the training set in a hundredfold experiment described above.

In the main paper, we present the results obtained for the ERA dataset for 80% of training data and the AUC measure (see Fig. 15). The results for ANN-UTADIS were similar for different hyperparameter values, ranging from 0.7807 to 0.7935. The highest average score was obtained for $\alpha = 0.02$, $L = 20$, and $\xi = 0.02$. However, they cannot be claimed as the best hyperparameter values unanimously. The Student's T-test with a confidence level of 0.95 indicated that the AUC mean was statistically indistinguishable for 7 out of 72 configurations. On the other extreme, the lowest AUC value was observed for $\alpha = 0.001$, $L = 10$, and $\xi = 0.05$. There are no strict trends here, however, it can be observed that the results for $L = 20$ and $L = 30$ are more often better than for $L = 10$.

For ANN-PROMETHEE, we observe a trend indicating that better results are achieved for lesser values of learning rate and standard deviation of the noise. The best AUC score (0.7840) is attained for $\alpha = 0.005$, $L = 10$, $\xi = 0.0$, being, however, statistically indistinguishable for 41 out of 72 configurations. In turn, the best results for the ANN-ELECTRE are achieved for a learning rate of 0.01 and 0.02. At the same time, the greater the learning rate and lower noise std, the better the results. The number L of components has no significant influence on the results. The best combination of parameters is $\alpha = 0.01$, $L = 30$, and$\xi = 0.0$ with mean AUC 0.7678 (we noted 20 other statistically indistinguishable configurations).

For ANN-Ch-Uncons., we observe the greatest differences in classification outcomes among all methods. For different values of hyperparameters, AUC ranges between 0.6387 and 0.7872. The best outcomes are obtained for learning rates between 0.01 and 0.2. The highest average score was obtained for $\alpha = 0.05$ and $\xi = 0.02$. However, no statistically sound difference between means was observed for the other 13 out of 36 vectors of hyperparameter values. Also, for ANN-Ch-Uncons, we did not observe a noticeable impact of the input noise on the final results.

Similar trends occur for the remaining methods, i.e., the value of a learning rate for which the best results are obtained is: for ANN-Ch-Pos. – between 0.02 and 0.1, for ANN-TOPSIS – between



(a) ANN-UTADIS (b) ANN-PROMETHEE (c) ANN-ELECTRE

(d) ANN-Ch-Uncons. (e) ANN-Ch-Pos. (f) ANN-Ch-Constr.

(g) ANN-OWA (h) ANN-TOPSIS

Fig. 15. The AUC value attained for the training set by various methods for different hyperparameter values for the ERA dataset.

0.05 and 0.2, for ANN-Ch-Constr – it is above 0.1, and for ANN-OWA – it is equal to 0.5. The best configurations for these methods are: for ANN-Ch-Pos. – $\alpha = 0.05$ and $\xi = 0.05$, for ANN-TOPSIS – $\alpha = 0.1$ and $\xi = 0.02$, for ANN-Ch-Constr – $\alpha = 0.2$ and $\xi = 0.05$, and for ANN-OWA – $\alpha = 0.5$ and $\xi = 0.05$.

The above conclusions hold only for the ERA dataset. For some other sets, the dependencies differed. The respective figures are presented in the e-Appendix (supplementary material available online).

6.2. Experimental results in terms of AUC and 0/1 loss

In this section, we report the experimental results for 17 approaches, including eight proposed in this paper. All experiments were carried out on a single CPU 2300MHz Intel(R) Xeon(R) E5-2650 v3 using Python 3.6 and the Pytorch 1.2.0 library. The training times are shown in the e-Appendix. The outcomes for the state-of-the-art methods are derived from Tehrani et al. (2012) and Sobrie et al. (2019).

In Tables 4–6, we report the mean AUC values for nine benchmark datasets and different proportions of the training and test sets. For each approach, we provide the standard deviation, rank according to the mean for a given problem, and an average rank for all datasets (see the last column). A few missing values in the tables for MIP indicate that this approach was not able to find a solution within a pre-defined time limit. In what follows, we will discuss in detail the results obtained for 80% share of the training set (see Table 6). Then, we will indicate the major differences for the remaining two settings.

Let us start by discussing the specificity of different datasets. In general, the best AUC values were attained for CPU, ESL, DBS, and CEV. For example, the mean AUC values for ANN-UTADIS for these four datasets were 0.9998, 0.9885, 0.9676, and 0.9410, whereas the respective means attained by ANN-ELECTRE were 0.9998, 0.9600, 0.9893, and 0.8786. Such high-quality scores for CPU or ESL indicate that the best-performing approaches assigned such comprehensive scores to the alternatives that inversed the original preference relation only for a few or several pairs in the testing sets. On the other extreme, the least AUC values were observed for BCC and ERA. For these problems, ANN-UTADIS attained average values equal to 0.7830 and 0.7957, whereas for ANN-ELECTRE – these were 0.7497 and 0.7695. This means that the input and output orders were not consistent for about 20–25% of pairs in the test set. Such differences confirm that the considered datasets posed

Table 4
Classification performance in terms of the mean and standard deviation of AUC for 20% training data and 80% test data.

Method	DBS	CPU	BCC	MPG	ESL	ERA	LEV	CEV	MMG	Avg. rank
ANN-UTADIS	0.9159 ± 0.0230 (8)	0.9979 ± 0.0024 (1)	0.7513 ± 0.0158 (4)	0.8870 ± 0.0086 (9)	0.9839 ± 0.0030 (4)	0.7844 ± 0.0081 (1)	0.8955 ± 0.0066 (2)	0.9384 ± 0.0041 (8)	0.8769 ± 0.0118 (13)	5.56
ANN-PROMETHEE	0.9289 ± 0.0224 (3)	0.9918 ± 0.0089 (3)	0.7524 ± 0.0162 (2)	0.8750 ± 0.0088 (10)	0.9840 ± 0.0030 (3)	0.7801 ± 0.0087 (2)	0.8923 ± 0.0075 (5)	0.8919 ± 0.0060 (15)	0.8869 ± 0.0162 (8)	5.67
ANN-Ch-Uncons.	0.9181 ± 0.0150 (6)	0.9798 ± 0.0082 (9)	0.7292 ± 0.0211 (8)	0.9640 ± 0.0089 (6)	0.9835 ± 0.0037 (5)	0.7758 ± 0.0081 (4)	0.8930 ± 0.0069 (3)	0.9685 ± 0.0026 (6)	0.8866 ± 0.0075 (11)	6.44
ANN-Ch-Pos.	0.9202 ± 0.0161 (5)	0.9751 ± 0.0072 (11)	0.7495 ± 0.0187 (5)	0.7394 ± 0.0614 (17)	0.9834 ± 0.0029 (6)	0.7771 ± 0.0076 (3)	0.8929 ± 0.0059 (4)	0.9284 ± 0.0038 (11)	0.8868 ± 0.0128 (9)	7.89
ANN-Ch-Constr	0.9164 ± 0.0242 (7)	0.9806 ± 0.0073 (8)	0.7515 ± 0.0169 (3)	0.8451 ± 0.0147 (12)	0.9848 ± 0.0030 (2)	0.7721 ± 0.0094 (5)	0.8918 ± 0.0059 (6)	0.9344 ± 0.0071 (9)	0.8920 ± 0.0081 (4)	6.22
ANN-ELECTRE	0.9285 ± 0.0210 (4)	0.9971 ± 0.0038 (2)	0.7325 ± 0.0177 (6)	0.8540 ± 0.0219 (11)	0.9854 ± 0.0023 (1)	0.7640 ± 0.0094 (9)	0.8852 ± 0.0077 (10)	0.8753 ± 0.0160 (16)	0.8960 ± 0.0119 (2)	6.78
ANN-OWA	0.9077 ± 0.0161 (10)	0.9411 ± 0.0110 (17)	0.7533 ± 0.0180 (1)	0.6514 ± 0.0159 (18)	0.9808 ± 0.0029 (7)	0.7654 ± 0.0074 (8)	0.8688 ± 0.0064 (14)	0.7240 ± 0.0070 (17)	0.8897 ± 0.0057 (5)	10.78
ANN-TOPSIS	0.8919 ± 0.0191 (12)	0.9130 ± 0.0191 (18)	0.7243 ± 0.0139 (9)	0.9533 ± 0.0046 (7)	0.7806 ± 0.0112 (18)	0.7369 ± 0.0076 (14)	0.8137 ± 0.0073 (17)	0.9655 ± 0.0028 (7)	0.8527 ± 0.0080 (17)	13.22
CR	0.9290 ± 0.0322 (2)	0.9822 ± 0.0121 (5)	0.6400 ± 0.0641 (18)	0.9788 ± 0.0160 (1)	0.9670 ± 0.0074 (12)	0.7669 ± 0.0334 (6)	0.8971 ± 0.0098 (1)	0.9825 ± 0.0080 (3)	0.8867 ± 0.0123 (10)	6.44
LR	0.8866 ± 0.0511 (14)	0.9806 ± 0.0124 (7)	0.6970 ± 0.0411 (12)	0.9675 ± 0.0068 (5)	0.9721 ± 0.0060 (8)	0.7602 ± 0.0331 (11)	0.8905 ± 0.0081 (7)	0.9332 ± 0.0033 (10)	0.8962 ± 0.0080 (1)	8.33
KLR-ply	0.9359 ± 0.0218 (1)	0.9716 ± 0.0072 (13)	0.6509 ± 0.0568 (17)	0.9704 ± 0.0075 (4)	0.9638 ± 0.0106 (13)	0.7555 ± 0.0139 (12)	0.8870 ± 0.0094 (8)	0.9818 ± 0.0058 (5)	0.8552 ± 0.0203 (16)	9.89
KLR-rbf	0.9053 ± 0.0433 (11)	0.9843 ± 0.0116 (4)	0.7124 ± 0.0290 (11)	0.9741 ± 0.0055 (3)	0.9705 ± 0.0099 (9)	0.7662 ± 0.0098 (7)	0.8860 ± 0.0128 (9)	0.9821 ± 0.0076 (4)	0.8938 ± 0.0121 (3)	6.78
MORE	0.8731 ± 0.0481 (16)	0.9749 ± 0.0235 (12)	0.6639 ± 0.0567 (15)	0.9501 ± 0.0263 (8)	0.9466 ± 0.0484 (17)	0.7198 ± 0.0329 (17)	0.8137 ± 0.0621 (18)	0.9888 ± 0.0063 (2)	0.8754 ± 0.0274 (14)	13.22
LMT	0.9151 ± 0.0228 (9)	0.9816 ± 0.0113 (6)	0.7310 ± 0.0675 (7)	0.9753 ± 0.0092 (2)	0.9696 ± 0.0086 (11)	0.7619 ± 0.0160 (10)	0.8797 ± 0.0182 (11)	0.9902 ± 0.0042 (1)	0.8890 ± 0.0259 (6)	7.00
META	0.8761 ± 0.0462 (15)	0.9531 ± 0.0247 (15)	0.6810 ± 0.0458 (13)	0.8337 ± 0.0291 (13)	0.9569 ± 0.0114 (15)	0.7256 ± 0.0238 (16)	0.8530 ± 0.0258 (15)	0.8968 ± 0.0116 (14)	0.8828 ± 0.0129 (12)	14.22
MIP	0.8637 ± 0.0463 (17)	0.9497 ± 0.0262 (16)	0.7155 ± 0.0365 (10)	0.8215 ± 0.0368 (15)	0.9510 ± 0.0166 (16)	0.7182 ± 0.0328 (18)	0.8424 ± 0.0291 (16)	-	0.8877 ± 0.0151 (7)	14.78
UTADIS	0.8886 ± 0.0496 (13)	0.9789 ± 0.0283 (10)	0.6650 ± 0.0527 (14)	0.8162 ± 0.0335 (16)	0.9704 ± 0.0095 (10)	0.7409 ± 0.0175 (13)	0.8707 ± 0.0146 (12)	0.9235 ± 0.0183 (13)	0.8650 ± 0.0294 (15)	12.89
UTADIS-G	0.8564 ± 0.0507 (18)	0.9552 ± 0.0366 (14)	0.6617 ± 0.0489 (16)	0.8314 ± 0.0328 (14)	0.9636 ± 0.0122 (14)	0.7307 ± 0.0233 (15)	0.8705 ± 0.0134 (13)	0.9269 ± 0.0149 (12)	0.8474 ± 0.0284 (18)	14.89

Table 5
Classification performance in terms of the mean and standard deviation of AUC for 50% training data and 50% test data.

Method	DBS	CPU	BCC	MPG	ESL	ERA	LEV	CEV	MMG	Avg. rank
ANN-UTADIS	0.9399 ± 0.0292 (4)	0.9991 ± 0.0017 (1)	0.7632 ± 0.0307 (3)	0.8911 ± 0.0164 (9)	0.9859 ± 0.0047 (3)	0.7880 ± 0.0144 (1)	0.8996 ± 0.0100 (4)	0.9395 ± 0.0060 (8)	0.8815 ± 0.0162 (14)	5.22
ANN-PROMETHEE	0.9446 ± 0.0266 (2)	0.9971 ± 0.0044 (3)	0.7636 ± 0.0336 (2)	0.8746 ± 0.0180 (10)	0.9856 ± 0.0042 (4)	0.7839 ± 0.0136 (2)	0.8946 ± 0.0131 (8)	0.8960 ± 0.0099 (14)	0.8874 ± 0.0168 (11)	6.22
ANN-Ch-Uncons.	0.9301 ± 0.0234 (8)	0.9890 ± 0.0060 (8)	0.7517 ± 0.0252 (6)	0.9713 ± 0.0084 (6)	0.9855 ± 0.0045 (5)	0.7823 ± 0.0163 (3)	0.8974 ± 0.0110 (5)	0.9707 ± 0.0036 (6)	0.8923 ± 0.0124 (8)	6.11
ANN-Ch-Pos.	0.9303 ± 0.0254 (7)	0.9821 ± 0.0082 (13)	0.7560 ± 0.0359 (5)	0.7538 ± 0.0489 (16)	0.9844 ± 0.0048 (6)	0.7816 ± 0.0148 (4)	0.8963 ± 0.0101 (6)	0.9302 ± 0.0060 (13)	0.8878 ± 0.0167 (10)	8.89
ANN-Ch-Constr	0.9299 ± 0.0299 (9)	0.9865 ± 0.0056 (11)	0.7641 ± 0.0314 (1)	0.8494 ± 0.0224 (12)	0.9870 ± 0.0039 (1)	0.7769 ± 0.0138 (5)	0.8957 ± 0.0105 (7)	0.9357 ± 0.0088 (10)	0.8952 ± 0.0113 (7)	7.00
ANN-ELECTRE	0.9416 ± 0.0251 (3)	0.9988 ± 0.0020 (2)	0.7318 ± 0.0368 (9)	0.8536 ± 0.0218 (11)	0.9864 ± 0.0042 (2)	0.7652 ± 0.0150 (11)	0.8869 ± 0.0110 (11)	0.8751 ± 0.0192 (16)	0.9019 ± 0.0128 (1)	7.33
ANN-OWA	0.9117 ± 0.0296 (15)	0.9447 ± 0.0138 (17)	0.7568 ± 0.0336 (4)	0.6575 ± 0.0281 (17)	0.9816 ± 0.0049 (7)	0.7665 ± 0.0150 (10)	0.8714 ± 0.0119 (15)	0.7236 ± 0.0113 (17)	0.8920 ± 0.0112 (9)	12.33
ANN-TOPSIS	0.9082 ± 0.0284 (16)	0.9193 ± 0.0176 (18)	0.7402 ± 0.0293 (7)	0.9545 ± 0.0097 (8)	0.7844 ± 0.0262 (18)	0.7416 ± 0.0171 (14)	0.8203 ± 0.0125 (17)	0.9662 ± 0.0033 (7)	0.8569 ± 0.0119 (16)	13.44
CR	0.9341 ± 0.0228 (5)	0.9920 ± 0.0073 (6)	0.6912 ± 0.0469 (15)	0.9818 ± 0.0075 (1)	0.9720 ± 0.0084 (12)	0.7705 ± 0.0310 (9)	0.9098 ± 0.0103 (1)	0.9912 ± 0.0024 (4)	0.9003 ± 0.0132 (2)	6.11
LR	0.9191 ± 0.0293 (11)	0.9914 ± 0.0056 (7)	0.7184 ± 0.0367 (11)	0.9803 ± 0.0084 (3)	0.9764 ± 0.0062 (8)	0.7633 ± 0.0241 (12)	0.8935 ± 0.0113 (9)	0.9362 ± 0.0071 (9)	0.8972 ± 0.0125 (5)	8.33
KLR-ply	0.9492 ± 0.0198 (1)	0.9771 ± 0.0109 (14)	0.7001 ± 0.0396 (12)	0.9776 ± 0.0083 (4)	0.9726 ± 0.0080 (11)	0.7740 ± 0.0148 (7)	0.8999 ± 0.0120 (3)	0.9950 ± 0.0019 (2)	0.8962 ± 0.0140 (6)	6.67
KLR-rbf	0.9174 ± 0.0316 (13)	0.9925 ± 0.0056 (5)	0.7294 ± 0.0344 (10)	0.9752 ± 0.0068 (5)	0.9754 ± 0.0070 (9)	0.7745 ± 0.0141 (6)	0.9012 ± 0.0128 (2)	0.9907 ± 0.0031 (5)	0.8995 ± 0.0091 (3)	6.44
MORE	0.9179 ± 0.0403 (12)	0.9873 ± 0.0149 (10)	0.6980 ± 0.0586 (13)	0.9563 ± 0.0313 (7)	0.9557 ± 0.0301 (17)	0.7215 ± 0.0381 (17)	0.8185 ± 0.0580 (18)	0.9921 ± 0.0042 (3)	0.8839 ± 0.0305 (13)	12.22
LMT	0.9259 ± 0.0289 (10)	0.9883 ± 0.0077 (9)	0.7387 ± 0.0656 (8)	0.9814 ± 0.0074 (2)	0.9707 ± 0.0120 (14)	0.7719 ± 0.0144 (8)	0.8920 ± 0.0164 (10)	0.9977 ± 0.0017 (1)	0.8976 ± 0.0153 (4)	7.33
META	0.9074 ± 0.0366 (17)	0.9701 ± 0.0140 (15)	0.6929 ± 0.0398 (14)	0.8337 ± 0.0231 (14)	0.9640 ± 0.0099 (15)	0.7366 ± 0.0233 (16)	0.8721 ± 0.0147 (14)	0.8960 ± 0.0073 (15)	0.8862 ± 0.0138 (12)	14.67
MIP	0.8998 ± 0.0336 (18)	0.9645 ± 0.0194 (16)	–	–	0.9563 ± 0.0114 (16)	0.7167 ± 0.0274 (18)	0.8511 ± 0.0219 (16)	–	–	17.33
UTADIS	0.9325 ± 0.0345 (6)	0.9940 ± 0.0131 (4)	0.6650 ± 0.5270 (16)	0.8272 ± 0.0243 (15)	0.9747 ± 0.0116 (10)	0.7437 ± 0.0211 (13)	0.8746 ± 0.0137 (12)	0.9339 ± 0.0138 (11)	0.8667 ± 0.0385 (15)	11.33
UTADIS-G	0.9117 ± 0.0332 (14)	0.9830 ± 0.0201 (12)	0.6571 ± 0.0524 (17)	0.8456 ± 0.0205 (13)	0.9714 ± 0.0069 (13)	0.7388 ± 0.0187 (15)	0.8738 ± 0.0134 (13)	0.9329 ± 0.0114 (12)	0.8439 ± 0.0253 (17)	14.00

Table 6
Classification performance in terms of the mean and standard deviation of AUC for 80% training data and 20% test data.

Method	DBS	CPU	BCC	MPG	ESL	ERA	LEV	CEV	MMG	Avg. rank
ANN-UTADIS	0.9676 ± 0.0319 (1)	0.9998 ± 0.0007 (1)	0.7830 ± 0.0656 (4)	0.9034 ± 0.0307 (9)	0.9885 ± 0.0086 (3)	0.7957 ± 0.0290 (1)	0.9044 ± 0.0218 (3)	0.9410 ± 0.0114 (8)	0.8856 ± 0.0280 (14)	4.89
ANN-PROMETHEE	0.9574 ± 0.0433 (4)	0.9988 ± 0.0029 (4)	0.7896 ± 0.0584 (2)	0.8794 ± 0.0328 (10)	0.9885 ± 0.0078 (4)	0.7891 ± 0.0291 (3)	0.8980 ± 0.0202 (9)	0.8977 ± 0.0168 (14)	0.8943 ± 0.0249 (10)	6.67
ANN-Ch-Uncons.	0.9492 ± 0.0507 (7)	0.9937 ± 0.0072 (8)	0.8074 ± 0.0595 (1)	0.9788 ± 0.0123 (5)	0.9880 ± 0.0077 (5)	0.7911 ± 0.0284 (2)	0.9035 ± 0.0207 (4)	0.9726 ± 0.0067 (6)	0.8988 ± 0.0200 (7)	5.00
ANN-Ch-Pos.	0.9368 ± 0.0534 (11)	0.9892 ± 0.0092 (13)	0.7712 ± 0.0636 (5)	0.7650 ± 0.0655 (16)	0.9867 ± 0.0091 (6)	0.7856 ± 0.0268 (4)	0.9019 ± 0.0197 (6)	0.9303 ± 0.0119 (13)	0.8923 ± 0.0228 (11)	9.44
ANN-Ch-Constr	0.9522 ± 0.0474 (5)	0.9905 ± 0.0101 (12)	0.7865 ± 0.0632 (3)	0.8503 ± 0.0394 (14)	0.9886 ± 0.0075 (2)	0.7812 ± 0.0300 (5)	0.8998 ± 0.0206 (7)	0.9376 ± 0.0131 (10)	0.8981 ± 0.0233 (8)	7.33
ANN-ELECTRE	0.9600 ± 0.0393 (3)	0.9998 ± 0.0011 (2)	0.7497 ± 0.0621 (8)	0.8582 ± 0.0388 (12)	0.9893 ± 0.0086 (1)	0.7695 ± 0.0284 (10)	0.8880 ± 0.0230 (11)	0.8786 ± 0.0245 (16)	0.9066 ± 0.0185 (2)	7.22
ANN-OWA	0.9293 ± 0.0548 (14)	0.9531 ± 0.0271 (17)	0.7660 ± 0.0674 (6)	0.6614 ± 0.0610 (17)	0.9838 ± 0.0102 (7)	0.7696 ± 0.0300 (9)	0.8726 ± 0.0222 (14)	0.7304 ± 0.0234 (17)	0.8954 ± 0.0214 (9)	12.22
ANN-TOPSIS	0.9322 ± 0.0517 (13)	0.9318 ± 0.0363 (18)	0.7573 ± 0.0548 (7)	0.9598 ± 0.0180 (7)	0.7911 ± 0.0615 (18)	0.7499 ± 0.0337 (13)	0.8251 ± 0.0233 (17)	0.9684 ± 0.0054 (7)	0.8600 ± 0.0305 (16)	12.89
CR	0.9427 ± 0.0443 (9)	0.9971 ± 0.0063 (6)	0.7349 ± 0.0692 (9)	0.9855 ± 0.0108 (1)	0.9766 ± 0.0150 (10)	0.7670 ± 0.0290 (11)	0.9122 ± 0.0202 (1)	0.9959 ± 0.0027 (3)	0.9135 ± 0.0233 (1)	5.67
LR	0.9224 ± 0.0514 (15)	0.9907 ± 0.0085 (11)	0.7253 ± 0.0715 (12)	0.9843 ± 0.0138 (2)	0.9722 ± 0.0167 (13)	0.7630 ± 0.0281 (12)	0.8928 ± 0.0234 (10)	0.9352 ± 0.0095 (12)	0.9048 ± 0.0237 (4)	10.11
KLR-ply	0.9608 ± 0.0347 (2)	0.9827 ± 0.0167 (14)	0.7071 ± 0.0720 (13)	0.9797 ± 0.0121 (4)	0.9746 ± 0.0141 (12)	0.7731 ± 0.0293 (8)	0.9048 ± 0.0201 (2)	0.9942 ± 0.0018 (4)	0.9011 ± 0.0199 (5)	7.11
KLR-rbf	0.9495 ± 0.0459 (6)	0.9984 ± 0.0052 (5)	0.7335 ± 0.0690 (11)	0.9771 ± 0.0142 (6)	0.9782 ± 0.0126 (8)	0.7759 ± 0.0315 (6)	0.9031 ± 0.0172 (5)	0.9970 ± 0.0013 (2)	0.8991 ± 0.0255 (6)	6.11
MORE	0.9409 ± 0.0539 (10)	0.9909 ± 0.0167 (9)	0.7042 ± 0.0853 (15)	0.9551 ± 0.0372 (8)	0.9507 ± 0.0508 (17)	0.7228 ± 0.0475 (18)	0.8078 ± 0.0661 (18)	0.9936 ± 0.0046 (5)	0.8889 ± 0.0363 (12)	12.44
LMT	0.9343 ± 0.0479 (12)	0.9959 ± 0.0078 (7)	0.7342 ± 0.0791 (10)	0.9841 ± 0.0106 (3)	0.9713 ± 0.0176 (14)	0.7735 ± 0.0296 (7)	0.8996 ± 0.0222 (8)	0.9993 ± 0.0017 (1)	0.9063 ± 0.0215 (3)	7.22
META	0.9019 ± 0.0606 (18)	0.9721 ± 0.0219 (15)	0.7056 ± 0.0864 (14)	0.8613 ± 0.0341 (11)	0.9613 ± 0.0170 (15)	0.7379 ± 0.0351 (16)	0.8663 ± 0.0265 (15)	0.8941 ± 0.0135 (15)	0.8860 ± 0.0265 (13)	14.67
MIP	0.9080 ± 0.0673 (17)	0.9656 ± 0.0237 (16)	–	–	0.9568 ± 0.0165 (16)	0.7242 ± 0.0477 (17)	0.8499 ± 0.0332 (16)	–	–	17.11
UTADIS	0.9476 ± 0.0401 (8)	0.9989 ± 0.0030 (3)	0.6651 ± 0.0659 (16)	0.8210 ± 0.0434 (15)	0.9778 ± 0.0117 (9)	0.7497 ± 0.0402 (14)	0.8741 ± 0.0217 (13)	0.9399 ± 0.0111 (9)	0.8682 ± 0.0470 (15)	11.33
UTADIS-G	0.9127 ± 0.0467 (16)	0.9909 ± 0.0214 (10)	0.6309 ± 0.0723 (17)	0.8507 ± 0.0365 (13)	0.9748 ± 0.0109 (11)	0.7448 ± 0.0300 (15)	0.8752 ± 0.0227 (12)	0.9373 ± 0.0110 (11)	0.8288 ± 0.0278 (17)	13.56

various challenges to the preference learning algorithms. In the e-Appendix, we discuss various characteristics that partially explain such results. For example, CPU involves six criteria, each with at least several different performances, and no single violation of the dominance or indistinguishability relation in the desired assignments. Analogously, the desired assignments for ESL agree with the dominance relation for the vast majority of pairs of alternatives, and only a tiny share of pairs are inconsistent with the dominance or indistinguishability. On the other extreme, the seven criteria for BCC involve just a few different performances, and almost 7% of all pairs of alternatives assigned to different classes violate the dominance.

Also, some datasets differentiated the considered sorting methods better than others. The greatest differences between mean AUC values were observed for MPG (for CR – 9855 and for ANN-OWA – 6614), CEV (for LMT – 0.9993 and for ANN-OWA – 0.7304), ESL (for ANN-ELECTRE – 0.9893 and for ANN-TOPSIS – 0.7911). This confirms that their specificity posed a significantly greater challenge to some approaches. On the contrary, the least differences were noted for DBS (for ANN-UTADIS – 0.9676 and for META – 0.9019) and CPU (for ANN-UTADIS – 0.9998 and for ANN-TOPSIS – 0.9318). Still, even for these benchmark problems, it was possible to distinguish the subsets of clearly better- or worse-performing methods.

The most favorable average ranks implied by the mean AUC measures for the nine datasets are attained by:

- ANN-UTADIS (4.89), which attains the best results for DBS, CPU, and ERA, positions in the top four for other three problems, and is ranked outside the top ten only for MMG;
- ANN-Ch-Uncons. (5.00), which is the most advantageous for BCC, while never dropping outside the upper half of the ranking; note that this method has a competitive advantage of not having to respect the pre-defined preference directions, which is particularly useful for datasets such as BCC (1st rank), MPG (5th rank), and MMG (7th rank), for which some originally nominal attributes have been arbitrarily transformed to monotonic criteria in Tehrani et al. (2012);
- CR (5.67), which attains the highest mean AUC for MPG, LEV, and MMG, while being ninth or lower for four other datasets;
- KLR-rbf (6.11), attaining ranks between second for CEV and eleventh for BCC;
- ANN-PROMETHEE (6.67), ranked in the top four for most datasets.

On the other extreme, the worst average ranks are attained by MIP (17.11), META (14.67), UTADIS-G (13.56), ANN-TOPSIS (12.89), MORE (12.44), ANN-OWA (12.22), and UTADIS (11.33). Hence, only ANN-OWA and ANN-TOPSIS achieved relatively worse results among the proposed algorithms. This can be attributed to simple preference models employed by these methods.

Following Tehrani et al., (2012), we applied the statistical tests to verify the significance of the performance differences. The Friedman test allowed us to reject the null hypothesis on all methods performing equally for all sizes of the training set and both considered measures (AUC and 0/1 loss). The detailed outcomes of a post hoc analysis for all pairs of algorithms conducted using the Nemenyi and Wilcoxon tests with a confidence level of 90% are discussed in the e-Appendix. In what follows, we directly compare pairwise only the approaches using similar preference models. When claiming that some performance difference in terms of AUC is significant, this is confirmed by the result of the Nemenyi test applied to a subset of algorithms using related models.

ANN-UTADIS performs significantly better than UTADIS (the Wilcoxon test) and UTADIS-G (the Wilcoxon and Nemenyi tests) based on mathematical programming. The reasons are as follows. First, minimizing the sum of regrets by UTADIS and UTADIS-G does

not correspond to the perspective captured by AUC. Also, the use of *Monotonic Block* by ANN-UTADIS gives a chance for inferring very flexible marginal value functions with characteristic points better fitting the input data. In turn, data augmentation prevents the model overfitting that occurs with UTADIS-G.

When it comes to outranking-based methods, ANN-PROMETHEE significantly outperforms MIP and META, which learn the parameters of the MR-Sort model using, respectively, Mixed Integer Programming and a dedicated heuristic. Furthermore, ANN-ELECTRE attains significantly better results than MIP. The ANN-based methods proposed in this paper use the NFS procedure, threshold-based sorting method, and flexible marginal preference, concordance, and discordance functions. In turn, the model used in MR-Sort is more complex with the boundary profiles whose performances need to be determined by the method and concordance functions without zones of indifference and weak preference, hence offering lesser flexibility.

The results attained for all algorithms using the Choquet integral model (i.e., three variants of ANN-Ch and CR) are very similar for DBS, CPU, ESL, LEV, and MMG. For BBC and ERA, CR was worse than the ANN-based methods. In turn, ANN-Ch-Constr. and ANN-Ch-Pos. were outperformed by CR on MPG and CEV. The variant without any constraint on the weights performed better for these challenging datasets because it could fit the data even better by inversing the pre-defined preference directions via assigning the negative weights. Overall, the Nemenyi test confirmed that ANN-Ch-Uncons. and CR were significantly better than ANN-Ch-Pos.

When it comes to logistic regression methods, KLR-ply and KLR-rbf perform, on average, better than LR. This is due to the non-monotonic KLR methods being able to capture low- (ply) or high-level (rbf) interactions. However, according to the Wilcoxon test, the statistically significant difference is observed only for KLR-rbf and LR. Moreover, the slight advantage of KLR methods is not implied by admitting non-monotonicity for datasets that originally involved nominal criteria (e.g., for MPG and MMG, LR attains better results than both KLR-rbf and KLR-ply).

The observations, rankings, and trends for other proportions of the training and test sets (see Tables 4 and 5) are very similar to the outcomes discussed above for the 80/20 division. However, with the decrease in the number of alternatives in the training set, the AUC decreases by a few percent for the ANN-based methods. For example, ANN-UTADIS attains an average AUC equal to 0.9676, 0.9399, and 0.9159 for DBS with 80/20, 50/50, and 20/80 shares of the training and test sets, whereas the analogous results attained by ANN-Ch-Constr. for BCC are 0.7865, 0.7641, and 0.7515. No or marginal performance deterioration is observed for ANN-PROMETHEE and ANN-ELECTRE for datasets with a larger number of alternatives, i.e., MPG, ERA, LEV, and CEV. For example, for ANN-PROMETHEE and MPG, AUC is 0.8794 for 80% training set, 0.8746 for 50%, and 0.8750 for 20%. As a result, the average ranks for these approaches are slightly better for the least size of training data than for more numerous learning sets. In fact, for the 20/80 division, ANN-PROMETHEE shares the best average rank with ANN-UTADIS. In the same spirit, the average ranks for ANN-Ch-Constr., ANN-Ch-Pos., and ANN-OWA get slightly better with the decrease of the training set's share. The opposite trend is observed for ANN-UTADIS and ANN-Ch-Uncons. The greatest improvement of ranks for smaller training data among the state-of-the-art algorithms is observed for LR and META. In contrast, the most significant deterioration is noted for KLR-ply, UTADIS, and UTADIS-G.

In Tables 7–9, we report the mean values of 0/1 loss for nine benchmark datasets and different proportions of the training and test sets. Unlike for AUC, lesser values of 0/1 loss are more favorable. Let us first focus on the results for 80% share of the training set (see Table 9). They confirm the conclusions derived from AUC analysis on the challenge posed by different datasets to the

Table 7
Classification performance in terms of the mean and standard deviation of 0/1 loss for 20% training data and 80% test data.

Method	DBS	CPU	BCC	MPG	ESL	ERA	LEV	CEV	MMG	Avg. rank
ANN-UTADIS	0.1460 ± 0.0316 (6)	0.0185 ± 0.0104 (2)	0.2371 ± 0.0140 (2)	0.1795 ± 0.0130 (9)	0.0620 ± 0.0077 (2)	0.2688 ± 0.0077 (6)	0.1648 ± 0.0092 (4)	0.1118 ± 0.0053 (8)	0.1787 ± 0.0160 (9)	5.33
ANN-PROMETHEE	0.1421 ± 0.0196 (5)	0.1716 ± 0.0528 (18)	0.2659 ± 0.0317 (7)	0.2052 ± 0.0135 (12)	0.0849 ± 0.0081 (12)	0.2959 ± 0.0110 (13)	0.1830 ± 0.0081 (15)	0.2362 ± 0.0087 (15)	0.1894 ± 0.0112 (16)	12.56
ANN-Ch-Uncons.	0.1406 ± 0.0237 (4)	0.0674 ± 0.0205 (4)	0.2406 ± 0.0167 (3)	0.0868 ± 0.0086 (7)	0.0638 ± 0.0080 (3)	0.2707 ± 0.0076 (7)	0.1686 ± 0.0077 (9)	0.0803 ± 0.0060 (6)	0.1795 ± 0.0082 (11)	6.00
ANN-Ch-Pos.	0.1340 ± 0.0289 (3)	0.0805 ± 0.0197 (8)	0.2488 ± 0.0130 (5)	0.2789 ± 0.0384 (18)	0.0652 ± 0.0074 (4)	0.2671 ± 0.0057 (4)	0.1706 ± 0.0069 (12)	0.1306 ± 0.0050 (11)	0.1742 ± 0.0152 (6)	7.89
ANN-Ch-Constr	0.1276 ± 0.0333 (1)	0.0706 ± 0.0151 (5)	0.2362 ± 0.0142 (1)	0.2071 ± 0.0138 (13)	0.0608 ± 0.0081 (1)	0.2716 ± 0.0074 (8)	0.1695 ± 0.0073 (11)	0.1222 ± 0.0051 (9)	0.1660 ± 0.0092 (1)	5.56
ANN-ELECTRE	0.1278 ± 0.0223 (2)	0.0175 ± 0.0144 (1)	0.3411 ± 0.0131 (18)	0.2348 ± 0.0270 (16)	0.0714 ± 0.0101 (7)	0.3073 ± 0.0080 (16)	0.1835 ± 0.0065 (16)	0.2569 ± 0.0298 (16)	0.1743 ± 0.0153 (7)	11.00
ANN-OWA	0.1477 ± 0.0208 (7)	0.1287 ± 0.0186 (16)	0.2455 ± 0.0125 (4)	0.2628 ± 0.0107 (17)	0.0708 ± 0.0062 (6)	0.2671 ± 0.0068 (5)	0.1816 ± 0.0066 (14)	0.2638 ± 0.0034 (17)	0.1802 ± 0.0076 (12)	10.89
ANN-TOPSIS	0.1671 ± 0.0231 (8)	0.1515 ± 0.0238 (17)	0.2551 ± 0.0157 (6)	0.1066 ± 0.0106 (8)	0.2674 ± 0.0133 (18)	0.2926 ± 0.0078 (11)	0.2270 ± 0.0086 (18)	0.0892 ± 0.0048 (7)	0.2125 ± 0.0073 (18)	12.33
CR	0.1713 ± 0.0424 (10)	0.0811 ± 0.0103 (9)	0.2775 ± 0.0335 (10)	0.0709 ± 0.0193 (1)	0.0682 ± 0.0129 (5)	0.2889 ± 0.0273 (9)	0.1499 ± 0.0122 (1)	0.0448 ± 0.0089 (3)	0.1725 ± 0.0120 (4)	5.78
LR	0.2124 ± 0.0650 (17)	0.0711 ± 0.0312 (6)	0.2893 ± 0.0240 (16)	0.0832 ± 0.0151 (6)	0.0733 ± 0.0107 (8)	0.2902 ± 0.0317 (10)	0.1655 ± 0.0082 (6)	0.1410 ± 0.0079 (12)	0.1729 ± 0.0122 (5)	9.56
KLR-ply	0.1695 ± 0.0437 (9)	0.0996 ± 0.0231 (15)	0.2760 ± 0.0243 (9)	0.0788 ± 0.0097 (4)	0.1488 ± 0.0278 (17)	0.3001 ± 0.0130 (15)	0.1627 ± 0.0119 (3)	0.0663 ± 0.0130 (5)	0.1960 ± 0.0160 (17)	10.44
KLR-rbf	0.1883 ± 0.0536 (12)	0.0802 ± 0.0292 (7)	0.2787 ± 0.0237 (11)	0.0772 ± 0.0107 (2)	0.0756 ± 0.0167 (9)	0.2934 ± 0.0112 (12)	0.1691 ± 0.0125 (10)	0.0618 ± 0.0151 (4)	0.1791 ± 0.0133 (10)	8.56
MORE	0.1932 ± 0.0511 (14)	0.0829 ± 0.0379 (10)	0.2827 ± 0.0255 (13)	0.0811 ± 0.0119 (5)	0.0838 ± 0.0241 (11)	0.3155 ± 0.0150 (17)	0.1707 ± 0.0186 (13)	0.0339 ± 0.0076 (1)	0.1764 ± 0.0137 (8)	10.22
LMT	0.1779 ± 0.0420 (11)	0.0850 ± 0.0256 (12)	0.2884 ± 0.0306 (15)	0.0773 ± 0.0148 (3)	0.0771 ± 0.0148 (10)	0.2963 ± 0.0126 (14)	0.1672 ± 0.0140 (7)	0.0432 ± 0.0116 (2)	0.1803 ± 0.0171 (13)	9.67
META	0.1897 ± 0.0423 (13)	0.0994 ± 0.0323 (14)	0.2824 ± 0.0273 (12)	0.2025 ± 0.0356 (11)	0.1042 ± 0.0171 (15)	0.2136 ± 0.0205 (2)	0.1674 ± 0.0187 (8)	0.1488 ± 0.0135 (13)	0.1697 ± 0.0087 (2)	10.00
MIP	0.1977 ± 0.0481 (15)	0.0900 ± 0.0345 (13)	0.2678 ± 0.0276 (8)	0.2080 ± 0.0326 (14)	0.1075 ± 0.0158 (16)	0.2093 ± 0.0174 (1)	0.1608 ± 0.0173 (2)	-	0.1716 ± 0.0140 (3)	10.00
UTADIS	0.2008 ± 0.0533 (16)	0.0652 ± 0.0362 (3)	0.2915 ± 0.0307 (17)	0.2225 ± 0.0318 (15)	0.0889 ± 0.0160 (13)	0.2368 ± 0.0187 (3)	0.1654 ± 0.0160 (5)	0.1300 ± 0.0142 (10)	0.1840 ± 0.0184 (15)	10.78
UTADIS-G	0.2136 ± 0.0460 (18)	0.0846 ± 0.0448 (11)	0.2852 ± 0.0245 (14)	0.2006 ± 0.0398 (10)	0.0903 ± 0.0153 (14)	0.3393 ± 0.0561 (18)	0.1943 ± 0.0247 (17)	0.1679 ± 0.0536 (14)	0.1820 ± 0.0188 (14)	14.44

Table 8
Classification performance in terms of the mean and standard deviation of 0/1 loss for 50% training data and 50% test data.

Method	DBS	CPU	BCC	MPG	ESL	ERA	LEV	CEV	MMG	Avg. rank
ANN-UTADIS	0.1093 ± 0.0353 (2)	0.0104 ± 0.0118 (2)	0.2276 ± 0.0165 (3)	0.1735 ± 0.0197 (9)	0.0546 ± 0.0099 (1)	0.2640 ± 0.0135 (4)	0.1566 ± 0.0110 (8)	0.1126 ± 0.0080 (8)	0.1717 ± 0.0192 (10)	5.22
ANN-PROMETHEE	0.1210 ± 0.0374 (6)	0.1381 ± 0.0569 (18)	0.2709 ± 0.0374 (11)	0.2031 ± 0.0232 (13)	0.0812 ± 0.0122 (14)	0.2900 ± 0.0138 (13)	0.1768 ± 0.0119 (14)	0.2273 ± 0.0138 (15)	0.1812 ± 0.0222 (16)	13.33
ANN-Ch-Uncons.	0.1178 ± 0.0306 (4)	0.0430 ± 0.0200 (4)	0.2196 ± 0.0199 (1)	0.0756 ± 0.0137 (7)	0.0582 ± 0.0114 (3)	0.2653 ± 0.0116 (7)	0.1605 ± 0.0113 (10)	0.0731 ± 0.0060 (6)	0.1711 ± 0.0138 (9)	5.67
ANN-Ch-Pos.	0.1207 ± 0.0313 (5)	0.0637 ± 0.0202 (11)	0.2391 ± 0.0254 (4)	0.2691 ± 0.0346 (17)	0.0605 ± 0.0121 (5)	0.2642 ± 0.0130 (5)	0.1648 ± 0.0125 (12)	0.1277 ± 0.0078 (10)	0.1687 ± 0.0161 (5)	8.22
ANN-Ch-Constr	0.1032 ± 0.0323 (1)	0.0562 ± 0.0155 (9)	0.2201 ± 0.0235 (2)	0.2015 ± 0.0207 (12)	0.0559 ± 0.0104 (2)	0.2673 ± 0.0117 (8)	0.1639 ± 0.0122 (11)	0.1187 ± 0.0083 (9)	0.1596 ± 0.0132 (1)	6.11
ANN-ELECTRE	0.1120 ± 0.0299 (3)	0.0101 ± 0.0111 (1)	0.3363 ± 0.0298 (17)	0.2335 ± 0.0390 (15)	0.0668 ± 0.0097 (6)	0.3075 ± 0.0150 (17)	0.1809 ± 0.0116 (16)	0.2568 ± 0.0187 (16)	0.1653 ± 0.0186 (2)	10.33
ANN-OWA	0.1363 ± 0.0311 (8)	0.1207 ± 0.0230 (16)	0.2395 ± 0.0216 (5)	0.2577 ± 0.0175 (16)	0.0677 ± 0.0112 (7)	0.2651 ± 0.0119 (6)	0.1787 ± 0.0104 (15)	0.2629 ± 0.0065 (17)	0.1770 ± 0.0124 (14)	11.56
ANN-TOPSIS	0.1480 ± 0.0343 (11)	0.1374 ± 0.0235 (17)	0.2453 ± 0.0196 (6)	0.1020 ± 0.0148 (8)	0.2678 ± 0.0371 (18)	0.2871 ± 0.0141 (11)	0.2246 ± 0.0119 (18)	0.0880 ± 0.0066 (7)	0.2093 ± 0.0130 (17)	12.56
CR	0.1572 ± 0.0416 (14)	0.0464 ± 0.0281 (5)	0.2687 ± 0.0282 (10)	0.0577 ± 0.0251 (1)	0.0601 ± 0.0126 (4)	0.2844 ± 0.0306 (9)	0.1372 ± 0.0125 (1)	0.0376 ± 0.0059 (4)	0.1667 ± 0.0144 (3)	5.67
LR	0.1708 ± 0.0380 (18)	0.0626 ± 0.0247 (10)	0.2799 ± 0.0245 (14)	0.0654 ± 0.0150 (2)	0.0704 ± 0.0113 (10)	0.2851 ± 0.0303 (10)	0.1651 ± 0.0133 (13)	0.1360 ± 0.0101 (12)	0.1701 ± 0.0158 (8)	10.78
KLR-ply	0.1333 ± 0.0333 (7)	0.0835 ± 0.0264 (15)	0.2591 ± 0.0287 (7)	0.0728 ± 0.0159 (4)	0.1023 ± 0.0225 (17)	0.2926 ± 0.0151 (14)	0.1520 ± 0.0160 (5)	0.0328 ± 0.0057 (3)	0.1721 ± 0.0164 (11)	9.22
KLR-rbf	0.1692 ± 0.0382 (17)	0.0547 ± 0.0233 (7)	0.2599 ± 0.0301 (8)	0.0744 ± 0.0151 (5)	0.0682 ± 0.0121 (8)	0.2882 ± 0.0142 (12)	0.1493 ± 0.0165 (4)	0.0463 ± 0.0086 (5)	0.1693 ± 0.0130 (7)	8.11
MORE	0.1457 ± 0.0413 (9)	0.0489 ± 0.0226 (6)	0.2640 ± 0.0288 (9)	0.0751 ± 0.0178 (6)	0.0695 ± 0.0139 (9)	0.3037 ± 0.0180 (16)	0.1486 ± 0.0157 (3)	0.0215 ± 0.0053 (2)	0.1691 ± 0.0140 (6)	7.33
LMT	0.1473 ± 0.0406 (10)	0.0674 ± 0.0243 (13)	0.2717 ± 0.0295 (12)	0.0672 ± 0.0164 (3)	0.0709 ± 0.0135 (11)	0.2956 ± 0.0148 (15)	0.1545 ± 0.0142 (6)	0.0174 ± 0.0069 (1)	0.1671 ± 0.0167 (4)	8.33
META	0.1623 ± 0.0469 (15)	0.0675 ± 0.0237 (14)	0.2750 ± 0.0317 (13)	0.1781 ± 0.0237 (11)	0.1004 ± 0.0186 (15)	0.2056 ± 0.0173 (2)	0.1592 ± 0.0122 (9)	0.1483 ± 0.0095 (14)	0.1732 ± 0.0151 (12)	11.67
MIP	0.1627 ± 0.0426 (16)	0.0640 ± 0.0239 (12)	–	–	0.1018 ± 0.0155 (16)	0.1958 ± 0.0137 (1)	0.1422 ± 0.0154 (2)	–	–	13.22
UTADIS	0.1480 ± 0.0421 (12)	0.0230 ± 0.0238 (3)	0.2854 ± 0.0246 (16)	0.2090 ± 0.0236 (14)	0.0783 ± 0.0163 (13)	0.2342 ± 0.0171 (3)	0.1556 ± 0.0132 (7)	0.1324 ± 0.0117 (11)	0.1758 ± 0.0152 (13)	10.22
UTADIS-G	0.1553 ± 0.0413 (13)	0.0555 ± 0.0328 (8)	0.2850 ± 0.0219 (15)	0.1753 ± 0.0251 (10)	0.0771 ± 0.0148 (12)	0.3305 ± 0.0491 (18)	0.1877 ± 0.0247 (17)	0.1430 ± 0.0436 (13)	0.1796 ± 0.0271 (15)	13.44

Table 9
Classification performance in terms of the mean and standard deviation of 0/1 loss for 80% training data and 20% test data.

Method	DBS	CPU	BCC	MPG	ESL	ERA	LEV	CEV	MMG	Avg. rank
ANN-UTADIS	0.0645 ± 0.0542 (1)	0.0046 ± 0.0137 (1)	0.2056 ± 0.0389 (3)	0.1587 ± 0.0324 (9)	0.0436 ± 0.0180 (1)	0.2527 ± 0.0210 (4)	0.1447 ± 0.0144 (4)	0.1081 ± 0.0154 (8)	0.1608 ± 0.0302 (7)	4.22
ANN-PROMETHEE	0.0932 ± 0.0580 (6)	0.1080 ± 0.0775 (17)	0.2656 ± 0.0591 (11)	0.1949 ± 0.0378 (13)	0.0757 ± 0.0251 (14)	0.2814 ± 0.0317 (11)	0.1706 ± 0.0212 (14)	0.2234 ± 0.0195 (15)	0.1691 ± 0.0235 (11)	12.44
ANN-Ch-Uncons.	0.0864 ± 0.0540 (3)	0.0266 ± 0.0265 (5)	0.1816 ± 0.0348 (1)	0.0614 ± 0.0218 (4)	0.0482 ± 0.0186 (3)	0.2556 ± 0.0260 (6)	0.1517 ± 0.0204 (8)	0.0672 ± 0.0119 (6)	0.1595 ± 0.0263 (6)	4.67
ANN-Ch-Pos.	0.0909 ± 0.0526 (5)	0.0385 ± 0.0261 (9)	0.2191 ± 0.0456 (5)	0.2669 ± 0.0470 (17)	0.0500 ± 0.0207 (4)	0.2552 ± 0.0245 (5)	0.1518 ± 0.0217 (9)	0.1238 ± 0.0147 (11)	0.1595 ± 0.0295 (5)	7.78
ANN-Ch-Constr	0.0673 ± 0.0516 (2)	0.0380 ± 0.0285 (8)	0.1909 ± 0.0412 (2)	0.1853 ± 0.0393 (12)	0.0455 ± 0.0178 (2)	0.2587 ± 0.0252 (7)	0.1538 ± 0.0208 (10)	0.1124 ± 0.0148 (9)	0.1486 ± 0.0221 (1)	5.89
ANN-ELECTRE	0.0868 ± 0.0553 (4)	0.0061 ± 0.0116 (2)	0.3200 ± 0.0423 (17)	0.2242 ± 0.0486 (15)	0.0593 ± 0.0207 (7)	0.3010 ± 0.0397 (17)	0.1777 ± 0.0205 (16)	0.2492 ± 0.0281 (16)	0.1551 ± 0.0243 (2)	10.67
ANN-OWA	0.1064 ± 0.0604 (7)	0.0973 ± 0.0433 (16)	0.2169 ± 0.0399 (4)	0.2583 ± 0.0426 (16)	0.0569 ± 0.0216 (6)	0.2589 ± 0.0249 (8)	0.1740 ± 0.0250 (15)	0.2588 ± 0.0144 (17)	0.1670 ± 0.0239 (10)	11.00
ANN-TOPSIS	0.1076 ± 0.0626 (8)	0.1180 ± 0.0461 (18)	0.2224 ± 0.0340 (6)	0.0890 ± 0.0271 (8)	0.2469 ± 0.0554 (18)	0.2789 ± 0.0236 (9)	0.2172 ± 0.0238 (18)	0.0814 ± 0.0086 (7)	0.1987 ± 0.0268 (17)	12.11
CR	0.1416 ± 0.0681 (13)	0.0212 ± 0.0301 (4)	0.2496 ± 0.0485 (7)	0.0551 ± 0.0160 (1)	0.0542 ± 0.0218 (5)	0.2813 ± 0.0280 (10)	0.1314 ± 0.0176 (1)	0.0273 ± 0.0089 (4)	0.1584 ± 0.0251 (3)	5.33
LR	0.1616 ± 0.0743 (17)	0.0640 ± 0.0335 (14)	0.2773 ± 0.0548 (14)	0.0611 ± 0.0263 (2)	0.0660 ± 0.0203 (10)	0.2843 ± 0.0302 (12)	0.1627 ± 0.0249 (13)	0.1328 ± 0.0173 (12)	0.1657 ± 0.0232 (9)	11.39
KLR-ply	0.1265 ± 0.0663 (10)	0.0754 ± 0.0372 (15)	0.2569 ± 0.0506 (8)	0.0727 ± 0.0268 (5)	0.0922 ± 0.0279 (15)	0.2918 ± 0.0290 (15)	0.1472 ± 0.0231 (5)	0.0286 ± 0.0075 (5)	0.1741 ± 0.0246 (15)	10.33
KLR-rbf	0.1343 ± 0.0672 (12)	0.0405 ± 0.0284 (10)	0.2598 ± 0.0529 (10)	0.0740 ± 0.0284 (7)	0.0657 ± 0.0229 (9)	0.2905 ± 0.0312 (13)	0.1496 ± 0.0233 (7)	0.0239 ± 0.0066 (3)	0.1696 ± 0.0271 (12)	9.22
MORE	0.1242 ± 0.0609 (9)	0.0412 ± 0.0299 (11)	0.2570 ± 0.0463 (9)	0.0737 ± 0.0269 (6)	0.0661 ± 0.0219 (11)	0.2988 ± 0.0276 (16)	0.1397 ± 0.0214 (3)	0.0190 ± 0.0070 (2)	0.1645 ± 0.0235 (8)	8.33
LMT	0.1433 ± 0.0667 (14)	0.0338 ± 0.0352 (6)	0.2707 ± 0.0554 (13)	0.0614 ± 0.0251 (3)	0.0691 ± 0.0228 (12)	0.2910 ± 0.0290 (14)	0.1474 ± 0.0232 (6)	0.0089 ± 0.0047 (1)	0.1595 ± 0.0283 (4)	8.11
META	0.1592 ± 0.0698 (16)	0.0640 ± 0.0304 (14)	0.2677 ± 0.0547 (12)	0.1686 ± 0.0369 (11)	0.1001 ± 0.0297 (16)	0.2031 ± 0.0250 (2)	0.1616 ± 0.0222 (12)	0.1506 ± 0.0166 (14)	0.1698 ± 0.0279 (13)	12.17
MIP	0.1480 ± 0.0811 (15)	0.0598 ± 0.0315 (12)	–	–	0.1008 ± 0.0247 (17)	0.1856 ± 0.0260 (1)	0.1359 ± 0.0185 (2)	–	–	13.22
UTADIS	0.1280 ± 0.0501 (11)	0.0152 ± 0.0214 (3)	0.2913 ± 0.0510 (15)	0.2080 ± 0.0388 (14)	0.0744 ± 0.0235 (13)	0.2356 ± 0.0292 (3)	0.1572 ± 0.0222 (11)	0.1336 ± 0.0167 (13)	0.1734 ± 0.0265 (14)	10.78
UTADIS-G	0.1683 ± 0.0667 (18)	0.0356 ± 0.0386 (7)	0.3016 ± 0.0478 (16)	0.1617 ± 0.0383 (10)	0.0656 ± 0.0228 (8)	0.3259 ± 0.0567 (18)	0.1781 ± 0.0253 (17)	0.1166 ± 0.0217 (10)	0.1778 ± 0.0246 (16)	13.33

preference learning algorithm and their ability to differentiate between these approaches. For example, the 0/1 loss values attained by ANN-UTADIS for CPU, ESL, and DBS are 0.0046, 0.0436, and 0.0645, indicating the inconsistencies in the suggested assignments only for a marginal share of test data. On the other extreme, these values for ERA and BCC are 0.2527 and 0.2056, respectively, confirming an incorrect classification for a significant share of alternatives. When it comes to the differences between average 0/1 losses for the best and worst-performing algorithms, they are the least for MMG, LEV, and DBS, while being the greatest for CEV, MPG, ESL, and BCC.

The most favorable average ranks implied by the 0/1 loss for the nine datasets are attained by:

- ANN-UTADIS (4.22), which has the least 0/1 loss for DBS, CPU, and ESL, while being ranked in the upper half of the ranking for all problems;
- ANN-Ch-Uncons. (4.67), which is at the top for BCC, while being ranked in the top six for 8 out of 9 datasets;
- CR (5.33), which attains the lowest mean of 0/1 loss for LEV and MPG,
- ANN-Ch-Constr. (5.89) ranked first for MMP and second for BDS, CPU, and ESL.

On the other extreme, the worst average ranks are attained by UTADIS-G (13.33), MIP (13.22), ANN-PROMETHEE (12.44), META (12.17), ANN-TOPSIS (12.11), LR (11.39), ANN-OWA (11.00), UTADIS (10.78), and ANN-ELECTRE (10.67). Note that the differences between the average ranks for the approaches in the lower half of the ranking are lesser than in the case of AUC.

When it comes to the direct comparison of the approaches using similar preference methods in terms of the 0/1 loss, ANN-UTADIS performs better than UTADIS for all datasets except ERA and better than UTADIS-G for all considered problems; ANN-ELECTRE is more advantageous than META only for 4 out of 9 problems, whereas the algorithms using the Choquet integral attain similar results for CPU, ESL, LEV, and MMG. Moreover, CR was worse than the ANN-Ch methods on DBS and BCC, whereas ANN-Ch-Constr. and ANN-Ch-Pos. were underperforming for MPG and CEV. On average, the latter approach attained the worst average rank among these four methods, most likely due to the least flexible model admitting only positive interactions for pairs of criteria.

With the decrease in the number of alternatives in the training set relative to the test set, the 0/1 loss increases for almost all methods (see Tables 7 and 8). For example, for ANN-UTADIS and DBS, its values are equal to 0.0645 for 80% training data, 0.1093 for 50%, and 0.1460 for 20%. The analogous results attained by ANN-Ch-Constr. for BCC are 0.1816, 0.2196, and 0.2406. The least performance deterioration can be observed for ANN-PROMETHEE and ANN-ELECTRE for BCC, MPG, ERA, LEV, and CEV. In particular, for PROMETHEE-ANN and BCC, the respective 0/1 losses are 0.2656 for 80/20, 0.2709 for 50/50, and 0.2659 for 20/80. In general, the average ranks for ANN-UTADIS, ANN-Ch-Uncons., and LR get slightly worse with the decrease of the training set's share, whereas the ranks for LR, META, and MIP exhibit an inverse trend. In the case of META and MIP, this can be explained by the greater efficiency of these algorithms when dealing with smaller data sizes. For example, for the 20/80 division, MIP identified the solutions for 8 out of 9 datasets, whereas for greater training sets, it failed to identify a sorting model for the additional three problems.

The conclusions derived from the analysis of the 0/1 loss agree with the ones formulated for AUC. On the one hand, ANN-UTADIS, ANN-Ch-Uncons., and CR are the best performing algorithms, whereas MIP, META, TOPSIS, OWA, UTADIS, and UTADIS-G attain the least advantageous results. A noticeable difference concerns the performance of ANN-ELECTRE and ANN-PROMETHEE, which are among the best approaches in terms of AUC but are rated poorly when considering the 0/1 loss. This means that these two outranking-based methods correctly reproduce the preference relations for the vast majority of pairs of alternatives while making more mistakes concerning their classification. It can be explained given the nature of these methods and the learning process. ANN-ELECTRE and ANN-PROMETHEE incorporate the NFS procedure with a score for each alternative derived from pairwise comparisons against all remaining alternatives. However, these scores are transformed into assignments by comparing them with the class thresholds. It turns out that the threshold inferred for the training set might not generalize well for the test set, leading to the misclassification of alternatives, which attain scores close to the threshold. This is confirmed by Fig. 16, which indicates that for ERA, changing the threshold value for the test set rather than using the one inferred from the learning data might improve the 0/1 loss even by a few percent.

In the e-Appendix, we report the experimental results for the ANN-based algorithms in terms of the F1 score as well as the outcomes given different performance measures obtained for the training set.

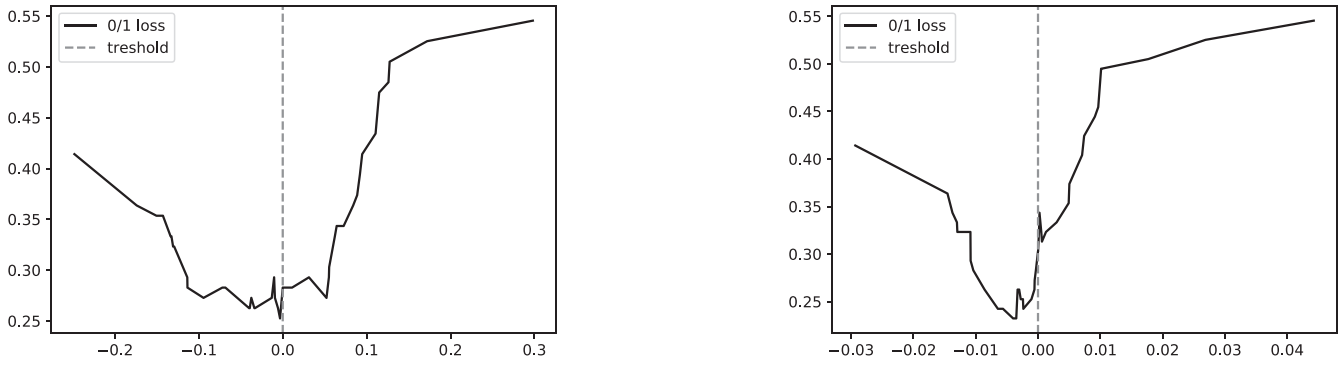
7. Conclusions and future work

The availability of data resources helps individuals and groups mine helpful information and make better-informed decisions. The spectrum of practical problems that emphasize handling large quantities of data becomes more extensive. This requires the development of dedicated techniques. In recent years, an often emphasized aspect is that such methods should support both the explainability of recommended decisions and the interpretability of the entire decision-making process.

In this paper, we have considered the problem of processing data into explainable and interpretable models. This has been done in the context of preference learning. It consists of training the models on a set of alternatives for which the preferences are known/available and predicting the preferences for all other options. Specifically, we considered learning the parameters of monotonic sorting models from large sets of assignment examples. In this kind of problem, alternatives need to be assigned to predefined, preference-ordered classes in the presence of multiple, potentially conflicting criteria.

We have advocated the use of intuitive models inspired by the development in the field of MCDA. This is consistent with the recent trends in Machine Learning (Rudin, 2019). The considered models offer measures for (i) quantifying the role of individual criteria and subsets of criteria, (ii) understanding the impact of particular performances on the decision, (iii) gaining insights on which performance differences are negligible, significant, or critical, and (iv) capturing the strength of criteria coalitions sufficient for claiming that one alternative is at least as good as another. Moreover, the applied operators offer a mathematically sound and elegant manner for aggregating the arguments supporting each alternative's strengths and weaknesses. Also, the considered threshold-based sorting procedure is easily understandable and transparent in deriving the assignments by comparing alternatives' comprehensive scores with the separating class thresholds.

As a concrete Machine Learning application of these models, we have proposed Artificial Neural Networks as a computation technique for conducting preference disaggregation. ANNs have been used before for classification in the context of extensive data. However, the non-linear models they derived could not be interpreted by human Decision Makers nor accepted by domain experts. Thanks to the suitably adjusted components, units, and architecture, we have made ANNs suitable for learning highly explainable models.



(a) ANN-PROMETHEE (b) ANN-ELECTRE

Fig. 16. The values of 0/1 loss (y-axis) for different separating class thresholds (x-axis) for the ERA problem.

The main benefits of the proposed preference learning algorithms are three-fold. First, we infer the parameters of the sorting models from decision examples, not requiring the Decision Maker to specify their values directly. We allow for simultaneous inference of all parameters of the sorting model, such as, e.g., criteria weights, concordance and discordance functions, and the comparison, veto, credibility, and separating class thresholds. This cannot be done efficiently with mathematical programming techniques that are traditionally applied in MCDA. Also, we avoid an arbitrary indication of meta-parameters such as shapes of preference functions or characteristic points of marginal values functions. In turn, we apply more general per-criterion (value, preference, concordance, or discordance) functions that offer greater flexibility in fitting the input data while maintaining the original spirit of MCDA.

Second, we contribute to the stream of making the MCDA methods suitable for handling inconsistent preference information that is too large to be dealt with by most traditional methods within an acceptable time. Sets of alternatives traditionally considered in MCDA consist of modestly-sized collections (Wallenius et al., 2008) and the development of the algorithms scaling up well with the number of alternatives has not been at the core of MCDA (Corrente et al., 2013). For example, the basic MCDA algorithms for dealing with inconsistency in the provided preference information are based on Mixed-Integer Linear Programming (MILP). Nonetheless, some existing MCDA and preference learning methods are capable of dealing with large inconsistent sets of assignment examples (see, e.g., Chandrasekaran et al., 2005; Dembczyński et al., 2009; Greco et al., 2001; Kotłowski & Słowiński, 2013; Manthoulis, Doumpos, Zopounidis, & Galariotis, 2020; Sobrie et al., 2019; Tehrani et al., 2012; Zopounidis & Doumpos, 2000). In this spirit, we demonstrate the feasibility of the proposed ANN-based approaches to the collections of over one thousand alternatives or the problems requiring comparing a few million pairs of alternatives. We know that the volume of datasets considered in some other sub-fields of ML is far more significant than in our experiments. Hence, demonstrating the usability of the proposed methods in areas typical for the ML applications remains a subject for future research. These include, e.g., finance, medicine, economy, and information retrieval, in which even some MCDA methods have been already used in the context of data sets with sizes exceeding those considered in this paper (e.g., bank failure prediction (Manthoulis et al., 2020), prognosis for hospice referral (Gil-Herrera et al., 2015), and recommender systems in numerous application domains (Manouselis & Costopoulou, 2007)).

Third, the extensive experiments on various benchmark problems indicate that the introduced algorithms are competitive in terms of predictive accuracy. This is particularly true for the three approaches called ANN-UTADIS, ANN-Ch-Uncons., and ANN-PROMETHEE. They incorporate preference models in the form of an additive value function with generalized marginal functions, 2-additive Choquet integral admitting significant variability of weights, and an outranking relation combined with the Net Flow Score procedure. These methods perform well in terms of the AUC measure, which focuses on preserving pairwise preference relations. In addition, ANN-UTADIS and ANN-Ch-Uncons. score favorably also on the 0/1 loss, which is directly related to the classification accuracy. On average, the predictions made by these algorithms were slightly more accurate than the recommendations delivered by the state-of-the-art methods, including logistic regression and its generalizations, rule ensemble methods, approaches based on mathematical programming, and a dedicated metaheuristic for an outranking-based classification model. The advantage of the ANN-based methods derives from a few factors, including incorporating more general preference functions, efficient optimization methods, and techniques for increasing noise resistance, preventing overfitting, and reducing the impact of the information processing order on the attained results.

From a broader perspective, the variability of different algorithms proposed in this paper gives a chance for adjusting the sorting model to the provided preference information, as postulated in Hanne (1997). In particular, we considered score-, distance-, and outranking-based approaches that admit different compensation levels, interactions between criteria, or per-criterion risk attitudes or curvatures of marginal functions. In MCDA, such factors need to be considered when selecting a single method a priori. However, in the preference learning context, all presented neural networks can be aggregated in a single ANN that would, in the end, activate only the part and underlying approach leading to the most advantageous results that fit the available indirect preferences in the best way.

The directions for future research can be divided into experimental and methodological. The former ones derive from the limitations of our study. First, some data sets considered in the experimental comparison involve nominal attributes arbitrarily transformed into monotonic criteria as described in Tehrani et al. (2012). While this increases the difficulty of the preference learning task, such an interpretation neglects the original performance scales without preference directions. In this perspective, we perceive the need to further test the preference learning algorithms on real-world data with correctly defined criteria and increase the variety of publicly available properly designed benchmark data sets. Second, when testing the performance of algorithms, we run only those originally proposed in this paper. For the remaining

methods, we recalled the results reported in the respective works (e.g., Sobrie et al., 2019; Tehrani et al., 2012) on the same benchmark problems. This could be questioned concerning the optimization of hyperparameters which is an essential component of the experimental study. We performed it differently than in Sobrie et al. (2019) and Tehrani et al. (2012). In particular, the performance of some algorithms (e.g., UTADIS) for which the results were reported in other works could be improved if their hyperparameters were set more carefully. Given this limitation, we want to emphasize the need for adopting proper processes for optimizing the hyperparameters of MCDA methods in future studies that will focus on performing comparative analyses. In our understanding, successfully implementing this postulate requires making the source code of all so far proposed methods in the preference learning stream publicly available. Third, when optimizing the parameters of the sorting model, one could investigate the impact of other misclassification errors than a sum of regrets or different techniques than AdamW.

Regarding future research related to the development of other methods, we envisage the following four directions. First, we will propose neural preference learning algorithms for other intuitive MCDA approaches. The most appealing ones include the ELECTRE (Costa, Rui Figueira, Vieira, & Vieira, 2019) and PROMETHEE (Pelissari, Oliveira, Amor, & Abackerli, 2019) methods with boundary or characteristic class profiles and value-based approaches admitting interactions between criteria (Liu et al., 2021) and non-monotonicity (Liu et al., 2019) of marginal value functions. Second, it is possible to combine different methods within a single neural network and aggregate their results into a comprehensive quality measure. The form of an aggregation operator and the weights associated with scores delivered by various approaches could be learned during the optimization process (Hanne, 1997). Third, it would be interesting to verify the impact of using an ensemble of models that attained a pre-defined threshold of the classification error. In this paper, we only used the model that performed the best during learning. However, some other models were only slightly worse, and their joint use on the test set could increase the robustness of recommended assignments. Finally, an appealing idea consists of adjusting the preference learning algorithms to an online setting (Sahoo, Pham, Lu, & Hoi, 2018). Unlike batch learning applied in this paper, it assumes preferences are provided in sequential order, and the method needs to update the classification model at each step. This would correspond to a common MCDA scenario in which the DM provides preferences in successive iterations.

Acknowledgments

The authors acknowledge financial support from the Polish National Science Center under the SONATA BIS project (grant no. DEC-2019/34/E/HS4/00045).

Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.ejor.2022.06.053

References

Alvarez, P. A., Ishizaka, A., & Martínez, L. (2021). Multiple-criteria decision-making sorting methods: A survey. *Expert Systems with Applications, 183*, 115368.

Angilella, S., Corrente, S., Greco, S., & Słowiński, R. (2013). Multiple criteria hierarchy process for the Choquet integral. In R. Purshouse, P. Fleming, C. Fonseca, S. Greco, & J. Shaw (Eds.), *Evolutionary multi-criterion optimization – 7th international conference, EMO 2013, Sheffield, UK, March 19–22, 2013. Proceedings* (pp. 475–489). Springer.

Brans, J. P., & De Smet, Y. (2016). PROMETHEE methods. In S. Greco, M. Ehrgott, & J. R. Figueira (Eds.), *Multiple criteria decision analysis: State of the art surveys* (pp. 187–219). New York, NY: Springer.

Chandrasekaran, R., Ryu, Y. U., Jacob, V. S., & Hong, S. (2005). Isotonic separation. *INFORMS Journal on Computing, 17*(4), 462–474.

Cinelli, M., Kadziński, M., Miebs, G., Gonzalez, M., & Słowiński, R. (2022). Recommending multiple criteria decision analysis methods with a new taxonomy-based decision support system. *European Journal of Operational Research, 302*(2), 633–651.

Corrente, S., Greco, S., Kadziński, M., & Słowiński, R. (2013). Robust ordinal regression in preference learning and ranking. *Machine Learning, 93*(2), 381–422.

Costa, A. S., Rui Figueira, J., Vieira, C. R., & Vieira, I. V. (2019). An application of the ELECTRE TRI-c method to characterize government performance in OECD countries. *International Transactions in Operational Research, 26*(5), 1935–1955.

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems, 2*(4), 303–314.

Dembczyński, K., Kotłowski, W., & Słowiński, R. (2006). Additive preference model with piecewise linear components resulting from dominance-based rough set approximations. In L. Rutkowski, R. Tadeusiewicz, L. A. Zadeh, & J. M. Żurada (Eds.), *Artificial intelligence and soft computing – ICAISC 2006* (pp. 499–508). Berlin, Heidelberg: Springer.

Dembczyński, K., Kotłowski, W., & Słowiński, R. (2009). Learning rule ensembles for ordinal classification with monotonicity constraints. *Fundamenta Informaticae, 94*, 163–178.

Deng, L., & Yu, D. (2014). Deep learning: Methods and applications. *Foundations and Trends in Signal Processing, 7*(3-4), 197–387.

Doshi-Velez, F., & Kim, B. (2017). Towards a rigorous science of interpretable machine learning. arXiv:1702.08608.

Doumpos, M., Marinakis, Y., Marinaki, M., & Zopounidis, C. (2009). An evolutionary approach to construction of outranking models for multicriteria classification: The case of the ELECTRE TRI method. *European Journal of Operational Research, 199*(2), 496–505.

Doumpos, M., & Zopounidis, C. (2004). Developing sorting models using preference disaggregation analysis: An experimental investigation. *European Journal of Operational Research, 154*(3), 585–598.

Doumpos, M., & Zopounidis, C. (2011). Preference disaggregation and statistical learning for multicriteria decision support: A review. *European Journal of Operational Research, 209*(3), 203–214.

Doumpos, M., & Zopounidis, C. (2018). Disaggregation approaches for multicriteria classification: An overview. In N. Matsatsinis, & E. Grigoroudis (Eds.), *Preference disaggregation in multiple criteria decision analysis: Essays in honor of Yannis Siskos* (pp. 77–94). Cham: Springer.

Figueira, J., Greco, S., Roy, B., & Słowiński, R. (2013). An overview of ELECTRE methods and their recent extensions. *Journal of Multi-Criteria Decision Analysis, 20*(1-2), 61–85.

Fürnkranz, J., & Hüllermeier, E. (2011). Preference learning: An introduction. In J. Fürnkranz, & E. Hüllermeier (Eds.), *Preference learning* (pp. 1–17). Berlin, Heidelberg: Springer.

Gil-Herrera, E., Aden-Buie, G., Yalcin, A., Tsalatsanis, A., Barnes, L. E., & Djulbegovic, B. (2015). Rough set theory based prognostic classification models for hospice referral. *BMC Medical Informatics and Decision Making, 15*(1), 98.

Goodman, B., & Flaxman, S. (2017). European union regulations on algorithmic decision-making and a "right to explanation". *AI Magazine, 38*(3), 50–57.

Greco, S., Matarazzo, B., & Słowiński, R. (2001). Rough sets theory for multicriteria decision analysis. *European Journal of Operational Research, 129*(3), 1–47.

Greco, S., Mousseau, V., & Słowiński, R. (2010). Multiple criteria sorting with a set of additive value functions. *European Journal of Operational Research, 207*, 1455–1470.

Guo, M., Zhang, Q., Liao, X., Chen, F. Y., & Zeng, D. D. (2021). A hybrid machine learning framework for analyzing human decision-making through learning preferences. *Omega, 101*, 102263.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (2009). The WEKA data mining software: An update. *ACM SIGKDD Explorations Newsletter, 11*(1), 10–18.

Hanne, T. (1997). Decision support for MCDM that is neural network-based and can learn. In J. Clímaco (Ed.), *Multicriteria analysis* (pp. 401–410). Berlin, Heidelberg: Springer.

Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2000). *Applied logistic regression*. New York: Wiley.

Hu, Y. C. (2009). Bankruptcy prediction using ELECTRE-based single-layer perceptron. *Neurocomputing, 72*(13-15), 3150–3157.

Hwang, C. L., & Yoon, K. (1981). Multiple attribute decision making: Methods and applications a state-of-the-art survey. In *Methods for multiple attribute decision making* (pp. 58–191)). Berlin, Heidelberg: Springer.

Kadziński, M., & Szczepański, A. (2022). Learning the parameters of an outranking-based sorting model with characteristic class profiles from large sets of assignment examples. *Applied Soft Computing, 116*, 108312.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv:1412.6980.

Köksalan, M., & Özpeynirci, S. B. (2009). An interactive sorting method for additive utility functions. *Computers and Operations Research, 36*(9), 2565–2572.

Kotłowski, W., & Słowiński, R. (2013). On nonparametric ordinal classification with monotonicity constraints. *IEEE Transactions on Knowledge and Data Engineering, 25*(11), 2576–2589.

Landwehr, N., Hall, M., & Frank, E. (2003). Logistic model trees. In N. Lavrac, D. Gamberger, H. Blockeel, & L. Todorovski (Eds.), *Machine learning: ECML 2003 – 14th european conference on machine learning, Cavtat-Dubrovnik, Croatia, September 22–26, 2003. Proceedings* (pp. 241–252). Springer.

Leroy, A., Mousseau, V., & Pirlot, M. (2011). Learning the parameters of a multiple

criteria sorting method. In R. Brafman, F. Roberts, & A. Tsoukias (Eds.), *Algorithmic decision theory – second international conference, ADT 2011, Piscataway, NJ, USA, October 26–28, 2011. Proceedings* (pp. 219–233). Springer.

Linkov, I., Galaitsi, S., Trump, B. D., Keisler, J. M., & Kott, A. (2020). Cybertrust: From explainable to actionable and interpretable artificial intelligence. *Computer, 53*(9), 91–96.

Liu, J., Kadziński, M., Liao, X., & Mao, X. (2021). Data-driven preference learning methods for value-driven multiple criteria sorting with interacting criteria. *INFORMS Journal on Computing, 33*(2), 586–606.

Liu, J., Liao, X., Kadziński, M., & Słowiński, R. (2019). Preference disaggregation within the regularization framework for sorting problems with multiple potentially non-monotonic criteria. *European Journal of Operational Research, 276*(3), 1071–1089.

Loshchilov, I., & Hutter, F. (2018). Fixing weight decay regularization in adam. https://openreview.net/forum?id=rk6qdGgCZ.

Malakooti, B., & Zhou, Y. Q. (1994). Feedforward artificial neural networks for solving discrete multiple criteria decision making problems. *Management Science, 40*(11), 1542–1561.

Manouselis, N., & Costopoulou, C. (2007). Analysis and classification of multi-criteria recommender systems. *World Wide Web, 10*(4), 415–441.

Manthoulis, G., Doumpos, M., Zopounidis, C., & Galariotis, E. (2020). An ordinal classification framework for bank failure prediction: Methodology and empirical evidence for US banks. *European Journal of Operational Research, 282*(2), 786–801.

Molnar, C. (2020). *Interpretable machine learning*. Lulu Press.

Mousseau, V., & Dias, L. (2004). Valued outranking relations in ELECTRE providing manageable disaggregation procedures. *European Journal of Operational Research, 156*(2), 467–482.

Olteanu, A. L., & Meyer, P. (2014). Inferring the parameters of a majority rule sorting model with vetoes on large datasets. In V. Mousseau, & M. Pirlot (Eds.), *DA2PL 2014: From multicriteria decision aid to preference learning* (pp. 87–94).

Pelissari, R., Oliveira, M. C., Amor, S. B., & Abackerli, A. J. (2019). A new flow-sort-based method to deal with information imperfections in sorting decision–making problems. *European Journal of Operational Research, 276*(1), 235–246.

Roy, B. (2010). Two conceptions of decision aiding. *International Journal of Multicriteria Decision Making, 1*(1), 74–79.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. arXiv:1609.04747.

Rudin, C. (2019). Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence, 1*(5), 206–215.

Sahoo, D., Pham, Q., Lu, J., & Hoi, S. C. (2018). Online deep learning: Learning deep neural networks on the fly. In *Proceedings of the 27th international joint conference on artificial intelligence* (pp. 2660–2666).

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data, 6*(1), 1–48.

Sobrie, O., Mousseau, V., & Pirlot, M. (2019). Learning monotone preferences using a majority rule sorting model. *International Transactions in Operational Research, 26*(5), 1786–1809.

Tehrani, A. F., Cheng, W., Dembczyński, K., & Hüllermeier, E. (2012). Learning monotone nonlinear models using the Choquet integral. *Machine Learning, 89*(1), 183–211.

Waegeman, W., De Baets, B., & Boullart, L. (2009). Kernel-based learning methods for preference aggregation. *4OR, 7*(2), 169–189.

Wallenius, J., Dyer, J. S., Fishburn, P. C., Steuer, R. E., Zionts, S., & Deb, K. (2008). Multiple criteria decision making, multiattribute utility theory: Recent accomplishments and what lies ahead. *Management Science, 54*(7), 1336–1349.

Yager, R. R. (1988). On ordered weighted averaging aggregation operators in multicriteria decisionmaking. *IEEE Transactions on Systems, Man, and Cybernetics, 18*(1), 183–190.

Zheng, S., Song, Y., Leung, T., & Goodfellow, I. (2016). Improving the robustness of deep neural networks via stability training. In L. Agapito, T. Berg, J. Kosecka, & L. Zelnik-Manor (Eds.), *2016 IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 4480–4488). IEEE.

Zopounidis, C., & Doumpos, M. (2000). PREFDIS: A multicriteria decision support system for sorting decision problems. *Computers and Operations Research, 27*(7-8), 779–797.