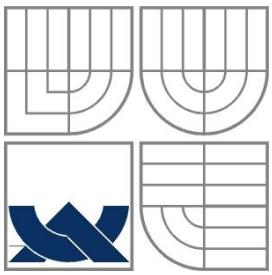
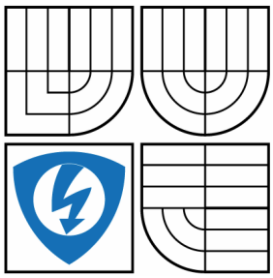


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

FUZZY SYSTÉMY S NETRADIČNÍMI ANTECEDENTY FUZZY PRAVIDEL

FUZZY SYSTEMS WITH UNTRADITIONAL ANTECEDENTS

DIPLOMOVÁ PRÁCE
MASTER‘S THESIS

AUTOR PRÁCE — Bc. Ondřej Klapil
AUTHOR

VEDOUCÍ PRÁCE — prof. Ing. Pavel Jura, CSc.
SUPERVISOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student: Bc. Ondřej Klapil ***ID:*** 125475
Ročník: 2 ***Akademický rok:*** 2014/2015

NÁZEV TÉMATU:

Fuzzy systémy s netradičními antecedenty fuzzy pravidel

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je prověřit možnosti tzv. cloud přístupu ve vstupně-výstupním datovém prostoru fuzzy systémů.

1. Prostudujte možnosti tzv. cloud přístupu ve vstupně-výstupním datovém prostoru FRB systémů.
2. Navrhněte a popište algoritmus tvorby cloudů.
3. Realizujte vlastní řešení v programovém prostředí MATLAB . Zohledněte možnost práce i v on-line režimu.
4. Ověřte tuto metodu na vybraných příkladech. Zohledněte možnost existence šumu v datech a vyhodnoťte jeho vliv.

DOPORUČENÁ LITERATURA:

1. Angelov P., Yager R.: A new type of simplified fuzzy rule-based system. International Journal of General Systems, Vol. 41, No. 2, February 2012, 163-185.
2. Klir G., Yuan B.: Fuzzy sets and fuzzy logic. Theory and Application, Prentice Hall 1995.

Termín zadání: 9.2.2015 ***Termín odevzdání:*** 18.5.2015

Vedoucí práce: prof. Ing. Pavel Jura, CSc.
Konzultanti diplomové práce:

doc. Ing. Václav Jirsík, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Cílem této práce je představení nového fuzzy systému typu AnYa, který, na rozdíl od klasických fuzzy systémů typu Takagi-Sugeno a Mamdani, používá typ antecedentu vycházejícím z reálně distribuce dat. V rámci práce bude zmíněný systém naprogramován a bude ověřena jeho funkčnost na dostupných testovacích datech.

Klíčová slova

fuzzy systémy, rekurzivní metoda nejmenších čtverců, datová hustota, mračna, antecedent

Abstract

The aim of this work is to introduce a new type of fuzzy system AnYa. This system, unlike the classical fuzzy systems Takagi-Sugeno and Mamdani, uses a type of antecedent based on real data distribution. As part of the work there will be mentioned system programmed and its functionality will be verified on testing data.

Keywords

fuzzy rule-based systems, recursive least square estimation, data density, clouds, antecedent

Bibliografická citace:

KLAPIL, O. *Fuzzy systémy s netradičními antecedenty fuzzy pravidel*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2015. 48s. Vedoucí diplomové práce byl prof. Ing. Pavel Jura, CSc.

Prohlášení

„Prohlašuji, že svou diplomovou práci na téma Fuzzy systémy s netradičními antecedenty fuzzy pravidel jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.
Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

V Brně dne: **18. května 2015** ……
podpis autora

OBSAH

ÚVOD

Fuzzy systémy se objevují již od sedmdesátých let minulého století, kdy byl popsán fuzzy systém typu Mamdani. Začátkem devadesátých let se těšily veliké oblibě pro aplikace inteligentních systémů. Ačkoliv se objevilo velké množství variant fuzzy systémů, téměř vždy dodržovaly klasickou strukturu fuzzifikace-inference-defuzzifikace. Tyto metody mají nedostatky právě v modulu fuzzifikace, kdy jsou data mapována na funkci příslušnosti, která nereprezentuje skutečné rozložení dat. Tento nedostatek se pokouší vyřešit nový fuzzy systém typu AnYa. Tato práce si klade za cíl ověřit funkčnost tohoto nového systému.

V první části budou představeny stávající fuzzy systémy a vysvětlena jejich funkce aby bylo možné je porovnat s novým systémem AnYa. Principy systému AnYa budou vysvětleny v druhé kapitole práce tak, aby bylo možné přistoupit k jeho realizaci v programovém prostředí MATLAB. Této realizaci se bude věnovat třetí kapitola.

V závěrečné části bude vytvořený systém otestován pro různé typy úloh a tím bude prověřena jeho funkce.

1. FUZZY LOGIKA [3,4]

Fuzzy logika vychází z matematické disciplíny fuzzy množin. Pojem „fuzzy množiny“ zavedl v roce 1965 profesor Lotfi A. Zadeh v článku *„Fuzzy sets“* pro časopis *Information and Control*. Slovo „fuzzy“ znamená „mlhavý“ či „nejasný“ a je zde proto, že se fuzzy množiny vyznačují právě takovými vlastnostmi. Díky jejich neostrosti se dá matematicky popsat např. lidské vnímání, které většinou neoperuje s čísly, ale s neurčitými výrazy typu „daleko – blízko“ atp.

1.1. Fuzzy množiny

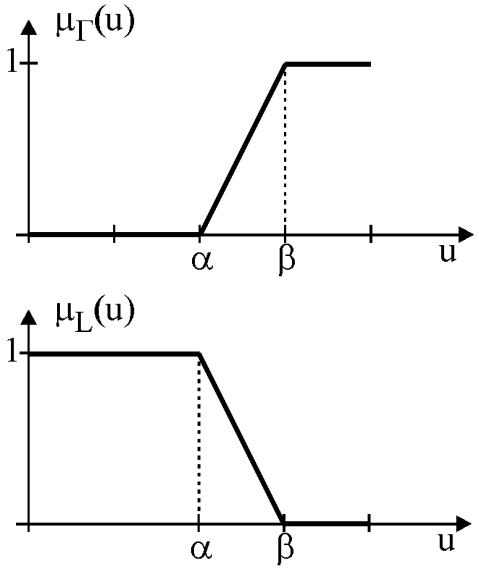
Na rozdíl od klasických množin, kdy libovolná klasická množina A definována v univerzu K a každý prvek $u \in K$, má pouze dva stavy – prvek u náleží množině A nebo prvek u nenáleží množině A. Ve fuzzy množinách může prvek univerza u náležet fuzzy množině F pouze částečně.

Matematicky lze zapsat jako:

$$Klasická\ množina\ (charakteristická\ funkce): \mu_A: K \rightarrow \{0,1\}, \mu_A(u) \in \{0,1\} \quad (2.1)$$

$$Fuzzy\ množina\ (funkce\ příslušnosti): \mu_F: K \rightarrow \langle 0,1\rangle, \mu_F(u) \in \langle 0,1\rangle \quad (2.2)$$

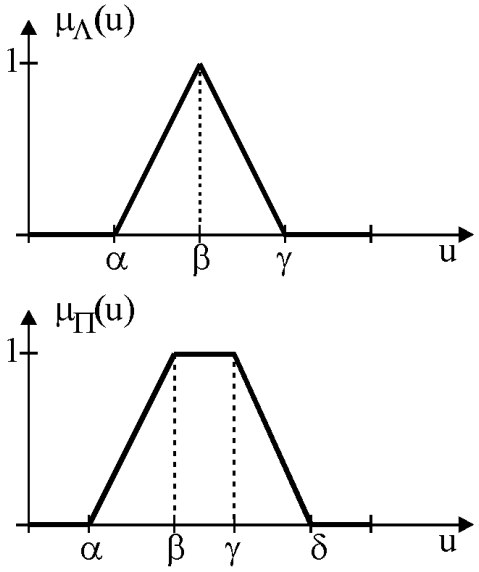
Funkce příslušnosti fuzzy množiny mapuje univerzum na celý interval $\langle 0,1\rangle$. Ve fuzzy množinách je pro klasickou množinu zaveden výraz „ostrá množina“. Tvar funkce příslušnosti je určen samotnou fuzzy množinou. Aby se zjednodušily výpočty, tak se obvykle volí snadno popsatelné tvary příslušnosti. Mezi nejčastěji používané po částech lineární funkce patří Γ-funkce , L-funkce, Λ-funkce a Π-funkce:



$$\Gamma(u,\alpha,\beta) = \begin{cases} 0 & u < \alpha \\ (u-\alpha)/(\beta-\alpha) & \alpha \le u \le \beta \\ 1 & u > \beta \end{cases}$$

$$L(u,\alpha,\beta) = \begin{cases} 1 & u < \alpha \\ (\beta-u)/(\beta-\alpha) & \alpha \le u \le \beta \\ 0 & u > \beta \end{cases}$$

Obr. 1: Vybrané funkce příslušnosti 1 [3]

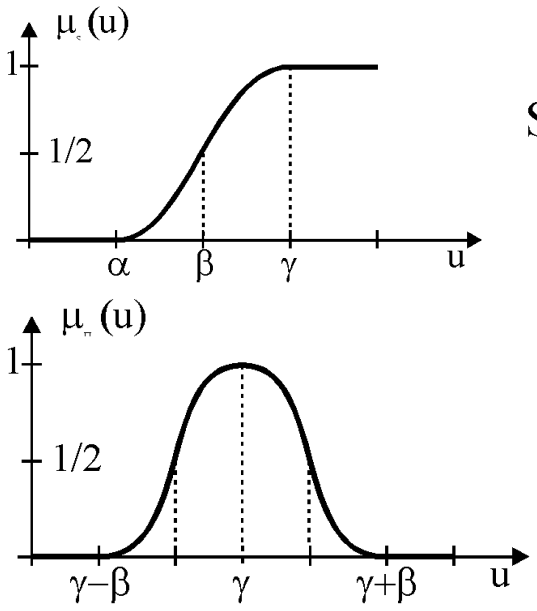


$$\Lambda(u,\alpha,\beta,\gamma)=\begin{cases}0 & u<\alpha\\ (u-\alpha)/(\beta-\alpha) & \alpha\le u\le\beta\\ (\gamma-u)/(\gamma-\beta) & \beta\le u\le\gamma\\ 0 & u>\gamma\end{cases}$$

$$\Pi(u,\alpha,\beta,\gamma,\delta)=\begin{cases}0 & u<\alpha\\ (u-\alpha)/(\beta-\alpha) & \alpha\le u\le\beta\\ 1 & \beta\le u\le\gamma\\ (\delta-u)/(\gamma-\delta) & \gamma\le u\le\delta\\ 0 & u>\delta\end{cases}$$

Obr. 2: Vyprané funkce příslušnosti 2 [3]

V určitých případech jsou používány i jiné funkce příslušnosti, jako například funkce příslušnosti definované Lotfi A. Zadehem: S-funkce a Zadehova *Π*-funkce:



$$S(u,\alpha,\gamma)=\begin{cases}0 & u\le\alpha\\ 2(u-\alpha)^2/(\gamma-\alpha)^2 & \alpha<u\le\beta\\ 1-2(u-\gamma)^2/(\gamma-\alpha)^2 & \beta<u\le\gamma\\ 1 & u>\gamma\end{cases}$$

$$\beta=(\alpha+\gamma)/2$$

$$\Pi(u,\beta,\gamma)=\begin{cases}S(u,\gamma-\beta,\gamma) & u\le\gamma\\ 1-S(u,\gamma,\gamma+\beta) & u\ge\gamma\end{cases}$$

Obr. 3: Vybrané funkce příslušnosti 3 [3]

Tyto funkce příslušnosti ovšem pouze zřídkakdy reprezentují skutečné rozložení dat.

Pomocí funkcí příslušnosti můžeme určit, zdali se dvě fuzzy množiny sobě rovnají, či jestli je nějaká fuzzy množina podmnožinou fuzzy množiny druhé.

Existují dvě fuzzy množiny A a B definované na univerzu X a jejich funkce příslušnosti μ_A a μ_B.

$$Rovnost: A = B \text{ právě tehdy, když } \mu_A(x) = \mu_B(x), x \epsilon X \qquad (2.3)$$

$$Podmnožina: A \subseteq B \text{ právě } tehdy, když\ \mu_A(x) \leq \mu_B(x), x \epsilon X \tag{2.4}$$

Abychom mohli s fuzzy množinami pracovat, byly zavedeny základní operace jako průnik (t-norma), sjednocení (s-norma) a doplněk (negace). Na rozdíl od klasických množin nejsou tyto operace definovány jednoznačně a můžeme se setkat s celou škálou jejich definicí. Pro fuzzy množiny A a B definovaná na univerzu U a pro všechna $u \in U$:

Některé t-normy:

$$průnik\ (Zadeh): \mu_{A\cap B}(u) = \min\left(\mu_A(u), \mu_B(u)\right) \tag{2.5}$$

$$průnik\ (dot\ product): \mu_{A\cap B}(u) = \mu_A(u) \cdot \mu_B(u) \tag{2.6}$$

Některé s-normy:

$$sjednocení\ (Zadeh): \mu_{A\cup B}(u) = \max\left(\mu_A(u), \mu_B(u)\right) \tag{2.7}$$

$$sjednocení: \mu_{A\cup B}(u) = \mu_A(u) + \mu_B(u) - \mu_A(u) \cdot \mu_B(u) \tag{2.8}$$

Negace:

$$doplněk\ (Zadeh): \mu_{A'}(u) = 1 - \mu_A(u) \tag{2.9}$$

Dále si definujeme operace nad fuzzy relacemi. Jedná se o operace projekce a cylindrické rozšíření. Máme-li definovanou fuzzy relaci A na kartézském součinu univerz $W \times U$, fuzzy množinu B definovanou na univerzu U a $w \in W$:

$$Projekce: proj\ A\ na\ U = \int_U \sup_{\forall w} (\mu_A(w,u))/u \tag{2.10}$$

$$Cylindrické\ rozšíření: ce(B) = \int_{W\times U} \mu_B(u)/(w,u) \tag{2.11}$$

Znaménko integrálu značí symbol výčtu na spojitém nebo nespočetném univerzu.

Poslední definovanou operací je operace kompozice. Jedná se o spojení cylindricky rozšířené fuzzy množiny, fuzzy relace a jejich projekce. Výsledkem je fuzzy množina. Použijeme-li značení z předchozího odstavce a přidáme fuzzy množinu V definovanou na univerzu U:

$$kompozice: V = B \circ A = proj(ce(B) \cup A)\ na\ U \tag{2.12}$$

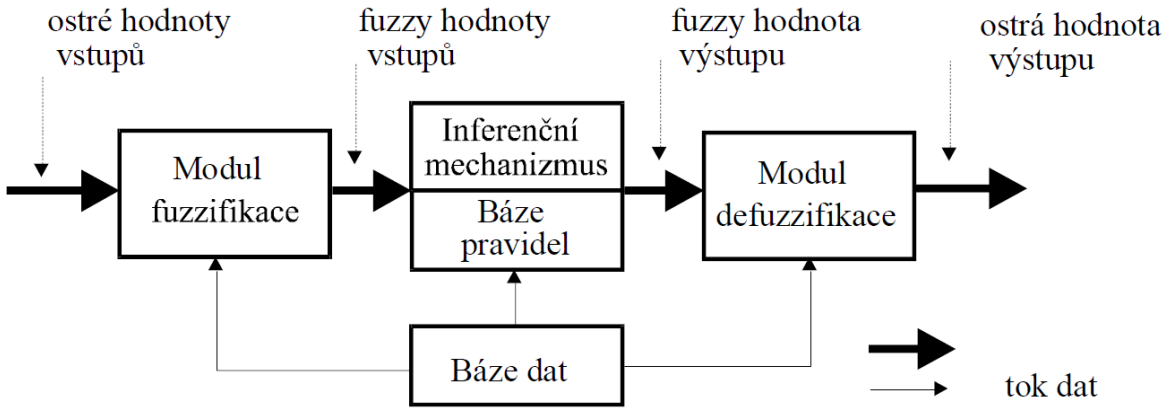
Opět lze použít pro průnik různou t-normu. Vznikají tak různé typy kompozice např. max-min kompozice nebo max-dot kompozice.

$$max - min: \mu_V(u) = max_{\forall w} \min(\mu_B(w), \mu_A(w,u)) \quad (2.13)$$

$$max - dot: \mu_V(u) = max_{\forall w}(\mu_B(w) \cdot \mu_A(w,u)) \quad (2.14)$$

1.2. Fuzzy systémy

Fuzzy systémy jsou systémy, v nichž se operuje i s neostrými hodnotami, reprezentovanými fuzzy množinami. Základní struktura fuzzy systému je obvykle složena ze tří částí: modul fuzzifikace inferencí mechanizmus + báze pravidel a modul defuzzifikace.

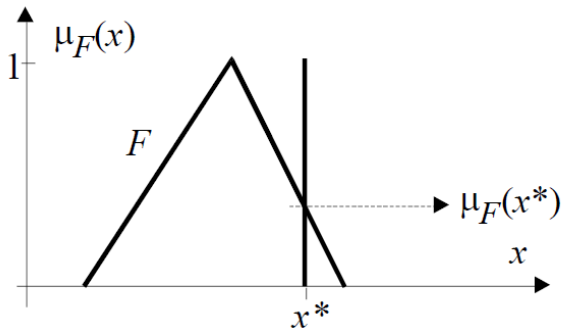


Obr. 4: Model fuzzy systému [3]

Modul fuzzifikace mapuje vstupní ostrá data na fuzzy množiny. Nejčastěji je používána fuzzifikace singletonem, kdy je vstupní ostré číslo x^* převedeno na fuzzy množinu s funkcí příslušnosti $\mu(x^*)$:

$$\mu(x) = \begin{cases} 1 & x = x^* \\ 0 & x \neq x^* \end{cases}$$

A poté mapováno na fuzzy množinu F:



Obr. 5: mapování singetonu na fuzzy funkci [3]

Báze pravidel sdružuje všechny pravidla pro daný systém a báze dat obsahuje informace o tvaru a polohách fuzzy množin v jejich univerzech. Inferenční mechanizmus a báze pravidel zastupují mechanismus tzv. přibližného usuzování. Při přibližném usuzování se rozhoduje na základě soboru pravidel, které představují fuzzy inferenci (úsudek). Tato pravidla představují jednotlivé implikace a vyjadřují kauzální vztah mezi fuzzy výroky. Obecně vypadají:

$$IF(fuzzy\ výrok)THEN(fuzzy\ výrok)$$

Fuzzy výrok za IF se nazývá předpoklad (antecedent), fuzzy výrok za $THEN$ se nazývá závěr (consequent). Jednotlivé fuzzy výroky mohou být spojením více fuzzy výroků logickými operátory *and* (pro fuzzy výroky se používá t-norma) a *or* (pro fuzzy výroky se používá s-norma). Pro fuzzy systémy byla vytvořena celá řada fuzzy implikací *if-then*, které jsou reprezentovány výslednými fuzzy relacemi R. Některé vychází z analogie klasické implikace dvouhodnotové logiky (implikace Zadeh, implikace Kleene-Dienes, implikace Lukasiewicz):

$$v_1 \rightarrow v_2 = (not\ v_1) or\ v_2 = (v_1\ and\ v_2) or (not\ v_1)$$

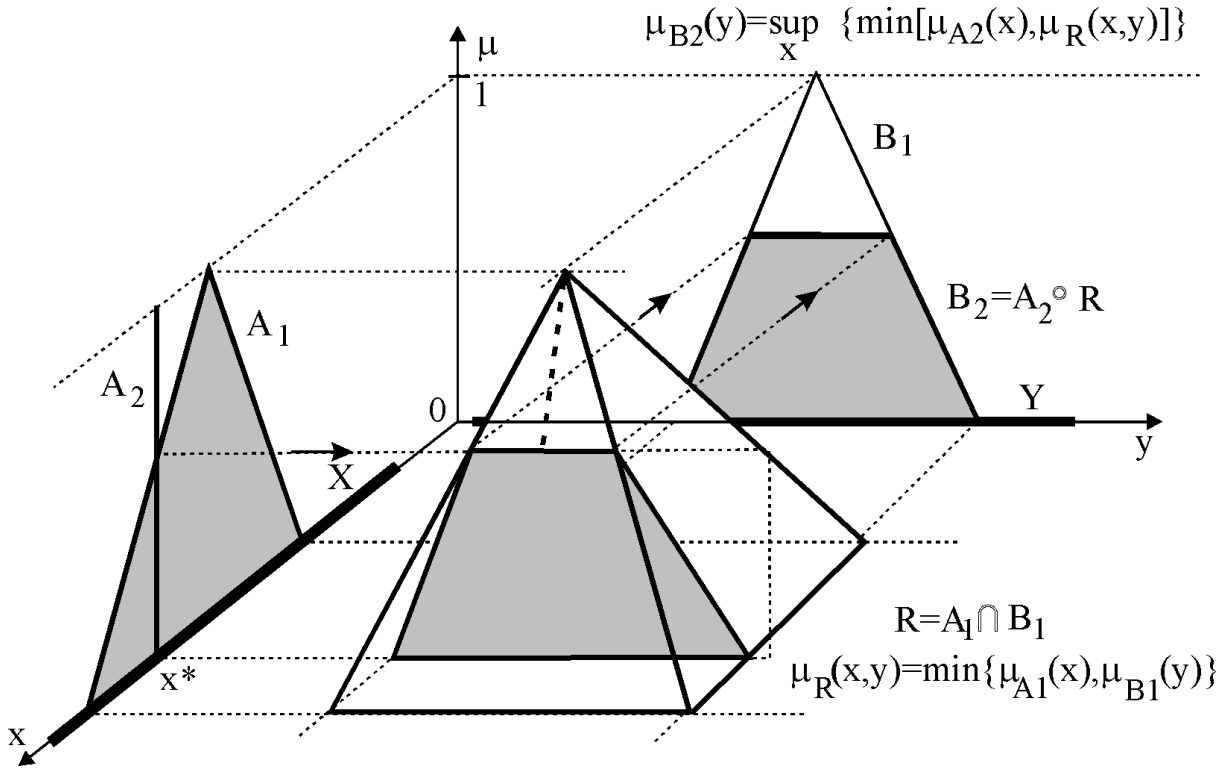
$$v_1, v_2 \ldots jednotlivé\ výroky$$

Některé implikace, zejména jedna z nejpoužívanějších implikací Mamdani a implikace Larsen, využívají:

$$v_1 \rightarrow v_2 = v_1\ and\ v_2$$

$$v_1, v_2 \ldots jednotlivé\ výroky$$

Liší se v použitých t-normách, s-normách a negaci.



Obr. 6: Implikace typu Mamdani [3]

Pro vyhodnocení implikace se využívá pravidlo *zobecněný modus ponens*. Máme-li definovány dvě proměnné $a \epsilon A, b \epsilon B$ a známe implikaci mezi nimi, tedy $IF(a\ je\ F_1)THEN(b\ je\ V_1)$, můžeme zkonstruovat fuzzy relaci R podle některé z výše popsaných implikaci. Platí $F \in A, V \in$ B a R ∈ A × B. Do fuzzy implikace vstupuje první proměnná a ve formě $(a\ je\ F_2)$. Do implikace tedy vstupuje hodnota F_2. Pravidlo modus ponens bude vypadat:

$$\begin{array}{rc} podmínka\ 1: & a\ je\ F_2 \\ podmínka\ 2: & IF(a\ je\ F_1)THEN(b\ je\ V_1) \\ vyhodnocení: & b\ je\ V_2 \end{array}$$

Výsledek je určen pomocí kompozice:

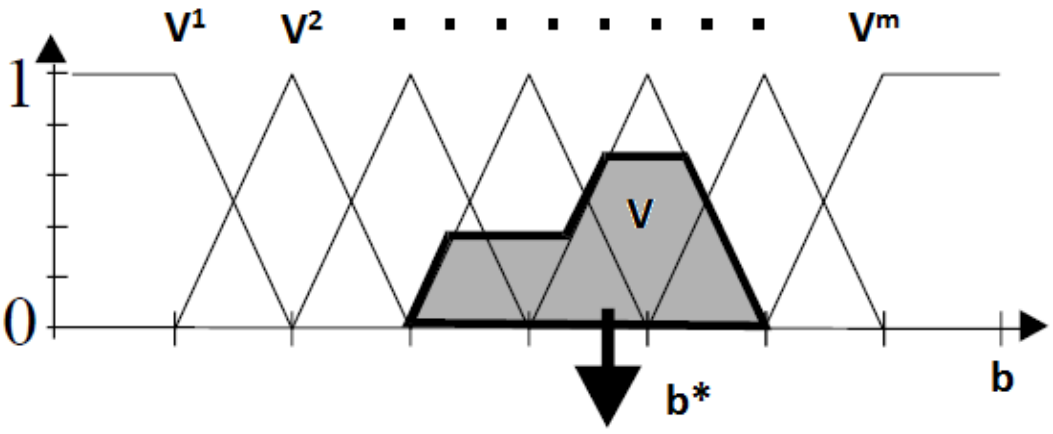
$$V_2 = F_2 \circ R = proj(ce(F_2) \cup R)\ na\ B \tag{2.15}$$

Můžeme využít například *max-min* kompozice, což je často využívané pro fuzzy systémy typu Mamdani.

Modul defuzzifikace převádí výstupní fuzzy hodnoty inferenčního mechanizmu na ostré výstupní hodnoty. Máme-li proměnnou $b \in B$ a univerzum B je pokryto systémem fuzzy množin $V^i, i = 1,2 \ldots m, V^i \in B$, a máme výsledek inferenčního mechanismu, kdy jsou jednotlivým fuzzy množinám přiřazeny výsledné hodnoty příslušnosti $\mu_{V^i}(b)$, tak můžeme použít některou z metod defuzzifikace. Je velké množství metod pro defuzzifikaci. Mezi nejpoužívanější patří metoda těžiště (Centre Of Gravity), mezi dalšími metodami jsou např. metoda středu součtů (Centre Of Sum) nebo metoda prvního maxima (First Of Maximum). Metoda těžiště určí ostrou hodnotu defuzzyfikace jako souřadnici těžiště fuzzy množin:

$$b^* = \frac{\int_B b \cdot \mu_V \cdot db}{\int_B \mu_V \cdot db} \qquad b^* = \frac{\sum_{i=1}^{m} b_i \cdot \mu_V(b_i)}{\sum_{i=1}^{m} \mu_V(b_i)} \tag{2.16}$$

První rovnice je pro spojité nebo nespočetné univerzum (znaménko integrálu v tomto případě značí skutečný integrál) a druhá rovnice je pro diskrétní univerzum (i singletony).



Obr. 7: Defuzzyfikace metodou těžiště [3]

Fuzzy systémy lze použít pro velkou škálu aplikací jako například aproximace funkcí, řízení a regulaci, predikci či klasifikaci. Nejpoužívanější fuzzy systémy jsou typu Takagi-Sugeno a Mamdani.

1.2.1. Takagi-Sugeno

Fuzzy systém Takagi-Sugeno má ve vyhodnocení pravidla jako závěr funkci vstupních proměnných. Nechť jsou $a_i, \dots, a_n$ vstupní proměnné náležící do univerz A_i, výstupní proměnná $b \in B$ a univerza A_i jsou pokryta fuzzy množinami $F_i^{m_i}$. Soubor k −pravidel fuzzy systému Takagi-Sugeno potom vypadá následovně:

$$IF\left(a_1\ je\ F_1^{m_1 j}\right) AND \dots AND\left(a_i\ je\ F_i^{m_i j}\right) THEN\left(b = f_j(a1, \dots, a_n)\right), j = 1,2, \dots kmax - dot{:}\ \mu_V(u) = max_{\forall w}\left(\mu_B(w) \cdot \mu_A(w,u)\right) \tag{2.17}$$

Jako defuzzifikace se používá fuzzy vážený průměr výstupů funkcí f_j:

$$b = \frac{\sum_{j=1}^{k} w_j f_j(a1, \dots, a_n)}{\sum_{j=1}^{k} w_j} \tag{2.18}$$

1.2.2. Mamdani

Fuzzy systém Mamdani se od systému Takagi-Sugeno liší pouze ve vyhodnocení pravidel. Po vyhodnocení je jako závěr fuzzy množina. Nechť je $a_i, \dots, a_{n_i}$ vstupní proměnné náležící do univerz A_i, výstupní proměnná $b \in B$, univerza A_i jsou pokryta fuzzy množinami F_i^m a univerzum B je pokryto fuzzy množinami V^n. Soubor k −pravidel fuzzy systému Mamdani vypadá následovně:

$$IF\left(a_1\ je\ F_1^{m_1 j}\right) AND \dots AND\left(a_i\ je\ F_i^{m_i j}\right) THEN\left(b = V^{nj}\right), j = 1,2, \dots k \tag{2.19}$$

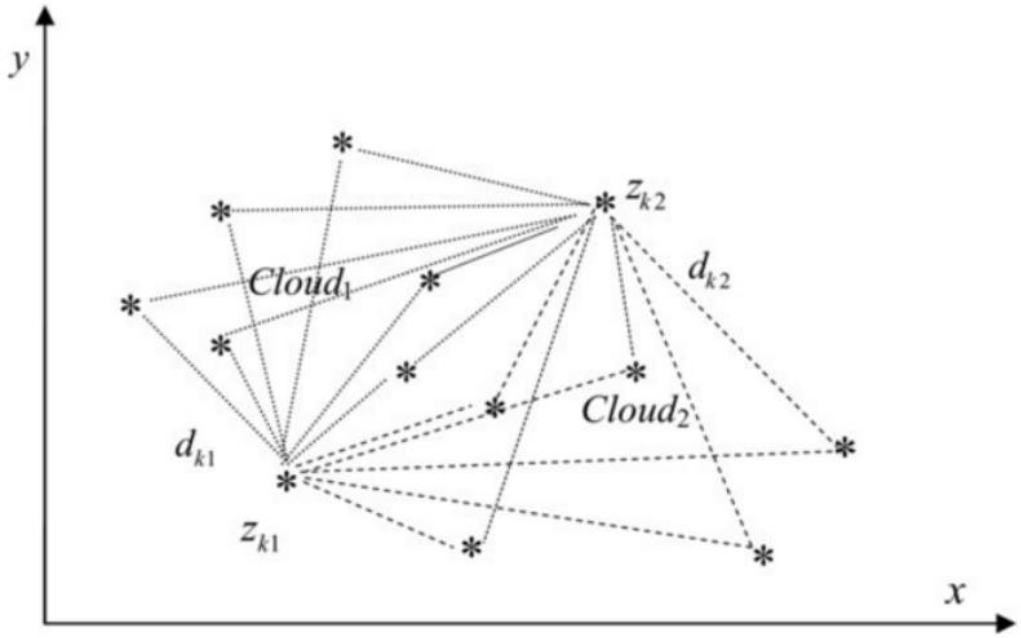
K defuzzyfikaci závěru se využívá některá z metod defuzzyfikace (velmi často metoda těžiště).

1.2.3. AnYa

Nově navrhovaný systém typu AnYa (Angelov-Yager) se výrazně liší od nejpoužívanějších systémů typu Mamdani nebo Takagi-Sugeno. Tyto systémy, které jsou popsány výše, používají pro svou činnost nějakou formu fuzzyfikace, která vyžaduje apriorní znalost vstupních dat. Navíc při fuzzifikaci téměř vždy vznikne ztráta dat, jelikož mapování vstupu na fuzzy množinu zřídkakdy reprezentuje skutečné rozložení dat. Vytvořený předpoklad (antecedent) je tedy zkreslený. Nový systém se tento nedostatek fuzzy systémů snaží odstranit tím, že data sdružuje do neparametrických datových Mračen (clouds) reprezentujících vstupně-výstupní prostor fuzzy systému. Výhodou je možnost práce s daty, o kterých nemáme žádné informace, kromě počtu vstupů a výstupů. Datová Mračna sdružují data se vzájemnou podobností ve vstupně-výstupním prostoru. Algoritmus bude popsán v kapitole 3.

2. FUZZY SYSTÉM ANYA [1][2]

Mějme komplexní, nelineární, nedeterministický systém, který můžeme popsat pouze vztahy vstup-výstup, tedy vektor vstupů $\mathbf{x}= [x_1, x_2, \dots, x_n]$ a vektory výstupů $\boldsymbol{y}^i = [y_1^i, y_2^i, \dots, y_m^i]$. Tato metoda funguje na principu popsání vztahů vstup-výstup na základě historie předešlých vstupně-výstupních dat, $\boldsymbol{z}_j = [\boldsymbol{x}_j^T; \boldsymbol{y}_j^T]^T, j = 1,2, \dots, k-1$ a momentálního k-tého vstupu $\boldsymbol{x}_k{}^T$.



Obrázek 1.: Reprezentace Mračna [1]

Metoda používá datová Mračna (Clouds), která jsou podobná datovým shlukům (cluster), ale liší se tím, že nepoužívají průměr nebo rozptyl, proto nemají určený tvar či hranice. Mračna jsou soubory dat v prostoru, mající svoji relativní hustotu. Hlavní výhodou je, že Mračna jsou neparametrické a reprezentují skutečné data. To je rozdíl oproti klasickým přístupům, kdy je třeba vstupy parametrizovat pomocí funkce příslušnosti, čímž dochází ke ztrátě dat, jelikož nahradíme skutečná data jejich odhady (viz. Kapitola 2). Mračna přímo reprezentují všechny předchozí záznamy dat, které jsou si navzájem blízké ve smyslu vztahu vstup-výstup a tvoří základní struktury systému AnYa. Nový vstup $\boldsymbol{x}_k$ či dvojice vstup-výstup $\boldsymbol{z}_k$ patří každému Mračnu s rozdílnou příslušností $\gamma \in [0; 1]$. To lze popsat:

$$(\boldsymbol{z}\ je\ jako\ \zeta^i)$$

Kde $\zeta^i \in R^{n+m}, i = [1, N]$, N je počet Mračen ve vstupně-výstupním prostoru, *n* je dimenze vstupů a *m* dimenze výstupů. Analogicky lze zapsat:

$$(\boldsymbol{x}\ je\ jako\ \chi^i)$$

Kde $\chi^i \in R^n, i = [1, N]$, N je počet Mračen ve vstupním prostoru a n je dimenze vstupů.

Rozlišujeme dvě základní hodnoty pro nový vzorek dat. První je lokální hustota i-tého Mračna pro tento nový vstup $\boldsymbol{x}_k$, která je definována vhodným jádrovým odhadem hustoty vzdálenosti mezi $\boldsymbol{x}_k$ a všemi vzorky i-tého Mračna:

$$\gamma_k^i = \boldsymbol{K}\left(\sum_{j=1}^{M^i} d_{kj}^i\right), \qquad i = [1, N] \; a \; M^i \; je \; počet \; vzorků \; i-tého \; Mračna \tag{3.1}$$

Dále rozlišujeme globální hustotu pro nový vzorek vstupně-výstupních dat $\boldsymbol{z}_k$, která je definována jako vhodný jádrový odhad hustoty vzdálenosti $\boldsymbol{z}_k$ a všech předchozích vstupně-výstupních vzorků:

$$\Gamma_k = \boldsymbol{K}\left(\sum_{j=1}^{k} d_{kj}\right) \tag{3.2}$$

Pro výpočet vzdálenosti je možné použít rozličné typy výpočtů – Eukludovská vzdálenost, kosinová vzdálenost atp.

Pro rekurzivní výpočet těchto základních hodnot je vhodné použít jako jádrový odhad hustoty Cauchyho jádrový odhad hustoty (je monotónní, maximální hodnota je 1 a asymptoticky se blíží nule, když argument se blíží nekonečnu):

$$\gamma_k^i = \frac{1}{1 + \left(\frac{1}{M^i}\right)\sum_{j=1}^{M^i}\left(d_{kj}^i\right)^2} = \frac{1}{1 + \left(\overline{d^2}\right)_k^i}, \qquad i = [1, N] \tag{3.3}$$

$\overline{d} = průměrná \; vzdálenost \; k-tého \; vstupu \; vůči \; všem \; bodům \; i-tého \; Mračna$

Toto lze převést na rekurzivní variantu výpočtu:

$$\gamma_k^i = \frac{1}{1 + \|x_k - \mu_k^L\|^2 + \Sigma_k^L - \|\mu_k^L\|^2} \tag{3.4}$$

$$\mu_k^L = \frac{M_k^i - 1}{M_k^i}\mu_{k-1}^L + \frac{1}{M_k^i}x_k, \qquad \mu_1^L = x_1 \tag{3.5}$$

$$\mu_k^L = lokální \; střední \; hodnota \; dat \; i-tého \; Mračna$$

$$\Sigma_k^L = \frac{M_k^i - 1}{M_k^i}\Sigma_{k-1}^L + \frac{1}{M_k^i}\|x_k\|^2, \qquad \Sigma_1^L = \|x_1\|^2 \tag{3.6}$$

$$\Sigma_k^L = lokální \; rozptyl \; dat \; i-tého \; Mračna$$

Podobně lze určit globální hustota:

$$\Gamma_k = \frac{1}{1 + \left(\frac{1}{k-1}\right)\sum_{j=1}^{k}\left(d_{kj}^{i}\right)^2} = \frac{1}{1 + \left(\overline{d^2}\right)_k} \tag{3.7}$$

Což lze opět zapsat rekurzivně:

$$\Gamma_k = \frac{1}{1 + \|z_k - \mu_k\|^2 + \Sigma_k - \|\mu_k\|^2} \tag{3.8}$$

$$\mu_k = \frac{k-1}{k}\mu_{k-1} + \frac{1}{k}x_k, \qquad \mu_1 = z_1 \tag{3.9}$$
$$\mu_k = globální\ střední\ hodnota\ dat\ pro\ všechna\ data\ (vstup\ i\ výstup)$$

$$\Sigma_k = \frac{k-1}{k}\Sigma_{k-1} + \frac{1}{k}\|z_k\|^2, \qquad \Sigma_1 = \|z_1\|^2 \tag{3.10}$$
$$\Sigma_k = globální\ rozptyl\ dat\ pro\ všechna\ data\ (vstup\ i\ výstup)$$

Stupeň příslušnosti vzorku vstupních dat $\boldsymbol{x}_k$ k Mračnu **i** je určen normalizovanou lokální hustotou vypočítanou:

$$\lambda_k^i = \frac{\gamma_k^i}{\sum_{j=1}^{N}\gamma_k^j}, \qquad i = [1, N] \tag{3.11}$$

Fuzzy pravidla tohoto systému tedy můžeme zapsat:

$$Pravidlo^i\colon\ IF\left(\boldsymbol{x}\ je\ jako\ \chi^i\right)\ THEN\left(y^i = x_e^T\pi^i\right)$$

Kde $x_e^T = [1, x^T]$ je rozšířený vektor vstupů a π je matice výstupních parametrů (consequent) i-tého Mračna

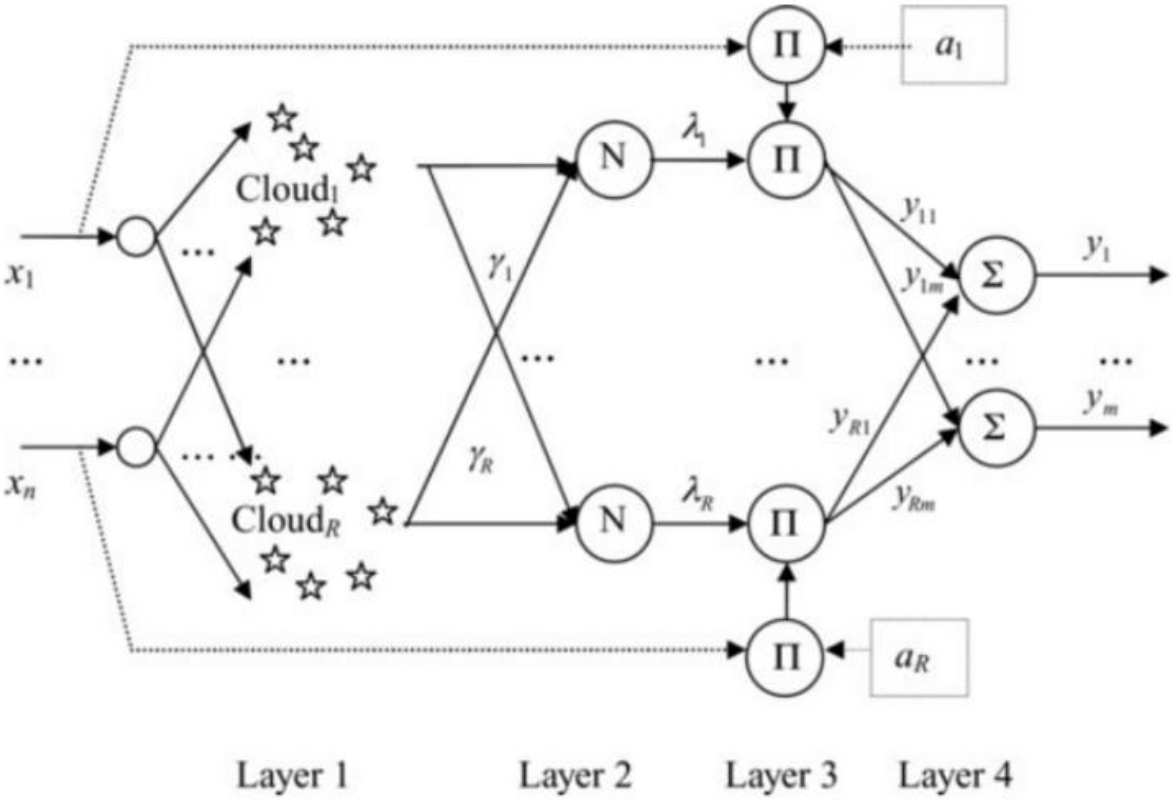
$$\pi^i = \begin{bmatrix} a_{01}^i & a_{02}^i & \dots & a_{0m}^i \\ a_{11}^i & a_{12}^i & \cdots & a_{1m}^i \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}^i & a_{n2}^i & \cdots & a_{nm}^i \end{bmatrix}$$

Výstup (defuzzifikaci) systému můžeme napsat:

$$y = \sum_{i=1}^{N}\lambda^i y^i \tag{3.12}$$

Jedná se o fuzzy vážený součet výstupů všech mračen, kdy normalizovaná lokální hustota λ^i je ekvivalentem k hodnotě funkce příslušnosti $\mu^i(\boldsymbol{x})$ u fuzzy systému Mamdani a váze pravidla w^i u fuzzy systému Takagi-Sugeno.

Systém lze graficky popsat jako čtyřvrstvou neuronovou síť:



Obrázek 2: Fuzzy systém AnYa jako neuronová síť [2]

V první vrstvě se vstupy rekurzivně porovnají se všemi předchozími záznamy jednotlivých Mračen. V druhé vrstvě se spočítá stupeň příslušnosti vstupu k jednotlivým Mračnům. Tyto první dvě vrstvy reprezentují výchozí předpoklad (antecedent). Třetí vrstva sloučí předpoklady se závěry (consequent) které reprezentují jednotlivé sub-systémy (Mračna). Poslední vrstva vypočítá výsledek systému jako vážený součet výstupů jednotlivých sub-systémů.

2.1. Systém

Systém je možno programovat jako dynamicky se vyvíjející. Buď začínáme existující strukturou Mračen, nebo začínáme od začátku. Existující mračna můžeme vytvořit na základě známých dat, či s přispěním experta. Začínáme-li od začátku, začneme tím, že vytvoříme první mračno z prvního vzorku vstupně-výstupních dat. Další Mračna tvoříme na základě pravidel, aby systém splňoval:

- Dobrou generalizaci vstupně-výstupního prostoru.
  Tohoto pravidla je dosaženo tím, že nové Mračno je založeno na základě vzorku dat, který má vysokou globální hustotu.
- Vyvarování se vysoké míry překrývání Mračen.
- Udržování kvality Mračen a odstraňování neužitečných či zastaralých Mračen.

Při přijetí k-tého vzorku vstupů rozhodneme, zda je třeba vytvořit nové Mračno podle podmínek:

$$\Gamma_r > \Gamma_k^i, \forall i | i = [1, N] \tag{3.13}$$

Rovnici (3.13) lze popsat jako: Globální hustota po přidání nového vstupně-výstupního vzorku je vyšší, než globální hustoty spočítané s odhady jednotlivých mračen ($z_k{}^i = [x_k, y^i]$).

$$\exists i | i = [i, N], \qquad |\gamma_k^i| > \Lambda \tag{3.14}$$

Existuje-li alespoň jedno Mračno s lokální hustotou menší, než Λ, tak nové Mračno nevytvoříme ani v případě platnosti (3.13). Rozhodujeme o míře překrytí vzorku dat s existujícími Mračny na základě statistiky a předpokládáme normální rozložení dat. Standardní hodnota parametru $\Lambda = e^{-1}$, což reprezentuje tzv. 1σ pravidlo. Tuto hodnotu můžeme v případě potřeby změnit. Čím větší hodnota, tím ochotněji se budou tvořit nové Mračna a naopak.

Hodnota parametru Λ [-]	**Míra překrytí [%]**
0.5	50
e^{-1}	31.73
1/10	10
1 / 21.98	4.55

Tabulka 1: Závislost míry překrytí na parametru Λ

Nejsou-li splněny podmínky pro vytvoření nového Mračna, tak se k-tý vzorek vstupů přiřadí Mračnu, kterému je daný vzorek nejbližší:

$$M^i = M^i + 1 \; pro \; i = \arg \max_{i=1 \; až \; N} (\gamma^i), \qquad i = [1, N] \tag{3.15}$$

a upravíme lokální hustotu tohoto Mračna podle (4).

Dále pro potřeby monitorování kvality Mračna je potřeba zavést pojmy stáří Mračna a užitečnost Mračna. Monitorování kvality mračna je potřebné zvláště pro „on-line“ režim systému, kdy např. sbíráme data přímo ze senzorů. Stáří Mračna vyjadřuje střední hodnotu času, kdy byly Mračnu přiřazeny data:

$$age^i = k - \overline{I_j}, \qquad i = [1, N] \tag{3.16}$$

$$\overline{I_j} = \frac{1}{M_k^i} \sum_{j=1}^{M_k^i} I_j \tag{3.17}$$

$$I_j = \check{c}asov\acute{y} \; index, kdy \; byla \; zapsána \; data \; do \; i - t\acute{e}ho \; mračna$$

Užitečnost Mračna se vypočítá jako průměr stupně příslušnosti vzorků i-tému Mračnu za délku života Mračna:

$$U_k^i = \overline{\lambda^i} = \frac{1}{k - I_i} \sum_{j=1}^{k} \lambda_j^i, \qquad i = [1, N]$$
$$I_j = \check{c}asov\acute{y}\ index\ založení\ i - tého\ Mračna \tag{3.18}$$

Užitečnost může být použita k odstraňování Mračen, když jejich užitečnost klesne pod hranici ε_1 (pro menší počet předpokládaných Mračen, cca. 50, je doporučeno volit hodnotu ε_1 kolem 0.1):

$$IF\left(U_k^i < \varepsilon_1\right) THEN\left(\lambda^i \leftarrow 0\right), \qquad i = [1, N] \tag{3.19}$$

2.2. Tvorba závěrů (consequents)

Tvorba závěrů pro jednotlivé subsystémy je omezena na pouhé jedno pravidlo pro subsystém:

$$Pravidlo^i\colon\ IF\left(\boldsymbol{x}\ je\ jako\ \chi^i\right) THEN\left(x \rightarrow t\check{r}ída^i\right), \qquad i = [1, N]$$

Pro samotnou klasifikaci lze použít jednoduché pravidlo „vítěz bere vše“:

$$T\check{r}ída = arg \max_{i=1\ až\ N} (\lambda_i) \tag{3.20}$$

Pro nalezení parametrů závěrů lze s úspěchem použít rekurzivní váženou metodu nejmenších čtverců. Celkový výstup ze systému lze popsat jako:

$$y = \psi^T \theta$$
$$\theta = [(\pi^1)^T, (\pi^2)^T, \dots, (\pi^N)^T]^T\ vektor\ parametrů\ subsystémů \tag{3.21}$$
$$\psi = [\lambda^1 x_e^T, \lambda^2 x_e^T \dots, \lambda^N x_e^T]^T\ vektor\ vážených\ vstupů\ stupněm\ příslušnosti\ pro\ lineární\ závěr$$
$$\psi = [\lambda^1, \lambda^2 \dots, \lambda^N]^T\ vektor\ vážených\ vstupů\ stupněm\ příslušnosti\ pro\ unikátní\ závěry$$

Hledáme parametry takové, aby se minimalizovala chyba výstupu ze systému od požadovaného výstupu vzorku vstupů x_k:

$$(Y - \psi^T \theta)^T (Y - \psi^T \theta) \rightarrow min \tag{3.22}$$

To lze provést rekurzivně:

$$\hat{\theta}_k = \hat{\theta}_{k-1} + C_k \psi_k \left(y_k - \psi_k^T \hat{\theta}_{k-1}\right), \qquad \hat{\theta}_1 = 0 \tag{3.23}$$

$$C_k = C_{k-1} - \frac{C_{k-1} \psi_k \psi_k^T C_{k-1}}{1 + \psi_k^T C_{k-1} \psi_k}, \qquad C_1 = \Omega \boldsymbol{I} \tag{3.24}$$

kde Ω je velké kladné číslo a $\boldsymbol{I}$ je jednotková matice o dimenzi $Nn \times Nn$.

3. REALIZACE SYSTÉMU ANYA V PROSTŘEDÍ MATLAB

Nově navrhovaný fuzzy systém typu AnYa byl v rámci této práce realizován v programu MATLAB 2014b. V této kapitole bude popsána jeho datová struktura a základní filosofie programu. Dále budou popsány jednotlivé varianty algoritmu pro různé úlohy. Mezi testované úlohy patří aproximace/predikce funkce a klasifikace.

3.1. Základní struktura programu

Jak bylo výše popsáno, fuzzy systém typu AnYa se chová jako MIMO systém. Na vstupu i výstupu jsou ostrá data. Struktura programu pro jednotlivé úlohy je v obecné rovině stejná a vypadá následovně:

1. ***Inicializace programu – vytvoření datových struktur a vytvoření prvního Mračna***
2. ***Načtení dat***
3. ***Tvorba nového Mračna nebo přiřazení dat existujícímu Mračnu***
4. ***Zhodnocení kvality Mračen***
5. ***Tvorba a úprava závěrů***
6. ***Výpočet výstupu***
7. ***Opakujeme body 2-6 dokud jsou dostupná data***

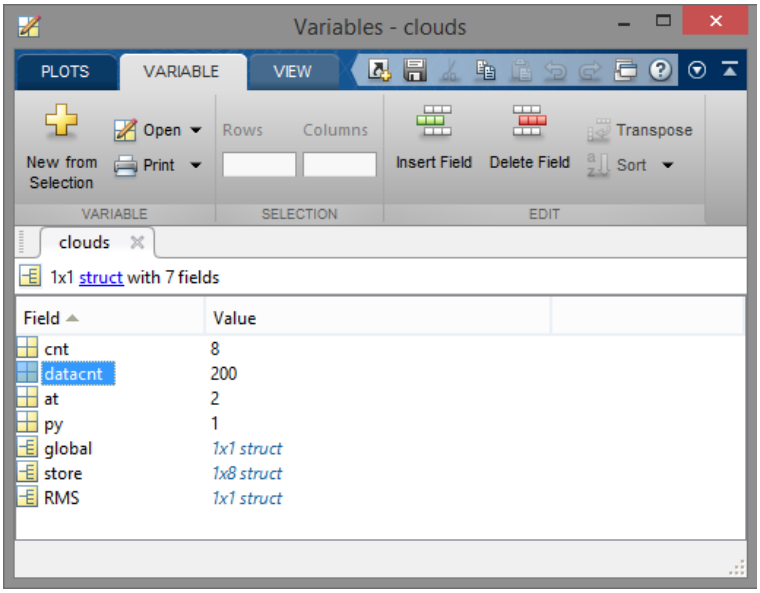
Toto platí pro online režim i pro režim učení (při režimu učení můžeme vynechat bod 6). Při vyhodnocování výstupu již existující struktury, kdy nepožadujeme tvorbu nových Mračen, ale pouze požadujeme výstup na základě vstupních dat (např. po režimu učení předkládáme testovací data), je struktura algoritmu omezena na:

1. ***Načtení dat***
2. ***Přiřazení dat existujícímu mračnu***
3. ***Výpočet výstupu***
4. ***Opakujeme body1-3 dokud jsou dostupná data***

3.2. Datová struktura

Strukturovaná proměnná „clouds“

Hlavní proměnná algoritmu, která reprezentuje blok báze dat fuzzy systému je nazvána „clouds“. Jedná se o strukturovanou proměnnou, která sdružuje nejdůležitější data potřebná pro funkci algoritmu.



Obr. 8: Datová struktura „clouds“

- Proměnná „clouds.cnt“ reprezentuje počet vytvořených Mračen.
- Proměnná „clouds.datacnt“ je počet dat, použitých k vytvoření stávající datové struktury „clouds“.
- „clouds.at“ a „clouds.py“ reprezentují informace o vstupních datech. Proměnná „at“ je počet vstupních proměnných a proměnná „py“ je počet výstupních proměnných jednotlivých dávek vstupních dat.

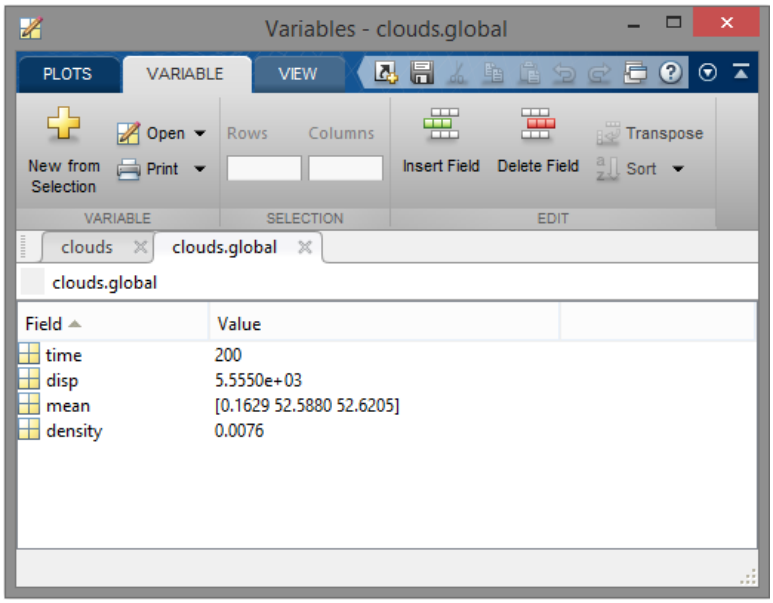
Dále se zde vyskytují tři strukturované proměnné. Jedná se o:

- „clouds.global“
- „clouds.store“
- „clouds.RMS“

Tyto struktury si dále popíšeme detailněji.

Proměnná „clouds.global“

Tato strukturovaná proměnná sdružuje informace o globální hustotě dat stávající datové struktury Mračen.

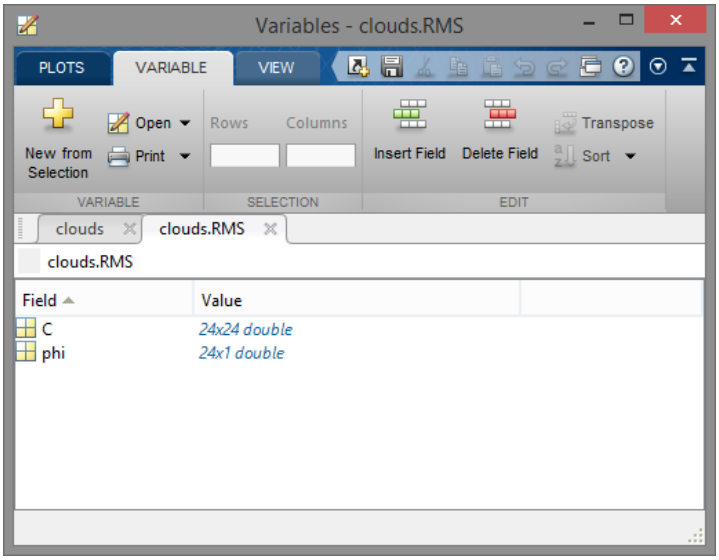


Obr. 9: Datová struktura "clouds.global"

- „clouds.global.time“ reprezentuje časový index, pro který platí vypočítaná globální hustota „clouds.global.density“
- „clouds.global.disp“ je globální rozptyl dat pro všechna data stávající datové struktury
- „clouds.global.mean“ vyjadřuje globální střední hodnotu všech dat stávající datové struktury
- „clouds.global.density“ je globální hustota všech dat stávající datové struktury pro daný časový index.

Proměnná „clouds.RMS“

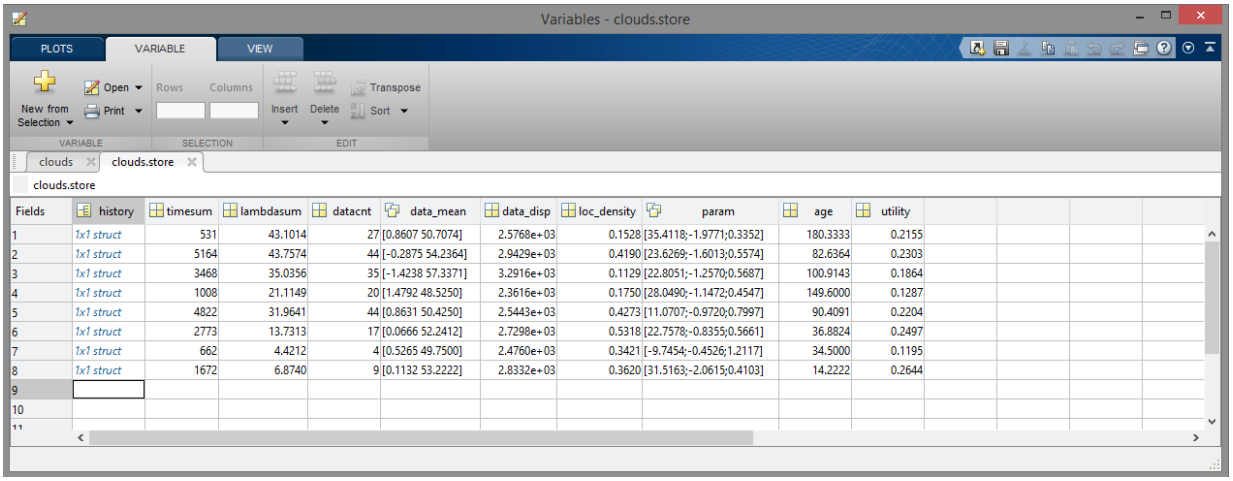
Strukturovaná proměnná „clouds.RMS“ sdružuje informace o závěrech (konsekventech) fuzzy systému AnYa pro stávající Mračna. Obsahuje matici C a vektor jednotlivých parametrů Mračen θ^T.



Obr. 10: srtukturovaná proměnná "clouds.RMS"

Proměnná „clouds.store“

Strukturovaná proměnná „clouds.store“ sdružuje informace o všech vytvořených Mračnech stávající datové struktury. Lze přistupovat i k jednotlivým Mračnům pomocí „clouds.store(i)“, kde i značí číslo Mračna, ke kterému chceme přistoupit.



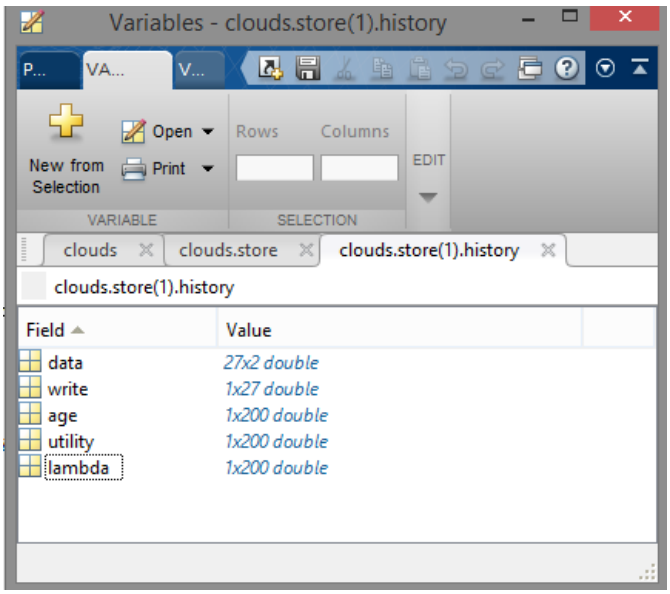
Obr. 11: Uložená Mračna v datové struktuře "clouds.store(i)"

Jednotlivé řádky reprezentují jednotlivá Mračna.

Mračna jsou popsána jejich určujícími vlastnostmi, tedy lokální hustotou γ^i, stářím Mračna age^i a užitečností Mračna U^i.

Stáří Mračen reprezentuje proměnná „clouds.store.age“ a užitečnost „clouds.store.utility“. Pro jejich výpočet jsou zde i pomocné proměnné „clouds.store.timesum“, které reprezentuje součet časových indexů zápisu do Mračen M^i a „clouds.store.lambdasum“, která reprezentuje součet normalizovaných lokálních hustot Mračen λ^i za celý jejich život.

Pro výpočet lokální hustoty Mračen podle rekurzivního algoritmu jsou používány proměnné „coluds.store.data_mean“ a „clouds.store.data_disp“, které jsou hodnoty lokální střední hodnoty dat v i-tém Mračnu a lokální hustota dat i-tého mračna.
„clouds.store.datacnt“ reprezentuje počet dat i-tého Mračna.
Velmi důležitou proměnnou každého Mračna je „clouds.store.param“. Tato proměnná reprezentuje vektor parametrů π^i použitých pro tvorbu závěru Mračna. Díky skutečnosti, že báze pravidel je omezena na jedno pravidlo na Mračno, můžeme snadno zjistit výstupní funkci jednotlivých Mračen.
Poslední strukturovanou proměnnou je „clouds.store.history“. Ta uchovává zapsané data a vlastnosti Mračna pro pozdější tvorbu grafů či kontroly funkčnosti algoritmu.



Obr. 12: datová struktura "clouds.store(i).history"

Jsou zde přítomny záznamy všech dat Mračna „clouds.store(i).history.data“, záznamy o časových indexech zápisu či čtení Mračna „clouds.store(i).history.write“, vývoj stáří „clouds.store(i).history.age“ a užitečnosti Mračna „clouds.store(i).history.utility“ a jejich normalizované lokální hustoty po celý život Mračna „clouds.store(i).history.lambda“.
Díky takto zvolené datové struktuře je pro jeho chod zapotřebí velmi malé množství pomocných proměnných a je zajištěna přehledná kontrola jeho funkčnosti.

3.3. Popis kódu programu pro jednotlivé varianty algoritmu AnYa

V této podkapitole bude vysvětlena funkce algoritmu na jeho kódu. Budou představeny varianty algoritmu pro jednotlivé úlohy. Výsledky budou shrnuty v další kapitole.

3.3.1. Úloha klasifikace

Celý algoritmus bude popsán na verzi pro klasifikaci. Až v další části bude věnováno místo na popsání rozdílů algoritmu ve verzi pro aproximaci.

Pro úlohu klasifikace je určující vlastností typ výstupu. Vstupy mohou být jakéhokoliv numerického typu, ale výstup z klasifikace bude obecně vektor hodnot určující zařazení do klasifikační skupiny. Tento výstup může být typu *singleton* či *enum*. Na základě toho je potřeba upravit tvorbu závěrů fuzzy systému a zvolit vhodnou metodu defuzzifikace. Jako nejefektivnější se ukázala metoda „vítěz bere vše“ a výstupní hodnoty typu *enum*.

Hlavní část algoritmu

Tato část je hlavní částí algoritmu. Řídí spouštění jednotlivých funkcí a vyhodnocuje jejich výsledky a tím realizuje strukturu z kapitoly 4.1. Je plně připravena na offline i online mód algoritmu. V případě offline modu jsou data načítána ve for cyklu. V případě online režimu se algoritmus umístí do funkce s daty jako vstup, výstupem bude výsledek defuzzifikace algoritmu. Pro správnou funkci algoritmu je potřeba inicializovat datovou strukturu:

```
global clouds;
clouds.cnt=0;
clouds.datacnt=0;
clouds.at=at;
clouds.py=py;
clouds.global=[];

omega=10^9;
```

Strukturovanou proměnnou „clouds“ je důležité označit jako globální proměnnou pro případ, že chceme algoritmus používat v online režimu. Lze to obejít tím, že vstupem a výstupem funkce s hlavní částí algoritmu bude proměnná „clouds“, ale řešení s její globální variantou je jednodušší implementovat např. do modulu programu MATLAB SIMULINK. Proměnné „at“ a „py“ jsou závislé na typu vstupních dat.

Samotný kód hlavní části algoritmu vypadá:

```
%tvorba prvniho Mracna
if (clouds.cnt==0)
    clouds=cloudgen_klas(clouds,ler(i,:),i);
    clouds.global=globalfcn(clouds.global,ler(i,:),i);
    gamma=compgam(clouds,ler(i,1:at));
else
    % shromazdovani informaci pro rozhodnuti
    glob_k_dens = globalfcntmp_klas( clouds.global,ler(i,:),clouds );
    clouds.global=globalfcn(clouds.global,ler(i,:),i);

    q_gen=true;
    gamma=compgam(clouds,ler(i,1:at));

    % rozhodovani o tvorbe noveho Mracna
    for j=1:clouds.cnt
        if (clouds.global.density<glob_k_dens(j))
            q_gen=false;
```

```
            end
            if (abs(gamma(j))>(exp(-1)))
                q_gen=false;
            end
        end

        for j=1:clouds.cnt
            lambda(j)=gamma(j)/sum(gamma);
        end

        % tvorba noveho Mracna ci prirazeni dat k existujicimu
        if(q_gen==true)
            clouds=cloudgen_klas(clouds,ler(i,:),i);
        else
            cloud_win=find(lambda==max(lambda));
            cloud_win=cloud_win(1);
            clouds=cloudup(clouds,cloud_win,ler(i,:),i);
        end
        gamma=compgam(clouds,ler(i,1:at));

    end

    % vypocet normalizovane lokalni hustoty
    for j=1:clouds.cnt
        lambda(j)=gamma(j)/sum(gamma);
    end

    %
    [clouds,lambda]=uti_age_up(clouds,i,lambda);

    % "Vitez bere vse"
    q=find(lambda==max(lambda));
    lambda=zeros(1,clouds.cnt);
    q=q(1); %pro pripad shody
    lambda(q)=1;

    %tvorba zaveru
    clouds=wRMSfcn_klas(clouds,ler(i,:),lambda,i,omega);

    % online vystup
    yo(i,:)=zeros(1,clouds.py);
    for j=1:clouds.cnt
        yo(i,:)=yo(i,:)+(clouds.store(j).param*lambda(j));
    end
```

Při přijetí vzorku vstupně-výstupních dat algoritmus vypočítá vektor globálních hustot dat pro vstupy vzorku dat a výstupy aproximovanými jednotlivými Mračny Γ_k^i díky funkci *globalfcntmp*. V uvedeném kódu je varianta pro klasifikaci. Tato funkce musí být před samotným výpočtem globální hustoty datové struktury, jelikož počítá s hodnotami předchozího vzorku dat a neupravuje samotnou strukturu „clouds“. Dále se spočítá skutečná globální hustota pomocí funkce *globalfcn* a vektor lokálních hustot jednotlivých Mračen s novým vzorkem dat (spočítá se pouze hypotetická lokální hustota Mračen, ještě není rozhodnuto, ke kterému Mračnu nový vzorek dat patří). Nyní máme všechny informace k tomu, abychom rozhodli, co s novým vzorkem dat. Provede se vyhodnocení podle rovnic (3.13), (3.14) a na základě toho nastanou dvě situace:

1) Bude vytvořeno nové Mračno (funkce *cloudgen*)

2) Vzorek dat bude přiřazen nejbližšímu Mračnu (fuknce *cloudup*)

Opět se spočítá vektor lokálních hustot jednotlivých Mračen, nyní již s potencionálně nově vytvořeným Mračnem a převede se na normalizovanou hodnotu λ_k^i podle rovnice (3.11). Pomocí funkce *uti_age_up* se zhodnotí kvalita Mračen. V našem případě je v kódu přítomna metoda upravení normalizované hodnoty λ_k^i podle pravidla „vítěz bere vše“ (3.20). Poslední částí algoritmu je tvorba závěrů pomocí funkce *wRMSfcn*, zde varianta pro klasifikaci a poslední řádek reprezentuje výstup online režimu algoritmu. Poslední část reprezentuje online výstup algoritmu.

Funkce na vytváření nových Mračen

Hlavní část algoritmu využívá tuto funkci *cloudgen* pro tvorbu nových Mračen. Vstupem je datová struktura „clouds“, vzorek vstupních dat „data“ a časový index tvorby nového mračna „time“. Výstupem je upravená datová struktura „clouds“.

```
function [clouds] = cloudgen_klas( clouds,data,time)
%CLOUDGEN Vytvoreni noveho mracna
%   Input: clouds,data,time Output: clouds

    clouds.cnt=clouds.cnt+1;

    clouds.store(clouds.cnt).history.true=data(clouds.at+1:end);
    data=data(1:clouds.at);

    clouds.datacnt=clouds.datacnt+1;
    clouds.store(clouds.cnt).history.data=data;
    clouds.store(clouds.cnt).history.write=time;
    clouds.store(clouds.cnt).timesum=time;
    clouds.store(clouds.cnt).lambdasum=0;
    clouds.store(clouds.cnt).datacnt=1;
    clouds.store(clouds.cnt).history.age=[];
    clouds.store(clouds.cnt).history.utility=[];
    clouds.store(clouds.cnt).history.lambda=[];

    clouds.store(clouds.cnt).data_mean=data;
    clouds.store(clouds.cnt).data_disp=norm(data)^2;
    clouds.store(clouds.cnt).loc_density=1/(1+norm(data...
        -clouds.store(clouds.cnt).data_mean)^2+clouds.store(clouds.cnt).data_disp...
        -norm(clouds.store(clouds.cnt).data_mean)^2);

    clouds.store(clouds.cnt).param=zeros(1,clouds.py);
end
```

Funkce nastavuje výchozí hodnoty nového Mračna, ukládá data do historie a počítá inicializační hodnoty střední hodnoty dat μ_1^L (3.5), lokálního rozptylu dat Σ_1^L (3.6) a lokální hustoty dat γ_1^i (3.4) nového Mračna. V posledním řádku je inicializace parametrů závěrů nového mračna. V tomto případě je závěr typu *singleton* nebo *enum*, jelikož se jedná o klasifikaci.

Funkce na aktualizaci vítězného Mračna

Tato funkce je podobná funkci tvorby nového Mračna. Vstupem je datová struktura „clouds“, index upravovaného Mračna „cloud_num“, vzorek vstupních dat „data“ a časový index „time“. Výstupem je upravená datová struktura „clouds“.

```
function [clouds] = cloudup( clouds,cloud_num,data,time)
%CLOUDGEN Aktualizace mracna
%   Input: clouds,cloud_num,data,time Output: clouds

    clouds.store(cloud_num).history.true...
        =[clouds.store(cloud_num).history.true;data(clouds.at+1:end)];

    data=data(1:clouds.at);
    clouds.datacnt=clouds.datacnt+1;

    clouds.store(cloud_num).history.data=[clouds.store(cloud_num).history.data;data];
    clouds.store(cloud_num).history.write=[clouds.store(cloud_num).history.write,time];
    clouds.store(cloud_num).timesum=clouds.store(cloud_num).timesum+time;
    clouds.store(cloud_num).datacnt=clouds.store(cloud_num).datacnt+1;

    clouds.store(cloud_num).data_disp=((clouds.store(cloud_num).datacnt-1)...
        /clouds.store(cloud_num).datacnt)*clouds.store(cloud_num).data_disp...
        +(1/clouds.store(cloud_num).datacnt)*norm(data)^2;

    clouds.store(cloud_num).data_mean=((clouds.store(cloud_num).datacnt-1)...
        /clouds.store(cloud_num).datacnt)*clouds.store(cloud_num).data_mean...
        +(1/clouds.store(cloud_num).datacnt)*data;

    clouds.store(cloud_num).loc_density=1/(1+norm(data...
        -clouds.store(cloud_num).data_mean)^2+clouds.store(cloud_num).data_disp...
        -norm(clouds.store(cloud_num).data_mean)^2);

end
```

Podobně jako funkce pro vytvoření nového Mračna, ukládá funkce pro aktualizaci Mračna data do historie, upravuje proměnné „timesum“, „datacnt“ a počítá hodnoty střední hodnoty dat μ_k^L (3.5), lokálního rozptylu dat Σ_k^L (3.6) a lokální hustoty dat γ_k^i (3.4) upravovaného Mračna. Tato funkce je univerzální pro všechny testované úlohy algoritmu.

Funkce na spočtení globální hustoty

Funkce se stará o výpočet globální hustoty datové struktury fuzzy systémů AnYa. Vstupem je datová struktura „clouds.global“ a vzorek dat „data“. Výstupem je upravená datová struktura „clouds.global“.

```
function [ glob ] = globalfcn( glob,data )
%GLOBALFCN Funkce na spocteni globalni hustoty
%   Input: clouds.global,data(:),time index   Output: clouds.global
    if (isempty(glob))
        glob.time=time;
        glob.disp=norm(data)^2;
        glob.mean=data;
        glob.density=1/(1+norm(data-glob.mean)^2+glob.disp-norm(glob.mean)^2);
    else
        glob.time=time;
        glob.disp=((glob.time-
1)/glob.time)*glob.disp+(1/glob.time)*norm(data)^2;
        glob.mean=((glob.time-1)/glob.time)*glob.mean+(1/glob.time)*data;
        glob.density=1/(1+norm(data-glob.mean)^2+glob.disp-norm(glob.mean)^2);
```

```
    end
end
```

Funkce řeší dva případy - pokud je datová struktura „clouds.global“ prázdná, tak funkce inicializuje její proměnné, pokud není, tak vypočítá aktuální hodnoty. Jedná se o globální střední hodnotu dat μ_k (inicializace i výpočet podle rovnice (3.9)), globální rozptyl dat Σ_k (podle rovnice (3.10)) a globální hustotu dat Γ_k (podle rovnice (3.8)). Tato funkce je opět univerzální pro všechny testované úlohy algoritmu.

Funkce na výpočet lokální hustoty Mračen

Pro výpočet lokální hustoty všech Mračen je použita funkce *compgam*. Tato funkce neupravuje datovou struktur „clouds“. Vstupem je struktura „clouds“ a vzorek dat „data“. Výstupem je vektor lokálních hustot Mračen „gamma“.

```
function [ gamma ] = compgam( clouds,data)
%COMPGAM Funkce na spočtení lokalni hustoty gamma
%   Input: clouds,data(:)   Output: gamma

    for i=1:clouds.cnt
        E=((clouds.store(i).datacnt)/(clouds.store(i).datacnt+1))...
            *clouds.store(i).data_disp+(1/(clouds.store(i).datacnt+1))*norm(data)^2;

        u=((clouds.store(i).datacnt)/(clouds.store(i).datacnt+1))...
            *clouds.store(i).data_mean+(1/(clouds.store(i).datacnt+1))*data;

        gamma(i)=1/(1+norm(data-u)^2+E-norm(u)^2);
    end
end
```

Stejně jako v případě aktualizace Mračna je lokální hustota dat γ_k^i jednotlivých Mračen vypočtena pomocí rovnic (3.5), (3.6) a (3.4). Tato funkce je univerzální pro všechny testované úlohy algoritmu.

Funkce na výpočet vektoru globálních hustot pro data s výstupem aproximovaným podle jednotlivých Mračen

Tato funkce je odvozena od funkce pro výpočet globální hustoty *globalfcn*. Rozdíl je, že tato funkce neupravuje datovou strukturu „clouds.global“, ale pouze počítá s jejími proměnnými. Tato struktura „clouds.global“ je jejím vstupem spolu s vzorkem dat „data“ a celou strukturou „clouds“. Výstupem je vektor odhadovaných globálních hustot „glob_dens“.

```
function [ glob_dens ] = globalfcntmp_klas( glob,data,clouds )
%GLOBALFCN_KLAS Funkce na vypocet globalni hustoty pro jednotlive predikce
%   Input:  clouds.global,data(:),clouds    Output: global(1:n)
%   Pouze pro klasifikaci nebo enum parametr

    for i=1:clouds.cnt

        data=[data(1:clouds.at),clouds.store(i).param];
```

```
disp=((glob.time)/(glob.time+1))*glob.disp+(1/(glob.time+1))*norm(data)^2;
        mean=((glob.time)/(glob.time+1))*glob.mean+(1/(glob.time+1))*data;
        glob_dens(i)=1/(1+norm(data-mean)^2+disp-norm(mean)^2);
    end

end
```

Funkce spočítá globální hustotu pro vzorek dat s výstupem aproximovaným každým Mračnem. Výstupem je tedy vektor globálních hustot Γ_k^i. Globální hustoty Γ_k^i jsou opět počítány podle rovnic (3.9), (3.10) a (3.8). Zobrazená verze je pro závěry typu *singleton* nebo *enum*, jelikož se jedná o verzi pro klasifikaci.

Funkce pro údržbu Mračen

Důležitou částí algoritmu pro kontrolu kvality je funkce *uti_age_up*. Tato funkce monitoruje stáří Mračen a jejich užitečnost. Vstupem je datová struktura „clouds“, časový index „time“ a vektor normalizovaných lokálních hustot „lambda“. Výstupem je upravená datová struktura „clouds“ a upravený vektor „lambda“.

```
function [ clouds ,lambda] = uti_age_up( clouds,time,lambda )
%UTI_AGE_UP     Funkce pro udrzbu mracen
%   Input: clouds, time index, lambda   Output: clouds, lambda

for i=1:clouds.cnt
    % age
    clouds.store(i).age=time-((1/clouds.store(i).datacnt)*clouds.store(i).timesum);
    if (isempty(clouds.store(i).history.age))
        clouds.store(i).history.age=clouds.store(i).age;
    else
        clouds.store(i).history.age=[clouds.store(i).history.age…,clouds.store(i).age];
    end

    % utility
    clouds.store(i).lambdasum=clouds.store(i).lambdasum+lambda(i);

    clouds.store(i).utility=(1/(time-(clouds.store(i).history.write(1)-1)))...
        *clouds.store(i).lambdasum;

    if (isempty(clouds.store(i).history.utility))
        clouds.store(i).history.utility=clouds.store(i).utility;
        clouds.store(i).history.lambda=lambda(i);
    else
        clouds.store(i).history.utility=[clouds.store(i).history.utility...
            ,clouds.store(i).utility];

        clouds.store(i).history.lambda=[clouds.store(i).history.lambda,lambda(i)];
    end

    if (clouds.store(i).utility<0.1)
        lambda(i)=0;
        lambda=lambda*(1/sum(lambda));
    end

end

end
```

V první části funkce vypočte stáří Mračen age^i podle rovnice (3.17) a (3.16) a uloží tuto hodnotu do historie pro monitorování stáří v čase. Druhá část zhodnotí užitečnost Mračen U_k^i

pomocí rovnice (3.18), opět uloží hodnotu do historie, poté na základě rovnice (3.19) rozhodne o úpravě vektoru normalizovaných lokálních hustot λ_k^i a aktualizuje jejich hodnotu tak, aby se jejich součet rovnal jedné. I tato funkce je univerzální.

Funkce pro tvorbu a úpravu závěru

Pro funkci fuzzy systému typu AnYa je velmi důležitá část tvorby závěrů (konsekventy). O tuto část se v algoritmu stará funkce *wRMSfcn*. Tato funkce se zároveň nejvíce liší mezi jednotlivými verzemi algoritmu pro různé úlohy. Vstupem je datová struktura „clouds“, vzorek dat „data“, vektor normalizovaných lokálních hustot „lambda“ a koeficient učení „omega“. Výstupem je upravená datová struktura „clouds“.

```
function [ clouds ] = wRMSfcn_klas( clouds,data,lambda,omega )
%WRMSFCN_KLAS   Funkce pro upravu zaveru - verze pro singleton,enum vystup
%   Input: clouds, data(:), lambda, time index, omega
%   Output: clouds
%   Moznost zmeny parametru na singleton ci enum

    if isempty(clouds.RMS)
        xi=clouds.store(1).param';
        clouds.RMS.C=omega*eye(clouds.cnt);
        clouds.RMS.phi=(clouds.store(1).param')';
    end

    [cx,cy]=size(clouds.RMS.C);
    [cx1,cy1]=size(omega*eye(clouds.cnt));
    if ((cx~=cx1) || (cy~=cy1))
        clouds.RMS.C=[clouds.RMS.C,zeros(cx,1);zeros(1,cy),1];
        clouds.RMS.phi=[clouds.RMS.phi;zeros(1,clouds.py)];
    end

    xi=[];
    for i=1:clouds.cnt
        xi=[xi,lambda(i)];
    end
    xi=xi';

    clouds.RMS.C=clouds.RMS.C-((clouds.RMS.C*xi*xi'*clouds.RMS.C)...
        /(1+xi'*clouds.RMS.C*xi));

    clouds.RMS.phi=clouds.RMS.phi+clouds.RMS.C*xi*(data(clouds.at...
        +1:end)-xi'*clouds.RMS.phi);

    for i=1:clouds.cnt
        %pro singleton parametr
        %clouds.store(i).param=clouds.RMS.phi(i);

        %pro enum parametr
        if (clouds.store(i).datacnt==1)
            clouds.store(i).param=clouds.store(i).history.true;
        else
            clouds.store(i).param=mode(clouds.store(i).history.true);
        end
    end
end
```

Funkce inicializuje hodnoty matice C_k (rovnice 3.24) a vektoru $\hat{\theta}_k$ (rovnice 3.23), pokud se jedná o první datový vzorek na prázdné struktuře „clouds“. Poté ověří, zdali bylo vytvořeno nové Mračno. V kladném případě vhodně rozšíří matici C_k a vektor $\hat{\theta}_k$ tak, aby bylo možné pokračovat ve výpočtu. Následné vytvoří fuzzy vážený vektor vstupů ψ a provede výpočet parametrů závěrů podle rovnic (3.24), (3.23). Nakonec upraví parametry jednotlivých Mračen. V tomto případě se jedná o verzi pro klasifikaci, která je určena pro závěry typu *singleton* nebo *enum*. Mezi těmito závěry přepínáme okomentováním příslušné varianty a okomentováním varianty druhé. Parametry pro *enum* variantu jsou nejčastěji zastoupené hodnoty výstupu v Mračnu, pro *singleton* variantu je to odhad na základě metody nejmenších čtverců.

3.3.2. Úloha aproximace

V této části budou popsány rozdíly kódu algoritmu oproti úloze klasifikace, jelikož velká část kódu je použitelná pro obě úlohy.

Pro úlohu aproximace je opět určující faktor výstup fuzzy systému. Výstup by měl co nejlépe aproximovat funkce na základě vstupních dat a proto je obvykle výstup typu *real*. Pro funkce, které se dají s úspěchem popsat nějakou lineární kombinací vstupních hodnot je tvorba závěrů stejná, jako byla popsána v kapitole 3. Můžeme ovšem použít i jiných typů výstupních funkcí, jediná změna bude ve funkci pro tvorbu závěrů a v inicializaci vektoru parametrů.

Rozdíly oproti klasifikaci

Hlavní část programu zůstává prakticky nezměněna, pouze používáme funkce určené pro aproximaci lineární kombinací vstupních dat nebo jiných typů závěrů. Pro aproximaci také není výhodné používat metodu „vítěz bere vše“ a proto je i tato část kódu vymazána.

Největší rozdíl je ve funkci *wRMSfcn*:

```
function [ clouds ] = wRMSfcn( clouds,data,lambda,omega )
%WRMSFCN   Funkce pro upravu zaveru - verze vitez bere vse
%   Input: clouds, data(:), lambda, time index, omega
%   Output: clouds
%   Verze pro linearni kombinaci vstupu

    if isempty(clouds.RMS)
        xi=clouds.store(1).param';
        clouds.RMS.C=omega*eye(clouds.cnt*(clouds.at+1));
        clouds.RMS.phi=(clouds.store(1).param')';
    end

    [cx,cy]=size(clouds.RMS.C);
    [cx1,cy1]=size(omega*eye(clouds.cnt*(clouds.at+1)));
    if ((cx~=cx1) || (cy~=cy1))


clouds.RMS.C=[clouds.RMS.C,zeros(cx,clouds.at+1);zeros(clouds.at+1,cy)...
            ,omega*eye(clouds.at+1)];

        clouds.RMS.phi=[clouds.RMS.phi;zeros(clouds.at+1,clouds.py)];
    end
```

```
    xe=[1,data(1:clouds.at)]';
    xi=[];
    for i=1:clouds.cnt
        xi=[xi,lambda(i)*xe'];
    end
    xi=xi';

    clouds.RMS.C=clouds.RMS.C-((clouds.RMS.C*xi*xi'*clouds.RMS.C)/(1+xi'...
        *clouds.RMS.C*xi));
    clouds.RMS.phi=clouds.RMS.phi+clouds.RMS.C*xi*(data(clouds.at+1:end)-
xi'...
        *clouds.RMS.phi);

    for i=1:clouds.cnt
        clouds.store(i).param=clouds.RMS.phi((i-1)*(clouds.at+1)+1:i...
            *(clouds.at+1),1:clouds.py);
    end
end
```

Opět většina kódu zůstává stejná, jelikož se jedná o metodu nejmenších čtverců, ale rozdíl je v tvaru vstupních dat. U klasifikace jsme jako vstupní proměnné používali pouze vektor normovaných lokálních hustot λ_k^i jako ekvivalent hodnoty funkce příslušnosti vzorku dat jednotlivým Mračnům, ale pro aproximaci pomocí lineární kombinace vstupů využíváme fuzzy vážený (pomocí λ_k^i) o jedničku rozšířený vektor vstupních dat. Celá funkce je tedy přepsána pro tento vstupní vektor. Současně je i změněna inicializace a práce s proměnou obsahující parametry Mračen ve funkcích *cloudgen*

```
    clouds.store(clouds.cnt).param=zeros(clouds.at+1,clouds.py);
```

a *globalfcn_tmp*.

```
    xe=[1,data(1:clouds.at)];
    data=[data(1:clouds.at),xe*clouds.store(i).param];
```

Zbytek kódu zůstává totožný.

4. OVĚŘENÍ METODY

Ověření bylo provedeno na datech získaných z UCI Machine Learning Repository (dostupné z http://archive.ics.uci.edu/ml). Byla prověřena funkčnost fuzzy systému AnYa pro úlohu klasifikace a aproximace funkce. Ke každé úloze byly vybrány tři soubory testovacích dat. Ve vybraných případech byl prověřen vliv šumu na výsledek úlohy.

4.1. Klasifikace

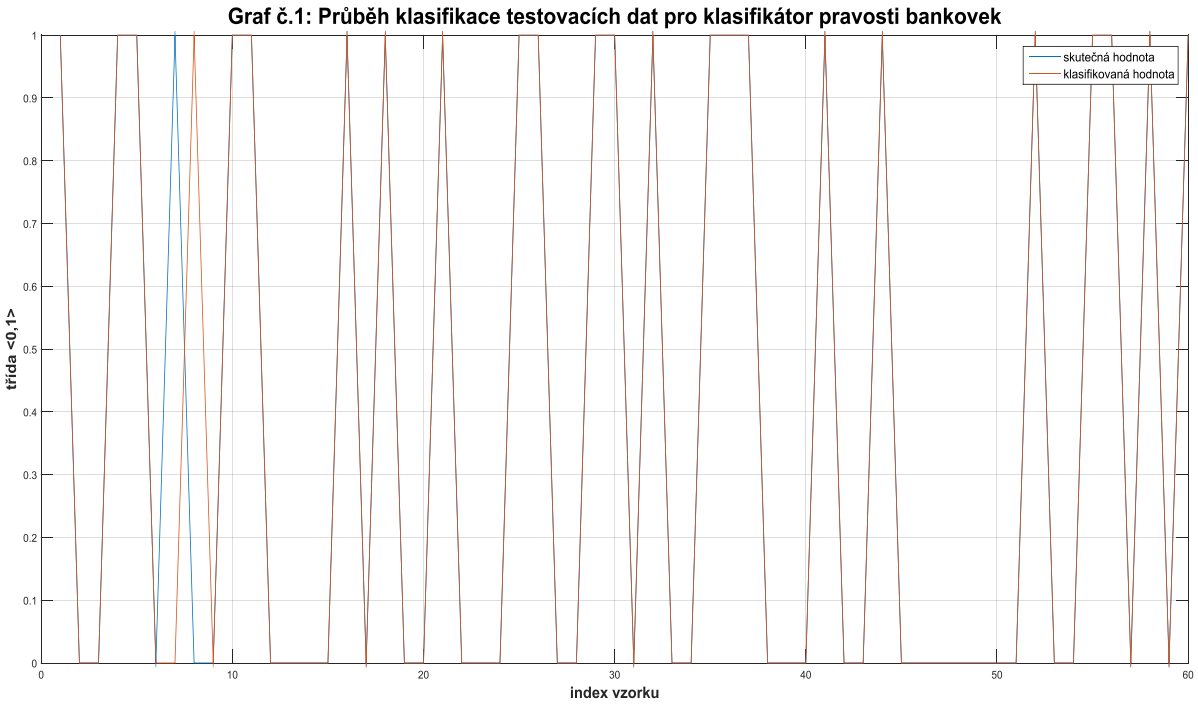
Pro otestování algoritmu byly zvoleny tři úlohy klasifikace. Tyto problémy byly zvoleny pro rozdílné typy vstupních proměnných a jejich rozložení, aby byla prostudována možnost použití fuzzy systému AnYa pro úlohu klasifikátoru.

4.1.1. Klasifikace bankovek

Datový soubor se skládal z 1372 vzorků dat. Jednalo se o reprezentaci fotek pravých a falešných bankovek. Jednotlivé vzorky se skládaly ze 4 reálných atributů, které tvořily vstup systému a jednoho výstupu, který reprezentoval třídu bankovky {pravá, falešná}. Vstupy reprezentují vlastnosti fotky získané její vlnkovou transformací.

Pro klasifikaci byla použita metoda „vítěz bere vše“ a výstup Mračna byla nejčastěji zastoupená hodnota výstupu pro data daného Mračna. Parametr Λ byl zvolen jako e^{-1}.

Pro test klasifikace algoritmu byla data náhodně rozdělena na trénovací a testovací množinu. Trénovací množina obsahovala 800 vzorků a testovací 572 vzorků. Aby byla určena průměrná přesnost klasifikace, tak byl algoritmus spuštěn desetkrát pro různé testovací a trénovací množiny. Průměrná přesnost klasifikace činila 96,75% a k dosáhnutí klasifikace bylo průměrně vytvořeno 119 Mračen.



Graf č. 1: Průběh klasifikace testovacích dat pro klasifikátor pravosti bankovek

Na grafu 1 lze vidět průběh klasifikace pro vzorek 70ti testovacích dat. Dosažená přesnost byla 96,67%, dva vzorky nebyly určeny správně. Díky velkému množství vytvořených mračen by byly průběhy vývoje užitečnosti či stáří mračen velice nepřehledné, proto zde nebudou uvedeny. Pro otestování vlivu šumu v datech na klasifikaci byl algoritmus naučený na nezašuměná data a testovací data obsahovala šum až 75%. Průměrná přesnost klasifikace po deseti pokusech byla 87,43%. Varianta, kdy byl algoritmus učen na zašuměná data (až 75%) a testovací data byla nezašuměná, dosáhl po deseti pokusech průměrné přesnosti 88,74% ale v průměru bylo vytvořeno o 30% více Mračen. Při variantě šumu až 75% v obou souborech dat byla průměrná přesnost klasifikace 82,15%. Z těchto výsledků vyplývá, že šum neměl výrazný vliv na klasifikaci.

4.1.2. Klasifikace onemocnění [6]

Další datový soubor pro klasifikaci pomocí fuzzy systému AnYa je soubor určený pro expertní systémy. Jedná se o smíšené hodnoty typu *real* a *integer*. Data se skládají z šesti vstupních dat (teplota ve stupních celsia a pět odpovědí ano/ne), výstupem jsou dvě diagnózy (ano/ne). Jedná se o zánět močového měchýře a Nephrititis. Soubor dat obsahoval 120 vzorků. Odpovědi byly kódovány (kvůli výhodnějšímu rozložení dat – zjištěno na základě analýzy počtu Mračen, užitečnosti a stáří Mračen při jiném kódování):

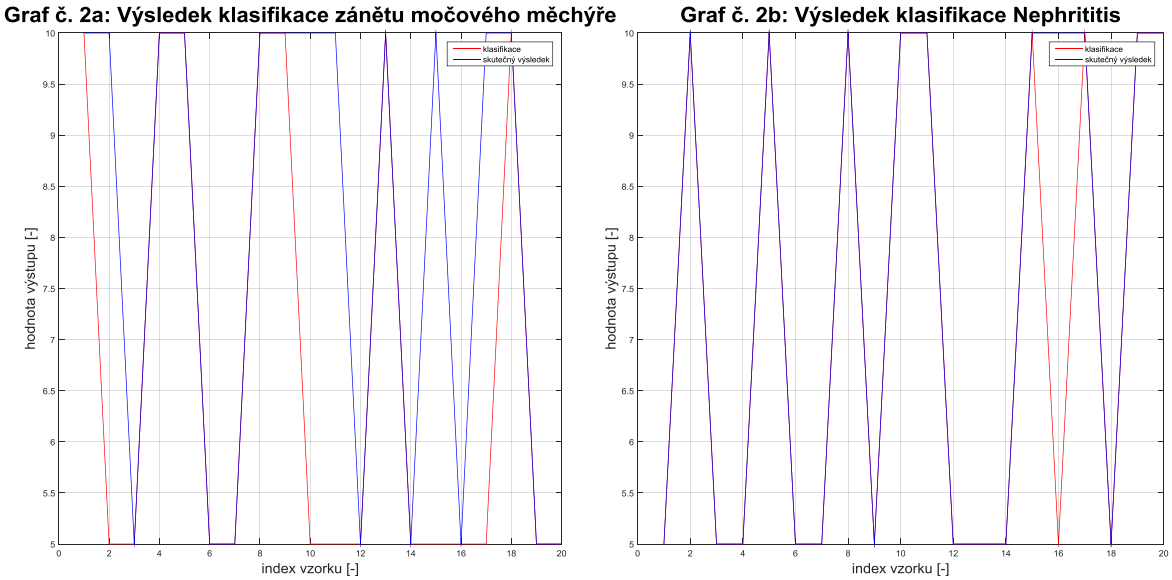
ANO = 1, NE = -1 pro vstupní proměnné.

ANO=10, NE=5 pro výstupní proměnné.

Závěry byly vytvořeny na základě nejčastěji se opakujícího výsledku dat jednotlivých Mračen a byla použita metoda „vítěz bere vše“. Parametr překrytí byl zvolen e^{-1}.

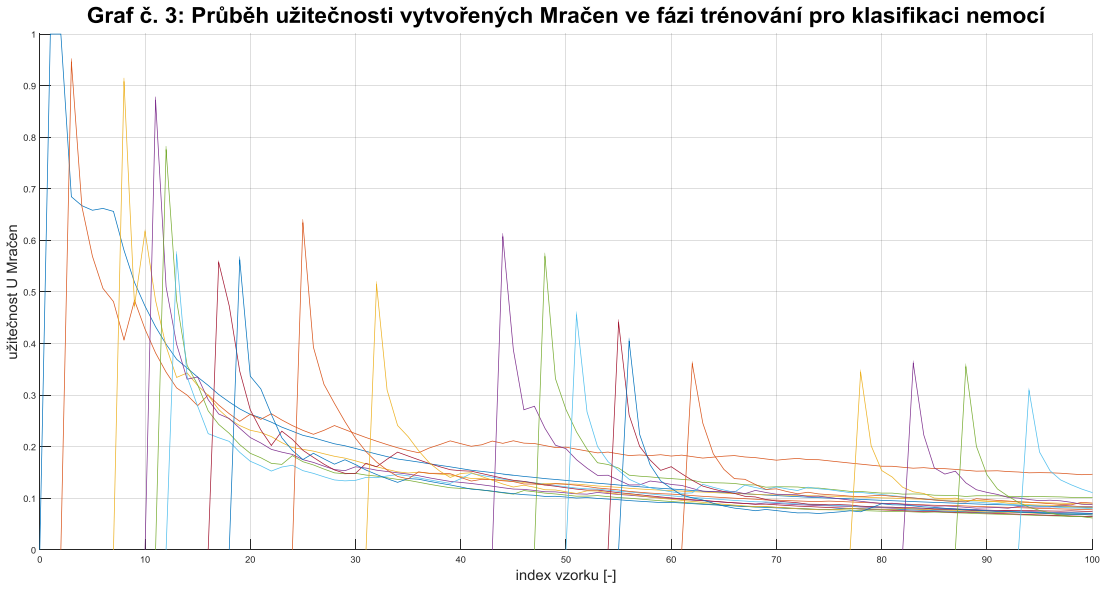
Data byla náhodně rozdělena na trénovací a testovací množinu. Trénovací množina obsahovala 100 vzorků, testovací pouze 20.

Průměrná přesnost klasifikace pro deset opakování byla 88,75% a systém vytvořil 16 Mračen.

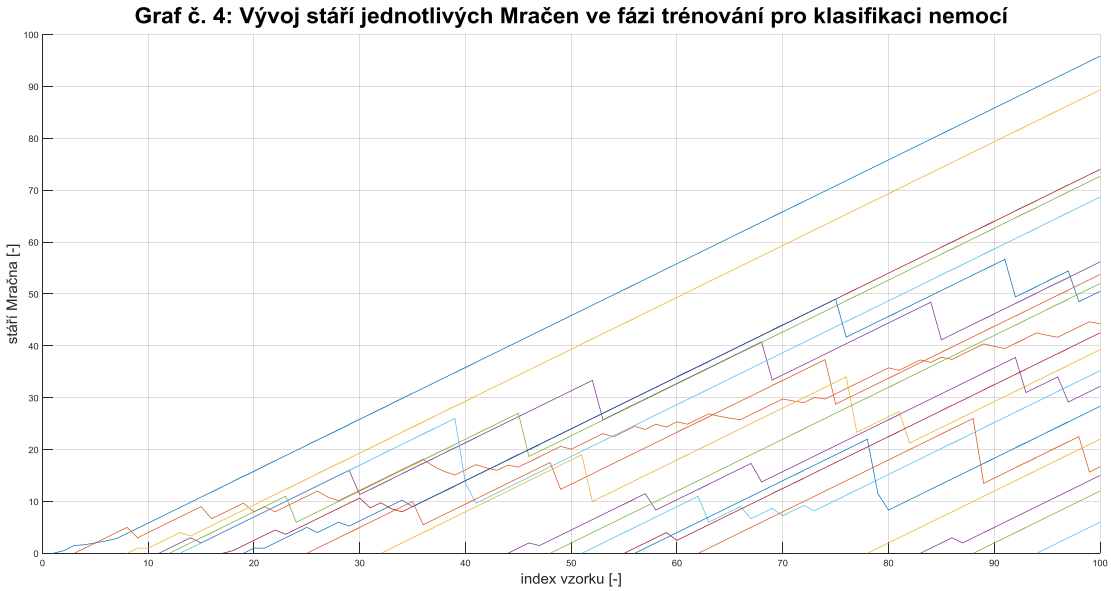


Graf č. 2: Graf výsledků klasifikace nemocí

Z grafu 2a a 2b lze vidět, že algoritmus lépe hodnotí druhý výstup.



Graf č. 3: Průběh užitečnosti Mračen pro klasifikaci nemocí



Graf č. 4: Průběh stáří Mračen pro klasifikaci nemocí

Z grafů 3 a 4 vyplývá, že nejužitečnější Mračno mělo index 2 (druhé vytvořené – červená linka). Jeho užitečnost se držela na 0,2.

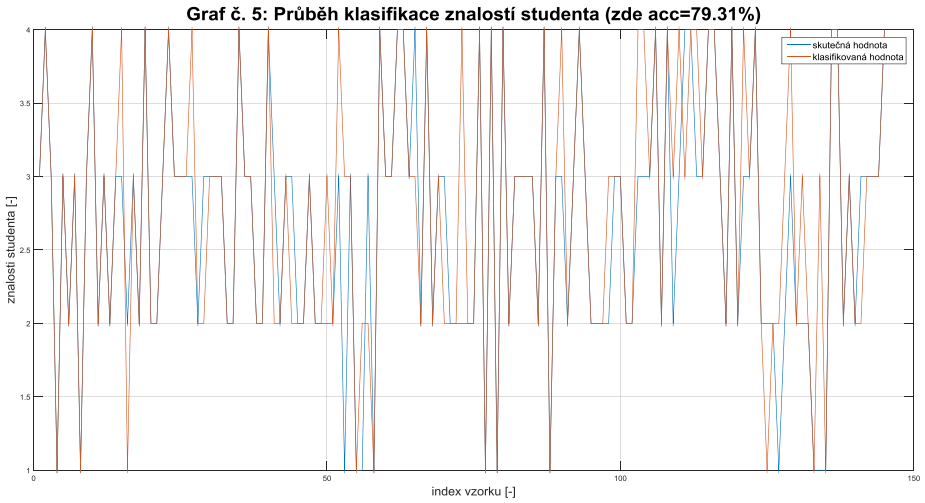
Vzhledem k typu vstupních dat je míra přítomného šumu irelevantní pokud by nedosahovala vysokých hodnot.

4.1.3. Klasifikace znalostí studentů [5]

Poslední datový soubor pro klasifikaci reprezentuje 257 vzorků reprezentujících znalosti studentů u zkoušek. Vstupními daty je 5 proměnných s hodnotami <0,1>, které reprezentují aspekty jako studijní čas, výkon u zkoušky atp. Výstupem je typ *enum* - ohodnocení zkoušejícího {very low, low, medium, high}. V algoritmu tyto hodnoty reprezentují čísla {1,2,3,4}.

Opět byla použita metoda „vítěz bere vše“ a parametr překrytí e^{-1}. Trénovací vzorek byl složen ze všech vzorků a pro otestování klasifikace bylo náhodně vybráno 145 vzorků. Po deseti cyklech trénování-klasifikace byla průměrná přesnost klasifikace 76,14%. Počet vytvořených Mračen byl 78. Vysoký počet vytvořených Mračen naznačuje, že pro tento datový soubor metoda není příliš vhodná.

Autoři článku tento datový soubor klasifikovali pomocí několika pokročilých algoritmů. Dosahovali přesnosti i 95%. Za zmínku ovšem stojí, že Bayesův klasifikátor dosahoval přesnosti pouze 73%, tudíž klasifikoval hůř než systém AnYa.



Graf č. 5: Průběh klasifikace znalostí studenta

4.2. Aproximace

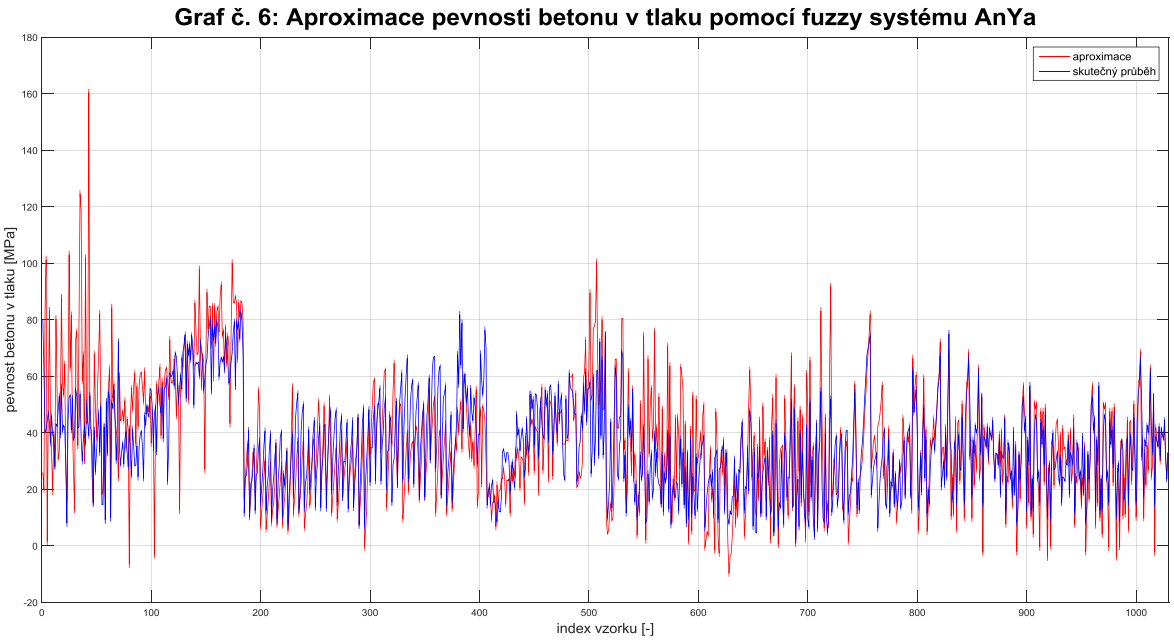
Pro otestování algoritmu byly zvoleny tři úlohy aproximace.

4.2.1. Aproximace pevnosti v tlaku betonu [7]

Pro aproximaci pevnosti tlaku v betonu byl využit datový soubor 1030 vzorků o sedmi vstupních proměnných a výstupní hodnotou pevnosti betonu v tlaku. Cílem byly tuto hodnotu aproximovat jako funkci vstupních proměnných. Beton byl popsán poměry materiálů ve směsi.

Pravidla byla tvořena lineární kombinací vstupních proměnných, $\Omega = 10$, parametr překrytí byl zvolen e^{-1}.

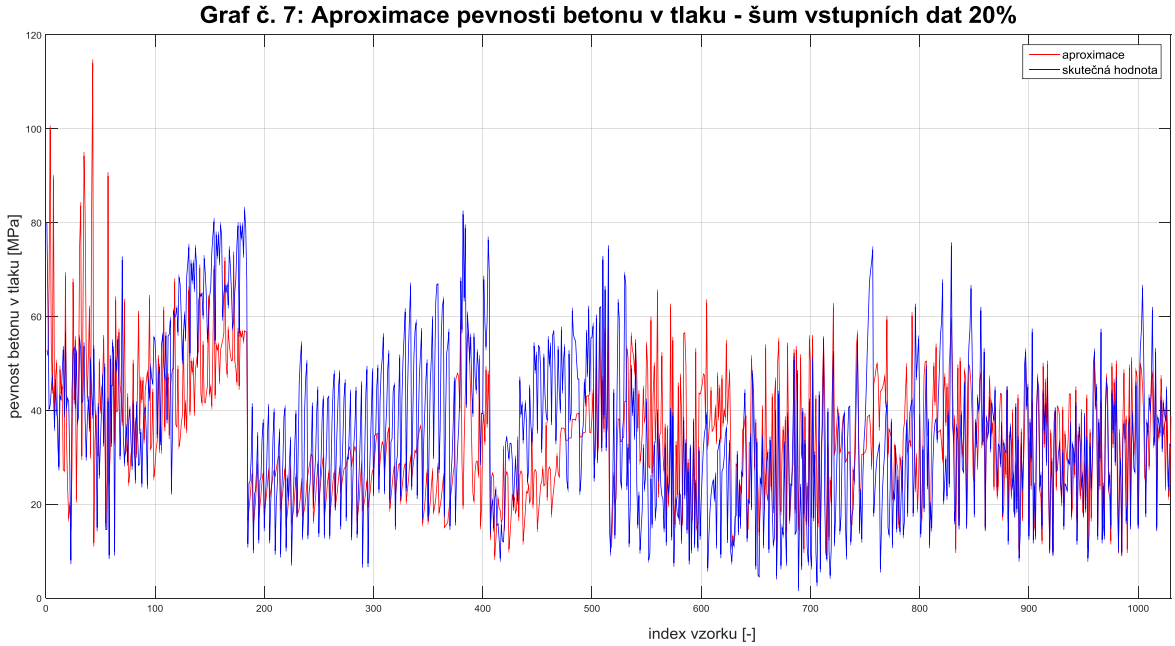
Algoritmus vytvořil pro aproximaci 188 Mračen. Výsledek aproximace je zobrazen v grafu č. 6.



Graf č. 6: Průběh aproximace pevnosti betonu

Lze vidět, že ačkoliv aproximace není přesná, lze z ní vyčíst trend vývoje pevnosti v tlaku betonu.

Je-li ve vstupech trénovací množiny přítomen šum 20%, pak aproximace vypadá následovně (graf č. 7):



Graf č. 7: Průběh aproximace pevnosti betonu (šum 20%)

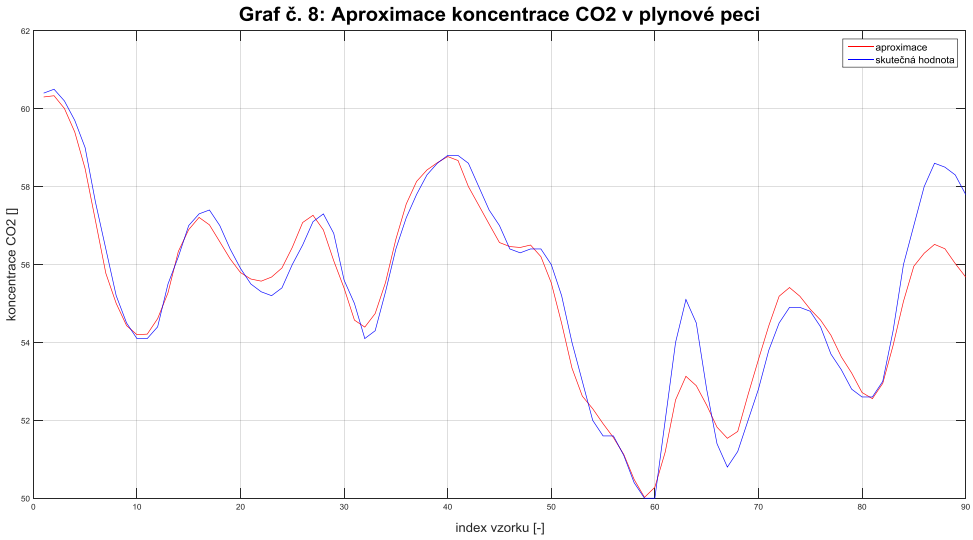
Je patrné, že má šum na aproximaci výrazný vliv.

4.2.2. Aproximace koncentrace CO2 v laboratorní plynové peci

Pro ověření pochopení konceptu byla provedena stejná aproximace jako v [2]. Aproximovaná funkce byla koncentrace CO2 v plynové peci. Data se skládala ze dvou vstupů a jednoho výstupu. Podle [2] lze tuto funkci nejlépe aproximovat jako:

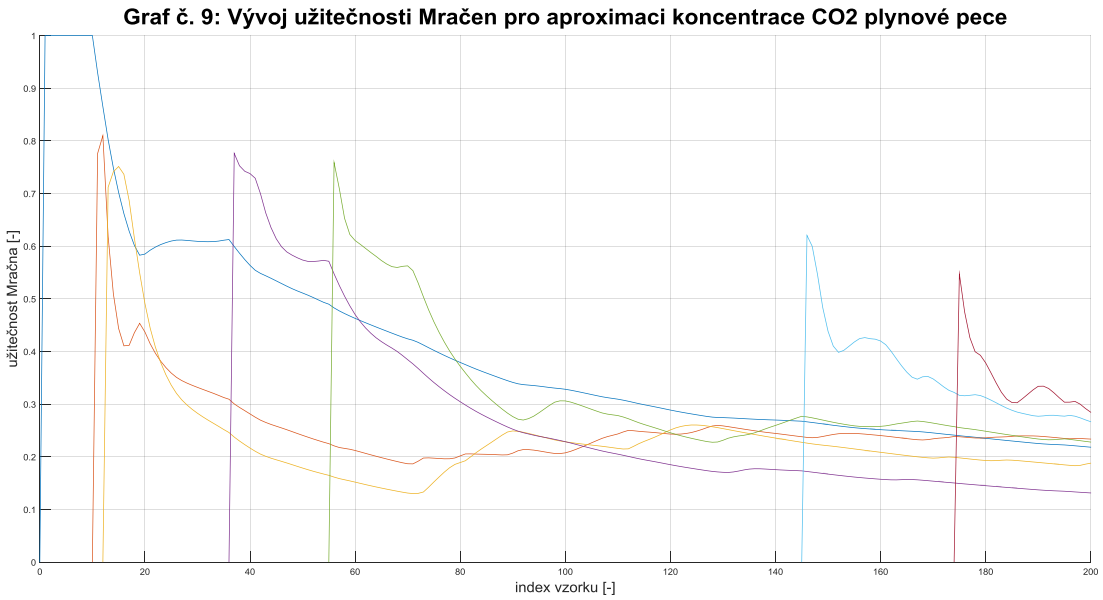
$$y(k) = f(y(k-1), u(k-4))$$

Proto vstupem jsou hodnoty $y(k-1)$ a $u(k-4)$. Trénovací množinu tvořilo 200 vzorků, testovací množina měla 90 vzorků. Hodnota $\Lambda = e^{-1}$, $\Omega = 10^9$ a výstupy Mračen byly lineární kombinace proměnných.

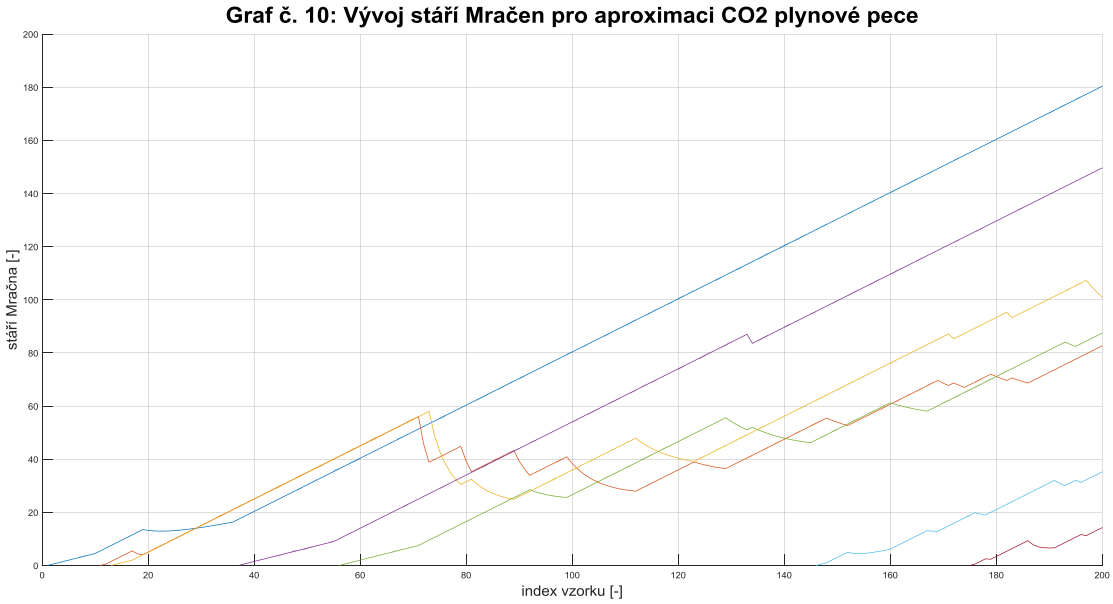


Graf č. 8: Aproximace koncentrace CO2 plynové pece

Aproximace se aproximované funkci blíží. Algoritmus vytvořil 7 Mračen, což je totožná hodnota jako v [2]. Vývoj stáří a užitečnosti Mračen je zobrazen na grafech 9 a 10.

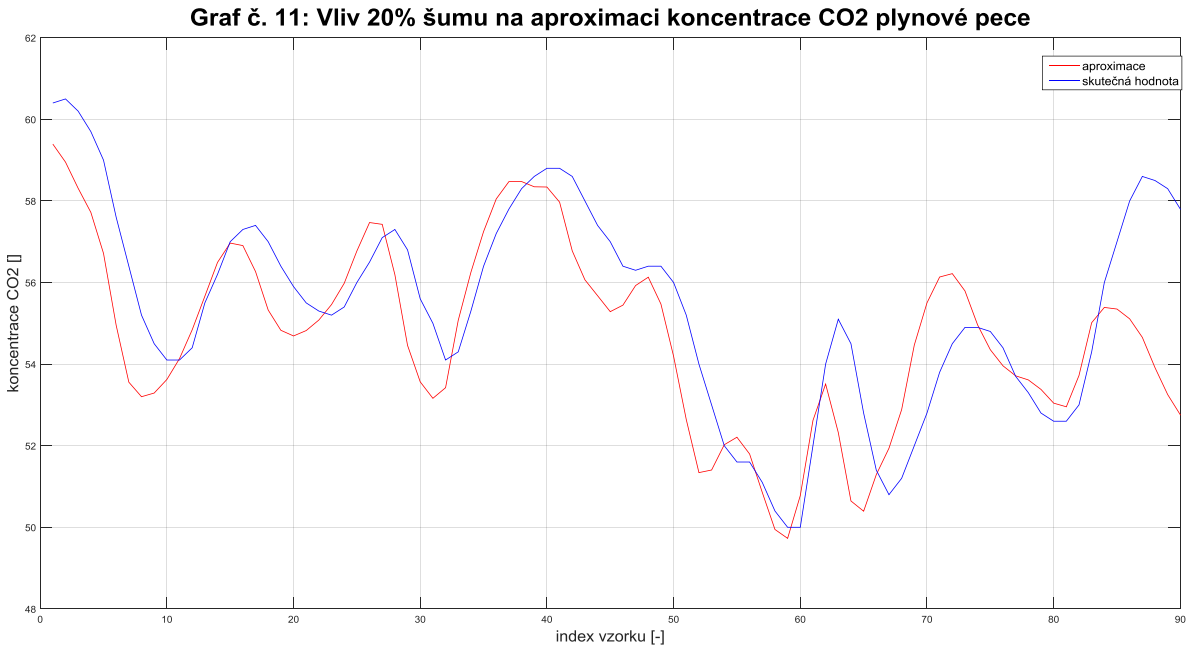


Graf č. 9: Průběh užitečností Mračen pro aproximaci koncentrace CO2 plynové pece



Graf č. 10: Průběh stáří Mračen pro aproximaci koncentrace CO2 plynové pece

Přidáním 20% šumu do trénovacích dat ovšem aproximaci zhorší (Graf č. 11)

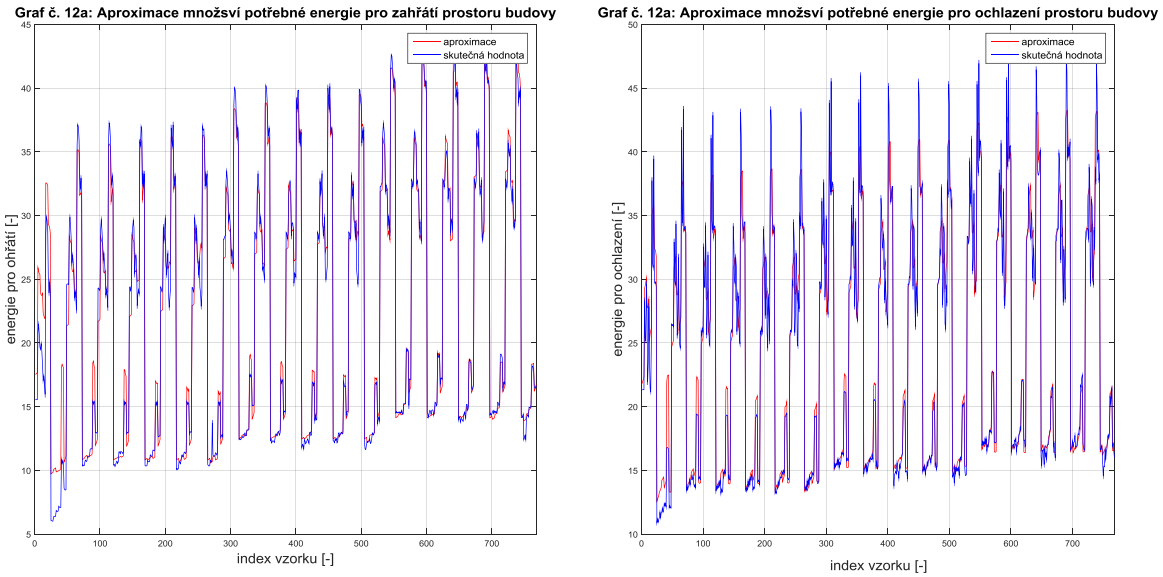


Graf č. 11: Průběh aproximace koncentrace CO2 plynové pece (šum 20%)

4.2.3. Aproximace energetické účinnosti budovy [8]

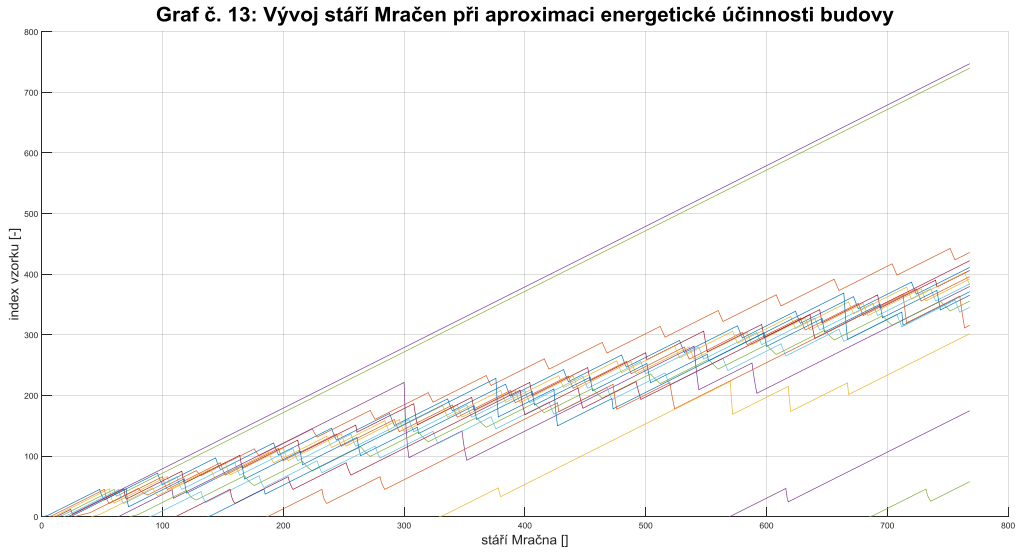
Tento datový soubor byl vybrán jako příklad aproximace více proměnných systémem AnYa. Jedná se o soubor 768 vzorku dat energetické analýzy 12 různých tvarů budov při rozdílných podmínkách. Vstupem je 8 parametrů budovy a výstupem jsou hodnoty energie potřebné pro zahřátí/ochlazení prostoru budovy.

Hodnota $\Lambda = e^{-1}$, $\Omega = 100$, výstupy Mračen byly lineární kombinace proměnných. Systém vytvořil 19 Mračen.

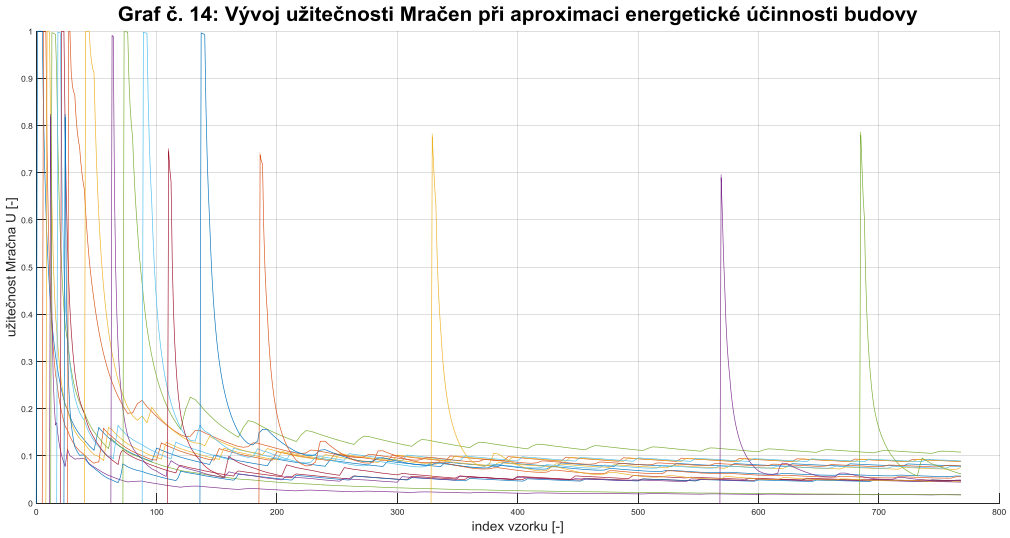


Graf č. 12: Průběhy aproximace energetické účinnotsi budov

Lze vidět, že obě výstupní hodnoty jsou věrně aproximovány.



Graf č. 13: Průběh vývoje stáří Mračen při aproximaci energetické účinnosti budov



Graf č. 14: Průběh vývoje užitečností Mračen při aproximaci energetické účinnosti budov

Z grafů 13 a 14 lze posoudit, že pro aproximaci jsou frekventovaně používány všechny Mračna kromě dvou.

4.3. Zhodnocení

Z uvedených příkladů použití fuzzy systémů AnYa pro aproximaci funkci a klasifikace vyplynulo, algoritmus je schopný fungovat, aniž by byla potřebná znalost vstupních dat. Při klasifikaci je potřeba se zamyslet nad výstupním vektorem a zvolit vhodný typ závěrů. Při aproximaci není potřeba znát nic než počet výstupních proměnných.

Při klasifikaci i aproximaci dosahoval systém solidních výsledků. Vliv šumu se výrazněji projevil u aproximace, u klasifikace byl jeho vliv zanedbatelný.

5. ZÁVĚR

V souladu s cíly diplomové práce byla prošetřena možnost cloud přístupu ve vstupně-výstupním datovém prostoru fuzzy systému. Takovým chováním se vyznačuje nový fuzzy systém AnYa. V rámci práce byl nový fuzzy systém popsán na základě [1] a [2]. Z popisu vyplývají výhody nového fuzzy systému v oblasti tvorby antecedentů fuzzy pravidel systému oproti stávajícím nejpoužívanějším systémům typu Mamdani či Takagi-Sugeno. Zejména jde o neparametričnost vstupně-výstupního prostoru fuzzy systémů a tudíž pro svou funkci nevyžaduje znalost těchto dat.

Nový fuzzy systém byl posléze realizován v programu MATLAB a v závěrečné části byl otestován na různých souborech dat pro různé úlohy. Systém AnYa se ukázal jako vhodný pro aproximaci funkcí, avšak také se projevila jeho náchylnost k šumu ve vstupních datech. Čím více jsou si data podobná, tím víc může šum způsobit nesprávné zařazení nových dat, čímž se ovlivní výsledek systému.

Při klasifikaci se vliv šumu téměř eliminoval. Z části za to může dobré rozložení vstupních dat v prostoru a z části tvorba závěrů systému pro klasifikaci, která probíhala metodou „vítěz bere všechno“. V samotných úlohách klasifikace dat si systém nevedl špatně a dosahoval přesností srovnatelných s klasickými klasifikačními metodami (Bayesův klasifikátor, k-nn…).

Celkově se zdá být nový systém typu AnYa přínosem pro fuzzy počty s neznámými daty. Další dobrou vlastností je jeho jednoduchost.

LITERATURA

[1] ANGELOV, Plamen. *Autonomous Learning Systems: From Data Streams to Knowledge in Real-time*. UK: John Wiley & Sons, 2012. ISBN 978-1-119-95152.

[2] ANGELOV, Plamen a Rona YAGER. A simple fuzzy rule-based system through vector membership and kernel-based granulation. In: *2010 5th IEEE International Conference Intelligent Systems*. New york city, NY: Ieee, 2010, s. 349-354. ISBN 9781424451647. DOI: 10.1109/is.2010.5548369.

[3] Jura P.: Základy fuzzy logiky pro řízení a modelování, nakladatelství VUTIUM 2003

[4] KLIR, George J. *Fuzzy Sets and Fuzzy Logic Theory and Applications*. 1. vyd. New Jersey: Prentice Hall, 1995, 574 s. ISBN 01-310-1171-5.

[5] H. T. Kahraman, Sagiroglu, S., Colak, I., Developing intuitive knowledge classifier and modeling of users' domain dependent data in web, *Knowledge Based Systems*, vol. 37, pp. 283-295, 2013.

[6] J.Czerniak, H.Zarzycki, Application of rough sets in the presumptive diagnosis of urinary system diseases, *Artifical Inteligence and Security in Computing Systems*, ACS'2002 9th International Conference, Kluwer Academic Publishers, str. 41-51.

[7] I-Cheng Yeh, Modeling of strength of high performance concrete using artificial neural networks, *Cement and Concrete Research*, Vol. 28, No. 12, pp. 1797-1808 (1998).

[8] A. Tsanas, A. Xifara, Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools, *Energy and Buildings*, Vol. 49, pp. 560-567, 2012.

Seznam příloh

Příloha 1. CD s naprogramovaným algoritmem AnYa + elektronické verze DP a grafy.