Uczenie preferencji II

1 Wstęp

W pierwszej części laboratoriów dotyczących uczenia preferencji skupiliśmy się na metodach uczenia maszynowego dla uczenia preferencji DM. W drugiej części zostaną przedstawione metody MCDA do wyznaczania parametrów dla sieci neuronowej. Przedstawione zostaną trzy metody:

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

Podczas naszych laboratoriów będziemy rozważać problemy sortowania, czyli problemy klasyfikacyjne z klasami uporządkowanymi pod względem preferencji. Rozwiązania te bazują na funkcji użyteczności $Sc(a_i)$, a przyporządkowanie ich do odpowiednich klas odbywać się będzie za pomocą progów t_h będących granicami pomiędzy klasami 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.

Metoda ta opiera się na **Choquet integral** [1], która jest metodą agregacji uwzględniającą interakcje między kryteriami. Ma ona postać sumy ważonej nad wszystkimi podzbiorami kryteriów $T \subseteq G$, gdzie wartość dla podzbioru kryteriów jest równa minimalnej wartości ocen tego podzbioru.

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

gdzie $g_j(a_i)$ jest oceną wariantu a_i na kryterium j.

Ze względu na kwestie praktyczne, zwykle ograniczamy przestrzeń możliwych rozwiązać do rozwiazań uwzględniajacych jedynie interakcjie pomiędzy parami kryteriów. Oznacza to że możemy zastosować 2-addytywną reprezentację Möbiusa 2:

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

gdzie w_j oznacza wagę kryterium j, a $w_{i,j}$ oznacza wagę dla pary kryteriów i i j. Jeśli waga jest większa od zera, to mówimy o pozytywnym oddziaływaniu między kryteriami - synergii. Jeśli wartość wagi jest mniejsza od zera, to mówimy o negatywnej interakcji między kryteriami - **redundancja**.
W celu zachowania kierunku preferencji na kryteriach należy wprowadzić dodatkowe ograniczenia na wartości wag. Po pierwsze, wagi kryteriów nie mogą zmieniać kierunków preferencji dla danego kryterium, co oznacza, że wszystkie wagi muszą być większe od zera:

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

Waga dla par kryteriów może być dodatnia lub ujemna, aby umożliwić pozytywne i negatywne interakcje. Suma wag dla każdego kryterium nie może być jednak mniejsza niż zero:

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

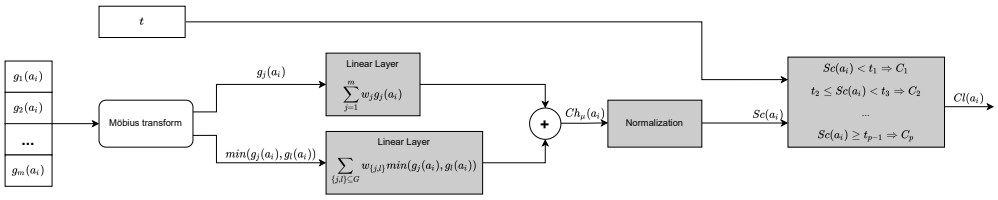
Ostatnim elementem jest normalizacja wag, czyli ich suma musi wynosić 1:

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

Powyższe równania można przedstawić w postaci prostej sieci neuronowej (rys. 1). Każdy obiekt będący wejściem do modelu musi być najpierw wstępnie przetworzony za pomocą transformaty Möbiusa, a następnie dane trafiają do jednej z dwóch warstw liniowych:

- pierwsza, który oblicza sumę ważoną dla każdego kryterium z ograniczeniami pochodzącymi z równania 4,
- drugi, który oblicza wartość interakcji pomiędzy kryteriami z ograniczeniami pochodzącymi z równania 5.

Wyniki z obu warstw są sumowane, normalizowane, a następnie przeprowadzana jest klasyfikacja z wykorzystaniem progów t, które również są optymalizowane podczas treningu sieci neuronowej.



Rysunek 1: Architektura sieci neuronowej realizującej metodę ANN-Ch-Constr.

Interpretacja tego modelu jest identyczna jak interpretacja oryginalnej metody. W zwiazku z powyższym aby obliczyć istotność kryteriówwykorzystujemy index **Shapley'a**, który bierze pod uwagę zarówno wagi kryteriów jak również interakcje między kryteriami:

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

3 Optymalizacja

W celu znalezienia optymalnych wartości parametrów metod wymagane jest wytrenowanie sieci neuronowych. Funkcją błędu, która jest minimalizowana podczas optymalizacji jest funkcja średniego żalu:

$$Minimize : loss = \frac{1}{|A^R|} \sum_{a_i^* \in A^R} regret(a_i^*), \tag{8}$$

gdzie:

$$regret(a_i^*) = \max\{t_{C_{DM}(a_i^*)} - Sc(a_i^*),\ Sc(a_i^*) - t_{C_{DM}(a_i^*)+1},\ 0\}. \tag{9}$$

co oznacza odległość jakości wariantu źle sklasyfikowanego do progu jego klasy docelowej.

Uwaga: aby uniknąć stronniczości związanej z kolejnością wariantów podawanych podczas treningu, konieczne jest stosowanie batcha zawierającego wszystkie warianty. W przypadku gdy wszystkie warianty nie mieszczą sięw pamięci to należy wykorzystywać możliwe duże rozmiary batcha
Zakładamy ze dane wejściowe są znormalizowane do do przedziału [0-1], gdzie 1 oznacza wartość najbardziej preferowaną, a 0 najmniej preferowaną.

4 Blok monotoniczny

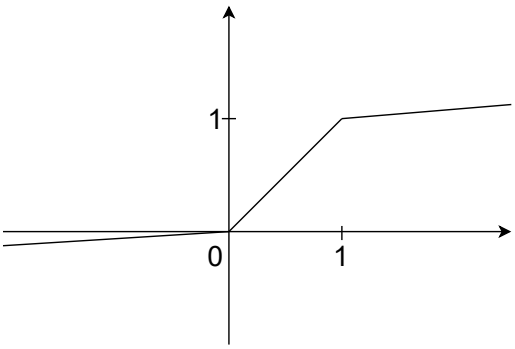
Aby zbudować bardziej złożone sieci neuronowe, musimy zdefiniować nową warstwę, która będzie potrafiła wykonać dowolne przekształcenie monotoniczne. Jedną z głównych własności sieci neuronowych jest fakt iż sieć neuronowa z pojedyńczą warstwą ukrytą i sigmoidalną funkcją aktywacji moze aproksymować dowolną ciągłą funkcję. Dokładność aproksymacji zależy wyłącznie od liczby neuronów w warstwie ukrytej:

$$u(\boldsymbol{x}) = \sum_{k=1}^{L} \alpha_k \sigma(y_k^T \boldsymbol{x} + \theta_k), \tag{10}$$

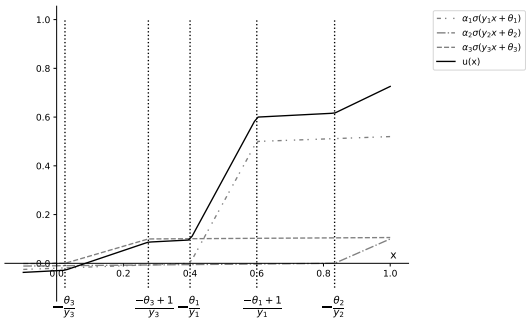
Możemy łatwo zauważyć, że może on realizować dowolną funkcję niemalejącą, jeśli wagi α_k i y_k są wartościami nieujemnymi, a funkcja aktywacji σ jest funkcją sigmoidalną. Przykładami takiej funkcji są funkcje sigmoid i hardsigmoid. Jednak gradient bu tych funkcji ma niewielkie wartości a w niektórych obszarach wynosi on zero z tego powodu cierpią one na problem zaniku gradientu. W tym celu aby łatwiej uczyć sieć neuronową wykorzystamy inną funkcję **LeakyHardSigmoid**:

$$LeakyHardSigmoid(x) = \begin{cases} \delta x, & if\ x < 0, \\ x, & if\ 0 \leq x \leq 1, \\ \delta(x-1)+1, & if\ x > 1, \end{cases} \tag{11}$$

gdzie δ jest niewielką wartością nachylenia [0,1).

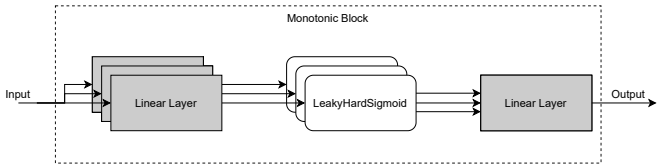


(a) LeakyHardSigmoid



(b) Przykład przekształcenia zmiennej przez blok monotoniczny zawierający 3 składowe.

Sieć neuronową, w której wagi α_k i y_k są wartościami nieujemnymi, a funkcją aktywacji jest LeakyHardSigmoid, której wejściem jest pojedyncza cecha, będziemy nazywać **blokiem monotonicznym** z L składowymi:



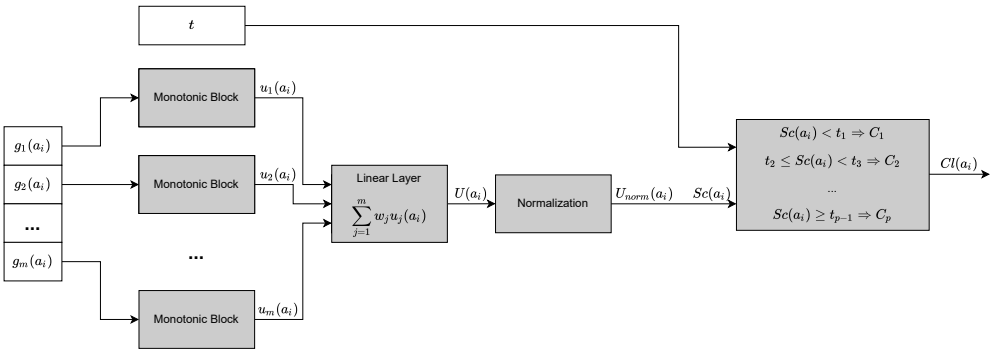
Rysunek 3: Schemat bloku monotonicznego

5 ANN-UTADIS

Metoda ANN-UTADIS jest implementacją metody UTADIS [2] będącej modelem addytywnej funkcji wartości:

$$U(a_i) = \sum_{j=1}^{m} w_j u_j(g_j(a_i)), \tag{12}$$

gdzie $u_j(g_j(a_i))$ jest cząstkową monotoniczną funkcją wartości.
Wykorzystując zdefiniowany wcześniej blok monotoniczny, sieć neuronowa realizująca tę metodę została przedstawiona na rysunku 4.



Rysunek 4: Architektura sieci neuronowej realizującej metodę ANN-UTADIS.

Ponieważ przedstawiony blok monotoniczny nie jest znormalizowany, wymagane jest znormalizowanie użyteczności globalnej w celu uzyskania wartości z przedziału [0,1]. Można to zrobić poprzez normalizację min-max z wartościami użyteczności alternatywy idealnej a^+ i antyidealnej a^-, które oznaczają sztuczne alternatywy o najlepszych i najgorszych wartościach na wszystkich kryteriach:

$$Sc_{ANN-UTADIS}(a_i) = \frac{U(a_i) - U(a^-)}{U(a^+) - U(a^-)}. \tag{13}$$

6 ANN-PROMETHEE

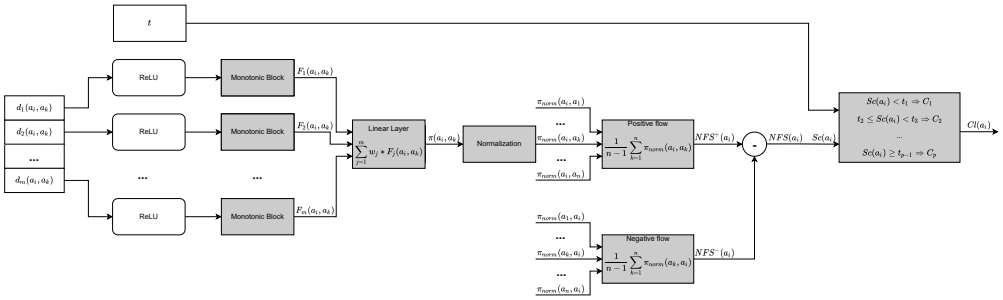
Metoda **PROMETHEE** [3] agreguje wyniki porównań parami każdgo wariantu z wszystkimi pozostałymi w całościowy indeks preferencji:

$$\pi(a_i, a_k) = \sum_{j=1}^{m} w_j F_j(d_j(a_i, a_k)), \tag{14}$$

Implementacja metody jest bardzo podobna do metody ANN-UTADIS. Jedyną różnicą jest fakt, że danymi wejściowymi dla sieci neuronowej są różnice w ocenach dla każdej pary wariantów.

W metodzie PROMETHEE, jeśli różnica ocen wynosi 0 lub mniej, to preferencja cząstkowa zawsze wynosi 0. Oznacza to, że wszystkie różnice poniżej zera można zastąpić wartością 0. Można to uzyskac wykorzystując funkcję ReLU.

W celu zagregowania relacji preferencji pomiędzy wariantami wykorzystujemy metodę **NetFlow-Score**, która buduje 2 rankingi: pozytywny i negatywny w oparciu o fakt, jak dany wariant przewyższy inne oraz jak jest przewyższany przez inne warianty. Te dwa rankingi są ostatecznie agregowane do jednej wartosci określajacej całościową jakość wariantu.



Rysunek 5: Architektura sieci neuronowej realizującej metodę ANN-PROMETHEE.

Literatura

[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.