Preference learning II

1 Introduction

In the first part of this laboratories regarding preference learning we focused on machine learning methods for DM's preference learning. In the second part there will be presented MCDA methods for founding parameters for neural network. There will be shown three methods:

1. ANN-Ch-Constr.
2. ANN-UTADIS
3. ANN-Promethee

During our laboratories we will consider sorting problems which means classification problems with preference-ordered classes. These solutions base on the utility function $Sc(a_i)$ and assigning them to proper classes will be done with the usage of thresholds t_h which are boundaries between classes C_h:

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

2 ANN-Ch-Constr.

This method bases on **Choquet integral** [1], which is an aggregating method which takes into account interactions between criteria. It has a form of a weighted

sum over all criteria subsets $T \subseteq G$ where value for a considered subset of criteria is equal to minimal scores' value on this criteria.

$$Ch_{\mu}(a_i) = \sum_{T \subseteq G} w_T \cdot \min_{j \in T} g_j(a_i), \tag{2}$$

where $g_j(a_i)$ is an evaluation of alternative a_i on criterion j.

Due to practical issues, usually the space of solutions is limited to take into account only interactions between pairs of criteria. Then we can state above Equation 2 as 2-additive Möbius transformation:

$$Ch_{\mu,2}(a_i) = \sum_{j=1}^{m} w_j g_j(a_i) + \sum_{\{j,l\} \subseteq G} w_{\{j,l\}} min(g_j(a_i), g_l(a_i)). \tag{3}$$

where w_j means the weight of criterion j and $w_{\{i,j\}}$ means weight for pair of criteria i and j. If weight is greater than zero, then we say about positive interaction between criteria - **synergy**. If value of a weight is less than zero, there is a negative interaction between criteria - **redundancy**.

In order to preserve the direction of preference on criteria, there need to be introduced additional constraints on weights values. Firstly, criteria weights cannot change the directions of preference for a considered criterion, which means that all the weight need to be greater than zero:

$$w_j \geq 0, \forall j \in \{1, \ldots, m\}. \tag{4}$$

The weight for pairs of criteria may be positive or negative in order to allow to positive and negative interactions. However, the sum of weights for each criterion cannot be lower than zero:

$$w_{\{j,l\}} + w_j \geq 0, \quad \forall j \in \{1, \ldots, m\}, \forall l \in \{j+1, \ldots, m\}. \tag{5}$$

The last element of this method is weights normalization, to sum them up to 1:

$$Sc_{ANN-Ch-Constr.}(a_i) = \frac{C_{\mu,2}(a_i)}{\sum_{j=1}^{m} w_j + \sum_{\{j,l\}\subseteq G} w_{\{j,l\}}}. \tag{6}$$

The above equations may be shown as a simple neural network (Figure 1). Each object which is an input to the model needs to be firstly preprocessed by Möbius transform and then data goes to one out of two linear layers:

- first one which calculates a weighted sum for each criterion with the constraints coming from Equation 4,
- second one which calculates interactions between criteria with constraints coming from Equation 5.

The results from both layers are summed, normalized and then there is run a classification with the usage of thresholds t, which are also optimized during the neural network training.

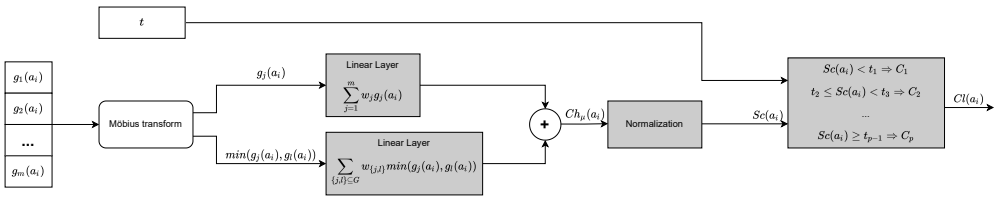

Figure 1: The architecture of neural network which implements ANN-Ch-Constr method.

The interpretation of criteria relevance may be calculated with the usage of **Shapley's index**, which takes into account both criteria weights and interactions between criteria.

$$\varphi(i) = w_i + \sum_{\{i,l\}\subseteq G} \frac{w_{\{i,l\}}}{2}. \tag{7}$$

3 Monotone block

In order to build more complex neural networks, we need to introduce a new layer which will be able to calculate monotonic transformations. Using the general equation for neural network with one hidden layer:

$$u(\boldsymbol{x}) = \sum_{k=1}^{L} \alpha_k \sigma(y_k^T \boldsymbol{x} + \theta_k), \tag{8}$$

we can easily notice that it can implement any monotonic function if weights α_k and y_k are non-negative values and activation function σ is a sigmoidal function. The examples of such a function are sigmoid and hardsigmoid. However, both these functions have small gradient values or areas where it is equal to zero. It causes that using these functions will result in problems with vanishing gradient even for not deep neural networks. To avoid this problem, we will use monotonic function **LeakyHardSigmoid**:

$$LeakyHardSigmoid(x) = \begin{cases} \delta x, & if\ x < 0, \\ x, & if\ 0 \leq x \leq 1, \\ \delta(x-1)+1, & if\ x > 1, \end{cases} \tag{9}$$

where δ is a small value in [0,1).

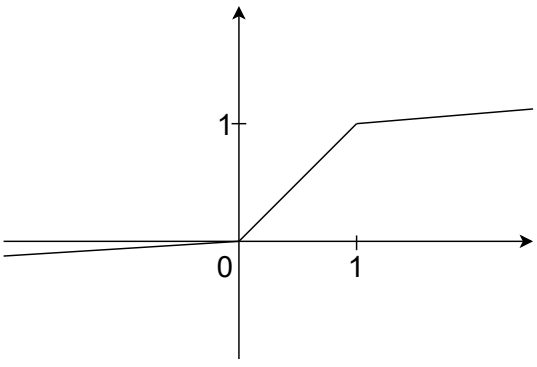

Figure 2: LeakyHardSigmoid

The neural network where weights α_k and y_k are non-negative values and activation function LeakyHardSigmoid, which input is a single feature, will be called **Monotone Block** with L components:

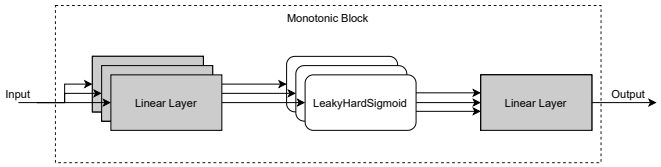

Figure 3: The architecture of monotone block.

For example this neural network with 3 components may make the following transformation:

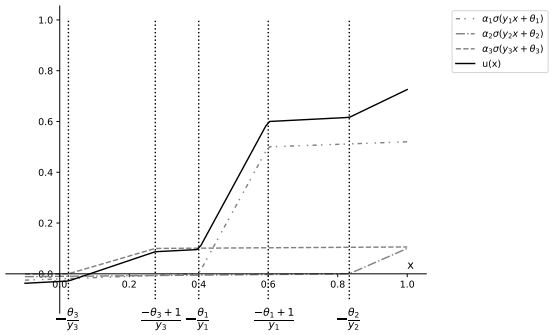

Figure 4: The plot of monotonic function which was created with the usage of monotone block.

4 ANN-UTADIS

The ANN_UTADIS method in an implementation of **UTADIS** method [2] which is a method of preference aggregation:

$$U(a_i) = \sum_{j=1}^{m} w_j u_j(g_j(a_i)), \tag{10}$$

where $u_j(g_j(a_i))$ is a monotonic function.
Using the previously defined monotone block, the neural network which implements this method is shown on Figure 5.

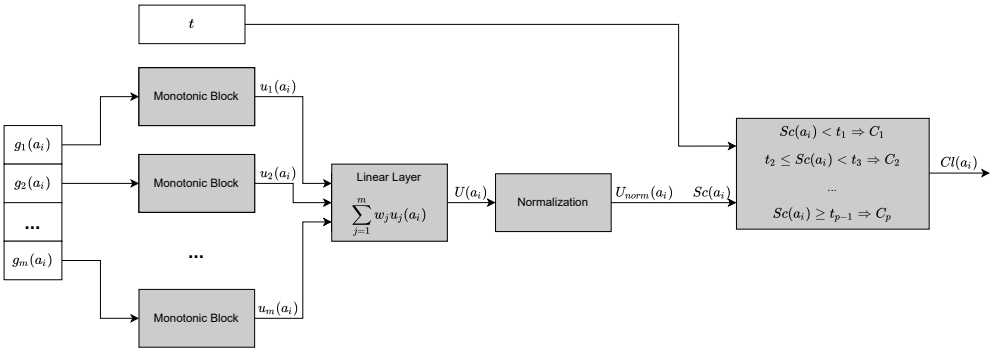

Figure 5: The architecture of neural network which implements ANN-UTADIS method.

As the presented monotone block is not normalized, it is required to normalize the global utility in order to get values between 0 and 1. It may be done by min-max normalization with utility values of ideal alternative a^+ and anti-ideal alternative a^-, which stand for artificial alternatives with the best and the worst values on all criteria:

$$Sc_{ANN-UTADIS}(a_i) = \frac{U(a_i) - U(a^-)}{U(a^+) - U(a^-)}. \tag{11}$$

5 ANN-Promethee

The **Promethee** method [3] bases on the getting knowledge from preference relations between alternatives.

The implementation of the method is very similar to ANN-UTADIS method. The only difference is the fact that as an input for the neural network we give differences in values for each pair of alternatives.

In the Promethee method, if the difference is 0 or less, the preference function is always 0. This means that all differences below zero can be replaced with the value 0. This can be done using e.g. the ReLU function on the input.

In order to aggregate te preference relation between alternatives, we use **NetFlowScore** method which builds 2 rankings: positive and negative basing on the fact how the considered alternative outranks others and is outranked by other alternatives. These two rankings are finally aggregated to one final ranking of alternatives.

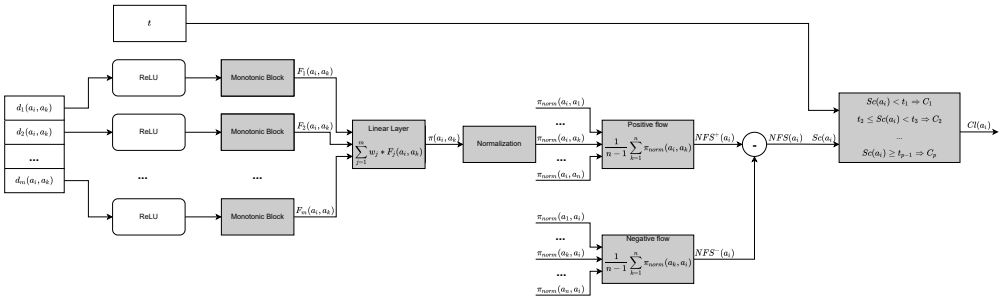

Figure 6: The architecture of neural network which implements ANN-Promethee method.

6 Optimization

In order to find optimal parameter values for methods, it is required to train neural networks. The error function which is minimized during the optimization is the

function of mean regret:

$$Minimize : loss = \frac{1}{|A^R|} \sum_{a_i^* \in A^R} regret(a_i^*), \tag{12}$$

where:

$$regret(a_i^*) = \max\{t_{C_{DM}(a_i^*)} - Sc(a_i^*),\ Sc(a_i^*) - t_{C_{DM}(a_i^*)+1},\ 0\}. \tag{13}$$

which means the distance of utility of alternative which is misclassified to the nearest threshold of its target class.

Attention: to avoid bias connected with the order of alternatives given during the training, there is a need to use a batch equal to the size of elements or as big as it is possible if the data does not fit the memory.

These methods require normalized criteria values to the interval [0-1] where 1 means the most preferred value and 0 is the least preferred.

The ANN-Ch-Constr., ANN-UTADIS and ANN-Promethee methods are presented in notebooks for solving a sorting problem with 2 classes for a problem Lectures Evaluation.

References

[1] Angilella, S., Corrente, S., Greco, S., & Słowiński, R. (2013, March). Multiple criteria hierarchy process for the Choquet integral. In International Conference on Evolutionary Multi-Criterion Optimization (pp. 475-489). Springer, Berlin, Heidelberg.

[2] Zopounidis, C., & Doumpos, M. (1999). A multicriteria decision aid methodology for sorting decision problems: The case of financial distress. Computational Economics, 14(3), 197-218.

[3] Brans, J. P., & De Smet, Y. (2016). PROMETHEE methods. In Multiple criteria decision analysis (pp. 187-219). Springer, New York, NY.