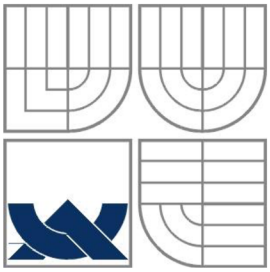
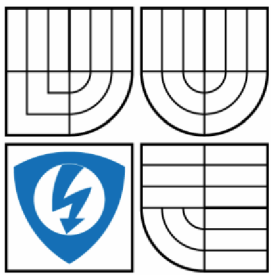


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

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
AUTHOR

Bc. Ondřej Klapil

VEDOUCÍ PRÁCE
SUPERVISOR

prof. Ing. Pavel Jura, CSc.

BRNO 2016

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, datová 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, data clouds, antecedent

Bibliografická citace:

KLAPIL, O. *Fuzzy systémy s netradičními antecedenty fuzzy pravidel*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2016. 58s. 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: **16. května 2016**

……
podpis autora

Poděkování

Děkuji vedoucímu diplomové práce Prof. Ing. Pavlu Jurovi, CSc. a Ing. Petru Honzíkovi, Ph.D. za pedagogickou a odbornou pomoc a další cenné rady při zpracování mé diplomové práce.

V Brně dne: **16. května 2016** ……

podpis autora

OBSAH

1. 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 v modulu fuzzifikace, kdy jsou data mapována na funkci příslušnosti, která nereprezentuje skutečné rozložení dat. Zmíněný nedostatek se pokouší vyřešit nový fuzzy systém typu AnYa. Práce si klade za cíl ověřit funkčnost tohoto nového systému.

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

V závěrečné části je vytvořený systém testován pro různé typy úloh a tím se ověřuje jeho funkce. Ve vybraných případech bude vyhodnocen vliv případného šumu v datech na fuzzy systém AnYa.

2. 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.

2.1. Fuzzy množiny

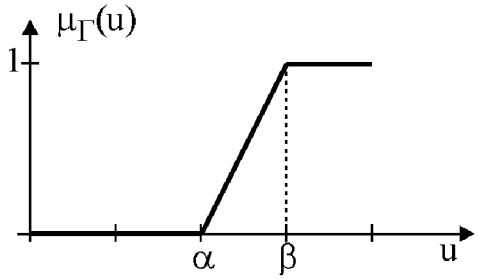
Na rozdíl od klasických množin, kdy libovolná klasická množina A definována v univerzu K a každý prvek $u \in K$, má pouze dva stavy – prvek u náleží množině A nebo prvek u nenáleží množině A. Ve fuzzy množinách může prvek univerza u náležet fuzzy množině F pouze částečně.

Matematicky lze zapsat jako:

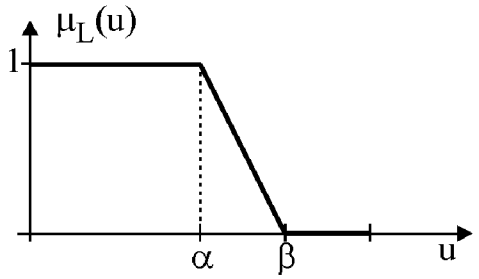
$$Klasická\ množina\ (charakteristická\ funkce): \mu_A: K \rightarrow \{0,1\}, \mu_A(u) \in \{0,1\} \quad (2.1)$$

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

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, volí se obvykle snadno popsatelné tvary příslušnosti. Mezi nejčastěji používané po částech lineární funkce patří *Γ*-funkce , L-funkce, *Λ*-funkce a *Π*-funkce:

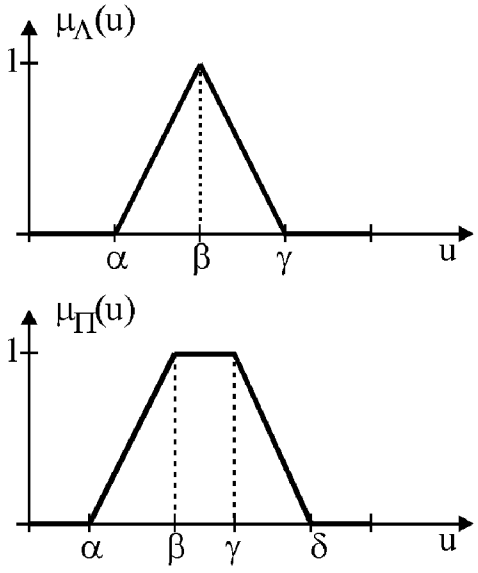


$$\Gamma(u,\alpha,\beta) = \begin{cases} 0 & u < \alpha \\ (u-\alpha)/(\beta-\alpha) & \alpha \le u \le \beta \\ 1 & u > \beta \end{cases}$$



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

Obr. 1: Vybrané funkce příslušnosti 1 [3]

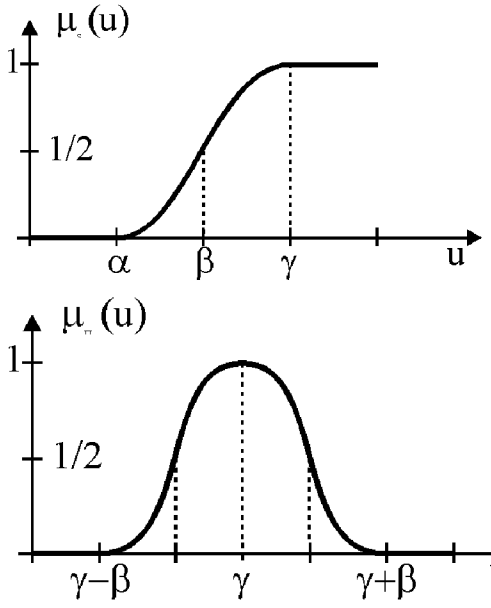


$$\Lambda(u,\alpha,\beta,\gamma)=\begin{cases}0 & u<\alpha\\ (u-\alpha)/(\beta-\alpha) & \alpha\le u\le\beta\\ (\gamma-u)/(\gamma-\beta) & \beta\le u\le\gamma\\ 0 & u>\gamma\end{cases}$$

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

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:

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

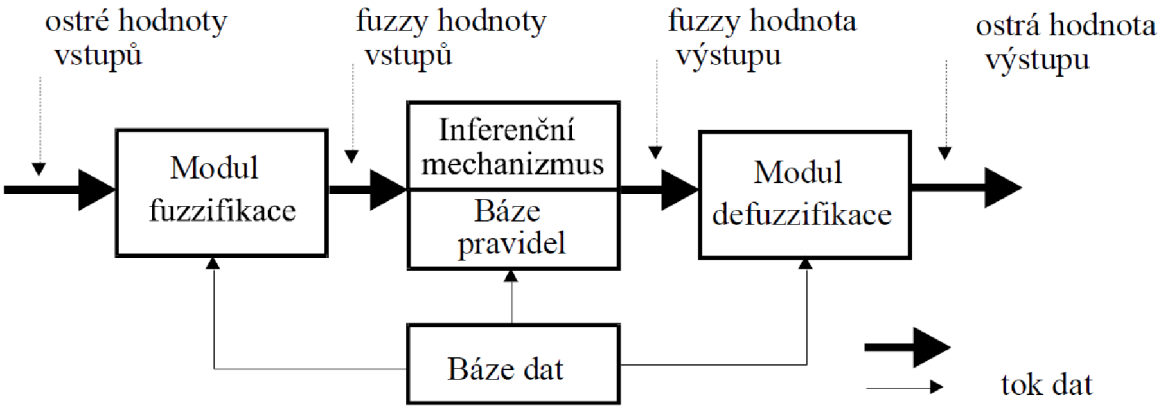
Opět lze použít pro průnik různou t-normu. Vznikají tak různé typy kompozice např. max-min kompozice nebo max-dot kompozice.

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

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

2.2. 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.

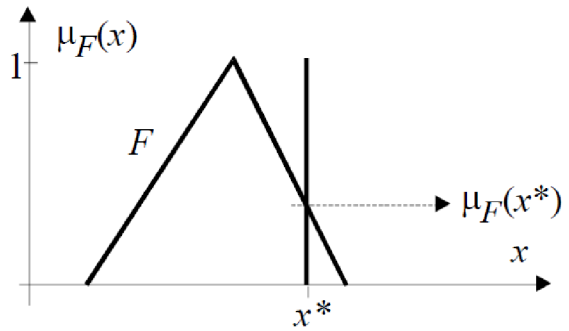


Obr. 4: Model fuzzy systému [3]

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):

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

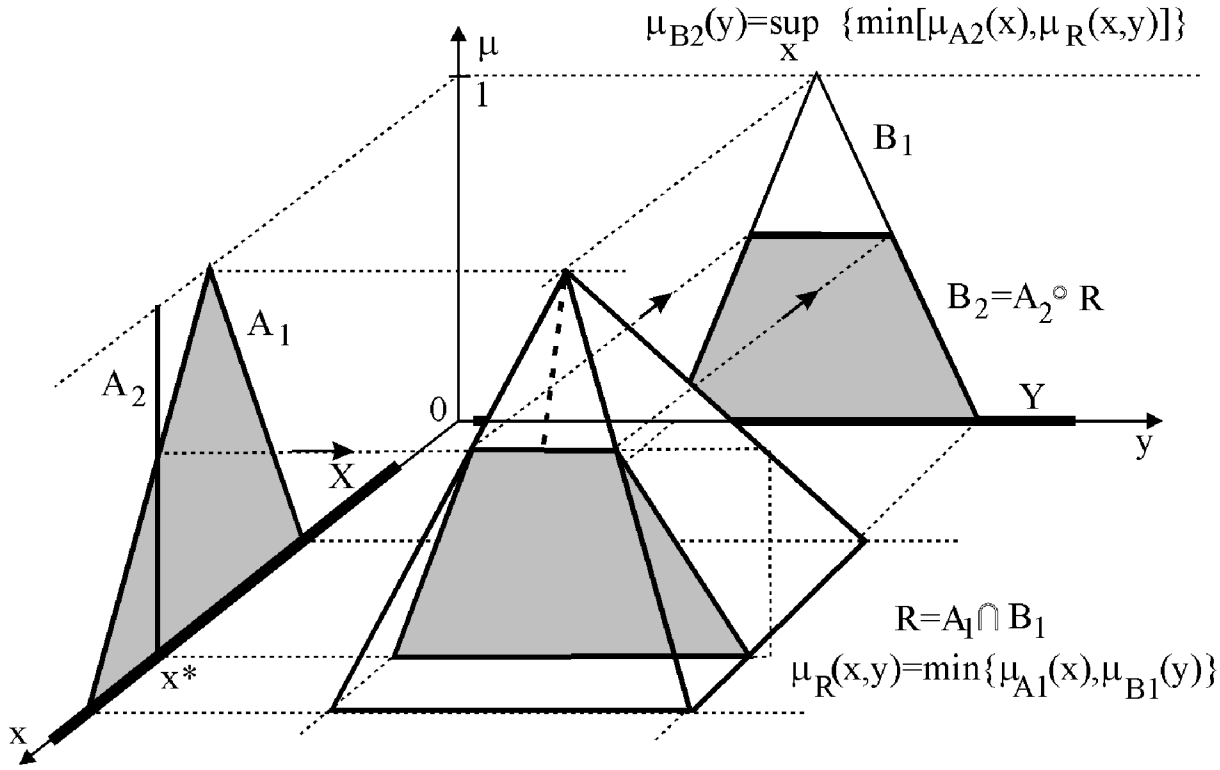
$$v_1, v_2 \ldots jednotlivé\ výroky$$

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 $\in$ A $\times$ 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:

$podmínka\ 1$:	$a\ je\ F_2$
$podmínka\ 2$:	$IF(a\ je\ F_1)THEN(b\ je\ V_1)$
$vyhodnocení$:	$b\ je\ V_2$

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.

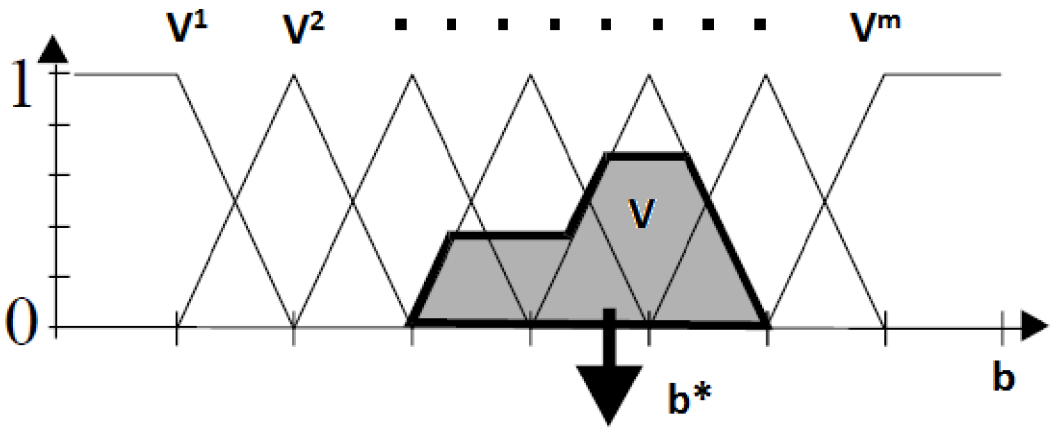
Modul defuzzifikace převádí výstupní fuzzy hodnoty inferenčního mechanizmu na ostré výstupní hodnoty. Máme-li proměnnou $b \in B$ a univerzum B je pokryto systémem fuzzy množin $V^i, i = 1,2 \ldots m, V^i \in B$, a máme výsledek inferenčního mechanismu, kdy jsou jednotlivým fuzzy množinám přiřazeny výsledné 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:

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

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.

2.2.1. Takagi-Sugeno

Fuzzy systém Takagi-Sugeno má ve vyhodnocení pravidla jako závěr funkcí 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:

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

2.2.2. 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).

2.2.3. AnYa

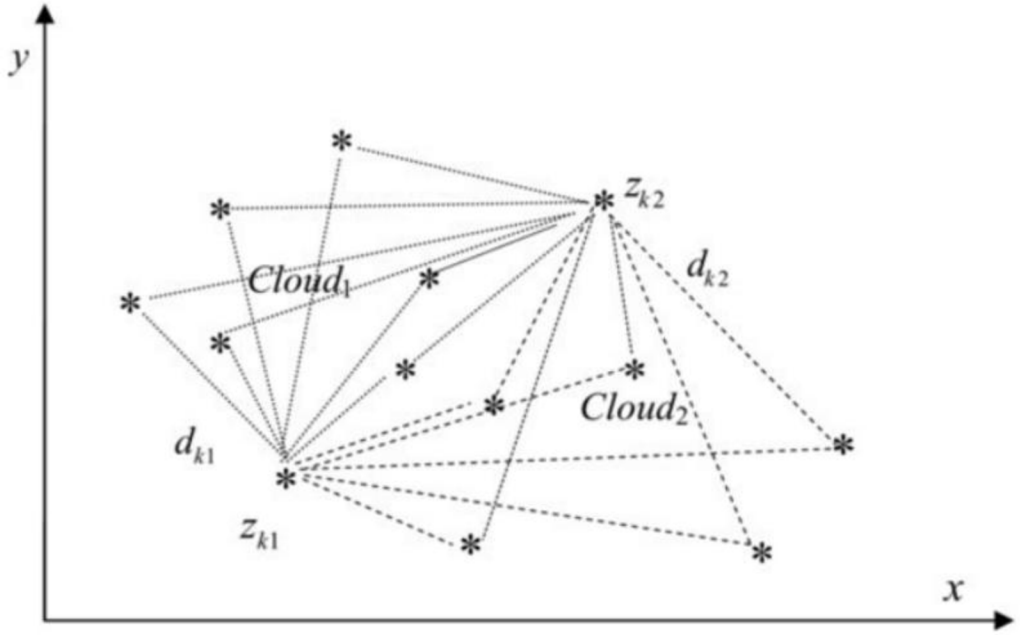
Nově navrhovaný systém typu AnYa (Angelov-Yager) se výrazně liší od nejpoužívanějších systémů typu Mamdani nebo Takagi-Sugeno. Tyto systémy, které jsou popsány výše, používají pro svou činnost nějakou formu fuzzyfikace, která vyžaduje apriorní znalost vstupních dat. Navíc při fuzzifikaci téměř vždy vznikne 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. Tento přístup k zpracovávání dat umožňuje nahradit parametrické funkce příslušnosti neparametrickými funkcemi lokální hustoty dat pro jednotlivá Mračna, což připomíná ostatní přístupy využívající jádrové funkce, jako například *support vector machines*.

Momentálně je systém AnYa, který vychází z jednoduchého fuzzy systému využívajícím vektorovou příslušnost a granulaci na bázi jádrových funkci [2] jediný fuzzy systém využívající sdružování dat do neparametrických datových Mračen. Díky této vlastnosti je možné používat systém AnYa na velikou škálu úloh – například modelování, klasifikace, zpracování obrazu (segmentace), identifikace systémů atp.

Tvorba datových Mračen a samotný systém AnYa je popsán v následující kapitole.

3. FUZZY SYSTÉM ANYA [1][2][3]

Mějme komplexní, nelineární, nedeterministický systém, který můžeme popsat pouze vztahy vstup-výstup, tedy vektor vstupů **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 8.: 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 1). 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

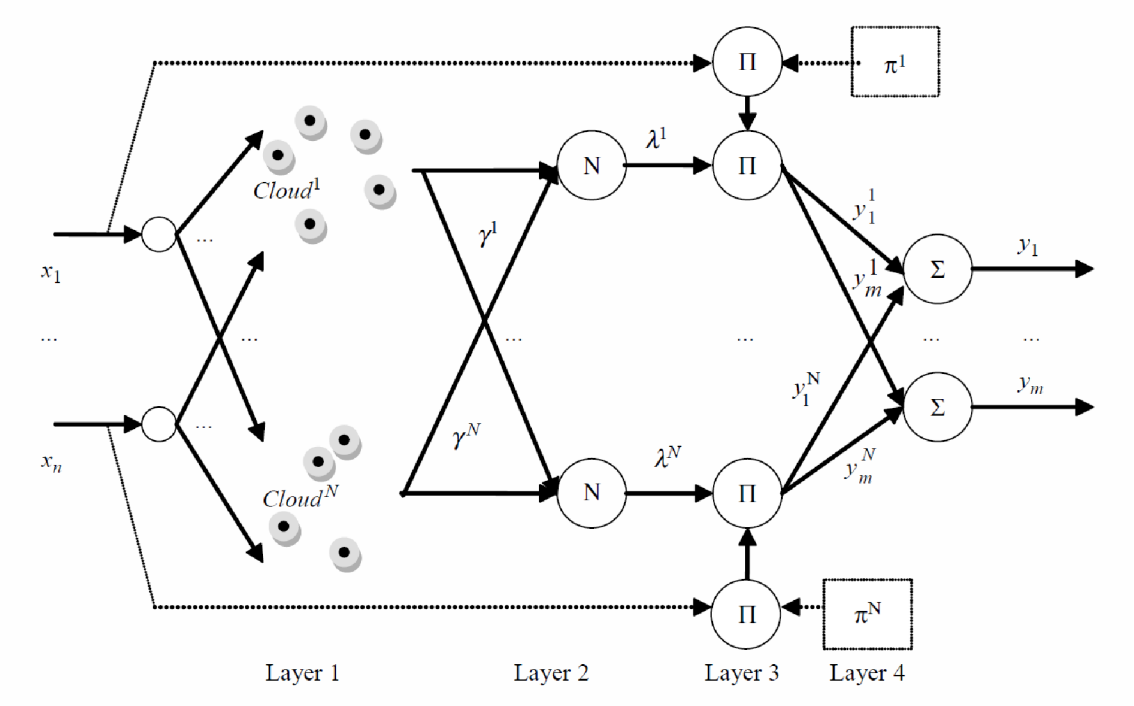
$$\pi^i = \begin{bmatrix} a_{01}^i & a_{02}^i & \dots & a_{0m}^i \\ a_{11}^i & a_{12}^i & \cdots & a_{1m}^i \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1}^i & a_{n2}^i & \cdots & a_{nm}^i \end{bmatrix}$$

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 9: Fuzzy systém AnYa jako neuronová síť [3]

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.

3.1. 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 (${z_k}^i = [x_k, y^i]$).

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

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 č. 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:

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

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:

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

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

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

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):

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

3.2. Tvorba závěrů (consequents)

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

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

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

$$Třída = arg \max_{i=1\ až\ N}(\lambda_i) \tag{3.20}$$

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

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

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 $Nn \times Nn$.

4. 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 jsou popsány jednotlivé varianty algoritmu pro různé úlohy. Mezi testované úlohy patří aproximace/predikce funkce a klasifikace.

4.1. 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 následující kroky:

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

4.2. Datová struktura

V realizaci systému AnYa můžeme identifikovat dvě hlavní datové struktury – hlavní datovou strukturu a pomocnou datovou strukturu.

4.2.1. Hlavní datová struktura

Hlavní datová struktura je nezbytná pro funkci algoritmu. Jedná se o vlastní řešení ukládání hodnot v průběhu samotného algoritmu.

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.

- Proměnná „clouds.cnt“ reprezentuje počet vytvořených Mračen.
- Proměnná „clouds.datacnt“ je počet dat, použitých k vytvoření stávající datové struktury „clouds“.
- „clouds.atcnt“ a „clouds.ycnt“ reprezentují informace o vstupních datech. Proměnná „atcnt“ je počet vstupních proměnných a proměnná „ycnt“ je počet výstupních proměnných jednotlivých dávek vstupních dat.
- „clouds.omega“ a „clouds.eps“ udržují informaci o nastavení algoritmu, a sice velikost parametru ε_1 reprezentujícím spodní hranici užitečnosti Mračen (rovnice (3.19)) a velikost parametru Ω používaného při tvorbě závěrů.
- „clouds.atcnt_true“ je pomocná proměnná, použitá pouze při experimentech pro aproximaci/modelování. Jedná se o skutečný počet vstupů v datovém souboru. Rozdíl oproti „clouds.atcnt“ bude pouze v případě, že data byla např. rozšířena o kombinace vstupních proměnných či jejich mocnin (tato možnost je naprogramována ve funkci pro preprocessing dat).

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

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

Tyto struktury jsou detailněji popsány dále.

Strukturovaná proměnná „clouds.global“

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

- „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.

Strukturovaná proměnná „clouds.RMS“

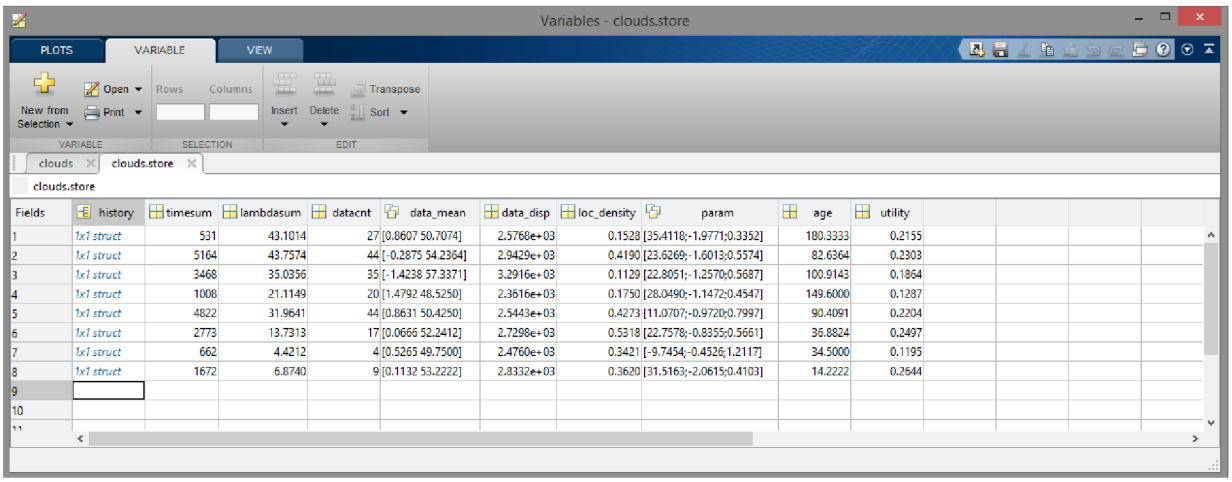
Strukturovaná proměnná „clouds.RMS“ sdružuje informace o závěrech (konsekventech) fuzzy systému AnYa pro stávající Mračna.

- Matice „clouds.RMS.C“ reprezentuje matici C

- Matice „clouds.RMS.phi obsahuje vektor jednotlivých parametrů Mračen θ^T.

Strukturovaná 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.



Fields	history	timesum	lambdasum	datacnt	data_mean	data_disp	loc_density	param	age	utility
1	1x1 struct	531	43.1014	27	[0.8607 50.7074]	2.5768e+03	0.1528	[35.4118;-1.9771;0.3352]	180.3333	0.2155
2	1x1 struct	5164	43.7574	44	[-0.2875 54.2364]	2.9429e+03	0.4190	[23.6269;-1.6013;0.5574]	82.6364	0.2303
3	1x1 struct	3468	35.0356	35	[-1.4238 57.3371]	3.2916e+03	0.1129	[22.8051;-1.2570;0.5687]	100.9143	0.1864
4	1x1 struct	1008	21.1149	20	[1.4792 48.5250]	2.3616e+03	0.1750	[28.0490;-1.1472;0.4547]	149.6000	0.1287
5	1x1 struct	4822	31.9641	44	[0.8631 50.4250]	2.5443e+03	0.4273	[11.0707;-0.9720;0.7997]	90.4091	0.2204
6	1x1 struct	2773	13.7313	17	[0.0666 52.2412]	2.7298e+03	0.5318	[22.7578;-0.8355;0.5661]	36.8824	0.2497
7	1x1 struct	662	4.4212	4	[0.5265 49.7500]	2.4760e+03	0.3421	[-9.7454;-0.4526;1.2117]	34.5000	0.1195
8	1x1 struct	1672	6.8740	9	[0.1132 53.2222]	2.8332e+03	0.3620	[31.5163;-2.0615;0.4103]	14.2222	0.2644
9										
10										

Obr. 10: 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. Ukládání historie není důležité pro běh algoritmu.

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 „clouds.store“ je pro chod algoritmu zapotřebí velmi malé množství pomocných proměnných a je zajištěna přehledná kontrola jeho funkčnosti.

Strukturovaná proměnná „clouds.trash“

Datová struktura „clouds.trash“ uchovává data spojená s údržbou počtu aktivních Mračen.

- Obsahuje data smazaných Mračen „clouds.trash.backup“ ve formě jejich struktury „clouds.store(i)“.
- Vektor „clouds.trash.gc_vect“ v naplněném stavu obsahuje indexy Mračen určených k odstranění z aktuální datové struktury. Je nezbytný pro funkci *g_colector*.

4.2.2. Pomocná datová struktura

Pomocná datová struktura sdružuje vstupní data a výstupní data, nastavení experimentů či metriky pro vyhodnocení činnosti algoritmu. Součástí jsou všechna data potřebná pro zopakování jednotlivých experimentů a zobrazení průběhu činnosti systému při jejich realizaci.

Datová struktura „history“

Jedná se o hlavní datovou strukturu obsahující veškeré potřebné informace pro vyhodnocení experimentu a jeho reprodukovatelnost. V průběhu činnosti programu jsou zde ukládána vstupní i výstupní data systému AnYa a jeho nastavení. Ačkoliv se pro různé experimenty datová struktura liší jak v počtu nebo typu ukládaných dat, lze pro všechny experimenty identifikovat několik základních okruhů ukládaných dat:

- Struktura sdružující všechny použité data „history.data“. Na jednom místě jsou shromážděny veškeré soubory tréninkových a validačních dat, předzpracované i bez předzpracování. Mohou být dále strukturovány, aby bylo možné odlišit např. jednotlivé iterace Křížové validace
- Struktura „history.settings“ ukládající nastavení systému AnYa pro každé spuštění hlavní funkce.

- Struktura „history.clouds“ sdružující všechny vytvořené datové struktury „clouds“ v průběhu experimentu.
- Samotné výsledky experimentů jsou ukládány do struktury „history.performance“. Zde je možné nalézt všechna data použitá pro vyhodnocení kvality systému AnYa, jeho výstupy.

Mezi specifické struktury vytvořené různými experimenty patří:

- „history.final“ – tato struktura ukládá průměrné hodnoty při vícenásobném opakování, například metodou křížové validace. Vyskytuje se u klasifikačních experimentů z Kapitoly 5.
- „history.comparsion“ - v případě experimentu popsaném v kapitole 5.1.1 se zde ukládají výstupy metody k-NN použité pro srovnání

Datová struktura „inp_data“

Spouštěcí skript programu využívá strukturu „inp_data“ jako ukládací prostor vstupních souborů dat pro jednotlivé experimenty. Ve struktuře se ukládají původní data, data rozdělená na tréninkové a validační soubory, normalizovaná či rozšířená data. Jednotlivé instance jsou poté předkládány hlavní funkci *mainfcn*.

Datová struktura „settings“

Struktura „settings“ obsahuje nastavení systém AnYa pro zajištění činnosti hlavní funkce *mainfcn*. Sdružuje informace o počtu vstupů, počtu výstupů datového souboru, předává informace o nastavení parametru ε_1 a Ω. Dále obsahuje specifická data potřebná pro činnost variant funkce *mainfcn*, jako například rozhodovací proměnnou, zdali je použita metoda „vítěz bere vše“ apod.

4.3. Popis programu

V této podkapitole bude vysvětlena funkce algoritmu. Budou představeny varianty algoritmu pro jednotlivé úlohy. Nachází se zde i popis podpůrné část algoritmu, starající se o přípravu dat, vyhodnocení výstupu a ukládání historie.

4.3.1. Vlastní algoritmus AnYa

Systém AnYa byl pro účely této práce naprogramován ve formě funkce. Tato je volána pomocí spouštěcího skriptu, ve kterém se systému postupně dodávají data. V této kapitole bude popsána pouze samotná realizace systému AnYa podle kapitoly 3.

Hlavní funkce

Systém typu AnYa je realizován pomocí funkce *mainfcn*, respektive jejích variant pro různé typy experimentů. Pro klasifikaci je to *classification_mainfcn* a obdobně pro aproximaci *aproximation_mainfcn*. Hlavní rozdíly mezi oběma funkcemi jsou v tvorbě závěrů a vracení rozličných dat pro potřeby vyhodnocení průběhu experimentu a archivaci jeho průběhu.

Funkce jsou připraveny pro offline režim, avšak je zde přítomna část kódu realizující online variantu algoritmu. Přesunutím inicializace mimo hlavní funkci (neinicializovat novou strukturu mračna při každém volání funkce), odstraněním tréninkového for cyklu, odstraněním validační části a přidáním výstupu reprezentujícím online výstup algoritmu lze docílit hlavní funkce plně připravené pro online režim.

Vstupy funkce

Pro experimenty zaměřené na aproximační úlohy jsou vstupy hlavní funkce:

- Strukturovaná proměnná „settings“
- Soubor neupravených validačních dat (pro vyhodnocení přesnosti aproximace) „val_true“
- Soubor tréninkových dat „tre_u“, které mohou být předzpracovány
- Soubor validačních dat „val_u“

Při klasifikační verzi Hlavní funkce jsou výstupy:

- Strukturovaná proměnná „inp_data“ sdružující tréninkové a validační data
- Strukturovaná proměnná „settings“

Výstupy funkce

Hlavní funkce určená pro aproximaci má výstupy:

- Konečnou podobu struktury Mračen „out_clouds“ za účelem vyhodnocení funkčnosti a archivaci do struktury „history“
- Matice hodnot „output“ reprezentující aproximovaný výstup systému AnYa
- Hodnotu střední kvadratické chyby „rmse“
- Hodnotu indexu NDEI (5.3) používanou pro vyhodnocení kvality aproximace „ndei“
- Index determinace „R2“

Výstupy klasifikační verze Hlavní funkce:

- Hodnota přesnosti klasifikace „acc“
- Počet vytvořených Mračen „cldcnt“
- Čtyři hodnoty „fp“, „fn“, „tp“ a „tn“ reprezentující počet falešně pozitivních, falešně negativních, skutečně pozitivních a skutečně negativních výsledků klasifikace vypočítaných funkcí, operuje-li klasifikátor s daty, jejichž výstupem je bipolární hodnota (např. ano/ne)
- Konečná podoba struktury Mračen „clouds“
- Matice výstupů ze systému „val_y“ klasifikovaných systémem na validačních datech

Popis funkce

Zpočátku je inicializována datové struktura na základě vstupních dat a nastaveni „settings“. Poté je realizována tréninková část algoritmu ve smyčce. Algoritmus je opakován tolikrát, kolik je vzorků ve struktuře tréninkových dat.

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*. 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 vypočítá 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, jak naložit 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 a na základě spodní hranice užitečnosti Mračen se označí Mračna určená ke smazání (podle rovnice (3.19)). Dále je realizován výpočet závěrů pomocí funkce *wRMSfcn*. Po výpočtu závěrů je v kódu přítomen výpočet online výstupu systému AnYa. V experimentech se nepoužívá. Poslední část tréninkové smyčky spouští funkci *g_collector*, která realizuje mazání označených neužitečných Mračen. Tím byl realizován celý algoritmus popsaný v kapitole 3.1 a končí tréninková smyčka. Systém by měl být naučen na tréninkový datový soubor.

Následující blok realizuje validační cyklus. V cyklu jsou systému předkládána validační data. Určí se, s jakou mírou data náleží jednotlivým Mračnům a vypočte se vektor normalizovaných hodnot lokální hustoty λ_k^i a aproximuje se výstup. Jeho hodnota se uloží do vektoru výstupů. Provádí se zde i výpočet pomocných hodnot určených pro vyhodnocení činnosti algoritmu.

Po validačním cyklu je blok pomocných výpočtů ukončením funkce a navrácením výstupních dat.

Změna aproximační verze oproti klasifikační je například v přidání úpravy vektoru normalizovaných hodnot lokální hustoty λ_k^i v případě klasifikace, kde je možnost použít metodu „vítěz bere vše“.

Funkce pro vytvoření nového Mračna

Hlavní část algoritmu volá funkci *cloudgen* při splnění podmínky pro tvorbu nových Mračen. Funkce obstará náležitosti vytvoření nového Mračna a začlení ho do datové struktury.

Vstupy funkce

- Hlavní datová struktura „clouds“
- Vzorek vstupních dat „data“
- Index vstupního vzorku dat „time“

Výstupy funkce

- Upravená datová struktura „clouds“

Popis funkce

Funkce *cloudgen* 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. Na závěr probíhá inicializace parametrů závěrů nového mračna. Zde je rozdíl mezi verzí pro klasifikaci a verzí pro aproximaci.

Pro úlohu aproximace pomocí lineární kombinace vstupů jsou parametry závěrů inicializovány jako matice o *x=clouds.atcnt+1* sloupcích a *y=clouds.ycnt* řádcích. Matice reprezentuje o jedničku rozšířený vektor vstupních dat pro každý výstup.

Při řešení klasifikační úlohy je výstup typu *singleton* či pevná hodnota *enum*. Parametr je proto v tomhle případě inicializována jako matice o rozměrech $1\times clouds.ycnt$.

Funkce na aktualizaci vítězného Mračna

Funkce *cloudgup* provádí aktualizaci vítězného Mračna o hodnoty aktuálního vstupu při splnění podmínky aktualizace stávajícího Mračna.

Vstupy funkce

- Hlavní datová struktura „clouds“
- Index vítězného Mračna „cloud_num“ které bude aktualizováno
- Vzorek vstupních dat „data“
- Index vstupního vzorku dat „time“

Výstupy funkce

- Upravená datová struktura „clouds“

Popis funkce

Funkce ukládá funkce pro aktualizaci vítězného 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 s použitím aktuálních vstupních dat. Tím se docílí aktualizace žádaného Mračna. Funkce je univerzální pro všechny testované úlohy algoritmu.

Funkce na výpočet globální hustoty

Funkce *globalfcn* se stará o výpočet globální hustoty datové struktury fuzzy systémů AnYa a její uložení do datové struktury.

Vstupy funkce

- Datová struktura „global“ reprezentující „clouds.global“
- Vzorek vstupních dat „data“
- Index vstupního vzorku dat „time“

Výstupy funkce

- Upravená datová struktura „global“ reprezentující „clouds.global“

Popis funkce

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“.

Vstupy funkce

- Hlavní datová struktura „clouds“
- Vzorek vstupních dat „data“

Výstupy funkce

- Vektor lokálních hustot „gamma“ reprezentující lokální hustoty všech Mračen

Popis funkce

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

Funkce *globalfcntmp* je odvozena od funkce pro výpočet globální hustoty *globalfcn*. Rozdíl je, že tato funkce neupravuje datovou strukturu „clouds.global“, ale pouze počítá s jejími daty. Na jejich bázi vytvoří vektor přechodných globálních hustot Γ_k^i pro řešení rovnice (3.13).

Vstupy funkce

- Datová struktura „global“ reprezentující „clouds.global“
- Vzorek vstupních dat „data“
- Hlavní datová struktura „clouds“

Výstupy funkce

- Vektor globálních hustot „“ reprezentující globální hustoty Γ_k^i pro všechna Mračna

Popis funkce

První se rozšíří vstupní data o na jejich základě predikovanou hodnotu k-tým Mračnem. Zde se liší funkce pro aproximaci a klasifikaci, jelikož počítají výstup systému jiným způsobem. Poté se spočítá globální hustota pro vzorek dat s výstupem aproximovaným každým Mračnem Γ_k^i podle rovnice (3.8) na základě globální střední hodnoty (3.9), globálního rozptylu dat (3.10). Toto se provede pro všechna Mračna v datové struktuře.

Funkce pro údržbu Mračen

Hlavní část algoritmu starající se o udržování kvality datové struktury Mračen. *uti_age_up* se stará o výpočet jejich užitečnosti a stáří a rozhodne, zdali je potřeba smazat některé z Mračen.

Vstupy funkce

- Hlavní datová struktura „clouds“
- Index vstupního vzorku dat „time“
- Vektor „lambda“ reprezentující normalizované hodnoty lokální hustoty λ_k^i pro všechna Mračna

Výstupy funkce

- Upravená datová struktura „clouds“
- Případně upravený vektor „lambda“

Popis funkce

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. Současně s tím označí Mračno nesplňující podmínku (3.19) pro smazání zápisem jeho indexu do proměnné „gc_vect“. 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*.

Vstupy funkce

- Hlavní datová struktura „clouds“
- Vzorek vstupních dat „data“
- Vektor „lambda“ reprezentující normalizované hodnoty lokální hustoty λ_k^i pro všechna Mračna
- Hodnota parametru Ω využívána v rovnici (3.24)

Výstupy funkce

- Upravená datová struktura „clouds“

Popis funkce

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). Zde se liší klasifikační a aproximační varianta. U klasifikace je jako vstupní proměnná používán pouze vektor normovaných lokálních hustot λ_k^i jako ekvivalent hodnoty funkce příslušnosti vzorku dat jednotlivým Mračnům. Pro aproximaci pomocí lineární kombinace vstupů využíváme fuzzy vážený (pomocí λ_k^i) o jedničku rozšířený vektor vstupních dat. Nakonec upraví parametry výstupů jednotlivých Mračen v datové struktuře „clouds“.

Funkce pro odstranění neužitečných Mračen

Na základě vektoru indexů „gc_vect“ smaže funkce *g_collector* požadované Mračna. V průběhu aktualizuje i ostatní data algoritmu, aby byly konzistentní s provedenou akcí.

Vstupy funkce

- Hlavní datová struktura „clouds“

Výstupy funkce

- Upravená datová struktura „clouds“

Popis funkce

Po načtení vektoru indexů Mračen určených pro smazání „clouds.trash.gc_vect“ provede výpočet indexů matic $\hat{\theta}_k$ a C_k týkajících se Mračen určených ke smazání a vytvoří detailní mazací vektor. Poté začne mazat nepotřebná mračna ze struktury „clouds.store“. Jejich kopie ukládá pro pozdější analýzu do struktury „clouds.trash.backup“. Nakonec podle detailního vektoru upraví matice $\hat{\theta}_k$ a C_k, čímž dokončí operaci smazání Mračen z datové struktury.

4.3.2. Pomocné funkce a spouštěcí skript

V předchozí části byla popsána vlastní implementace systému AnYa. Aby systém fungoval, musí se mu předat nastavení parametrů a vlastní data. O to se stará spouštěcí skript, který volá pomocnou funkci na generování dat. Výstup ze systému je poté zpracován a vyhodnocen.

Funkce na generování vstupních dat a nastavení

Pomocná funkce *dataselector* generuje balíky vstupních dat na míru experimentu.

Vstupy funkce

- Rozhodovací *integer* „decision“, který určuje experiment.
- Nedefinovaná proměnná „swi“, která v některých případech předává úroveň šumu či metodu tvorby datových souborů.

Pro klasifikační verzi je přítomno více vstupních proměnných:

- „repeat“ – integer reprezentující počet vytvořených datovýchsouborů
- Datová struktury „inp_history“ jež předává strukturu „inp_data“ pro vygenerování šumu.
- Datová struktura „set_history“ pro aktualizaci struktury „settings“ při vytváření datových souborů obsahujících šum

Výstupy funkce

- Datová struktura „inp_data“.

- Datová struktura „settings“.

Popis funkce

Na základě vstupní proměnné „decision“ funkce vygeneruje data pro žádaný experiment. Data jsou uloženy do strukturované proměnné „inp_data“. Tato proměnná obsahuje celý datový soubor, i jeho části určené pro trénování a validaci algoritmu. Struktura proměnné se může měnit na základě potřeby experimentu. V některých případech je vygenerováno i větší množství tréninkových a validačních dat pro více spuštění hlavní funkce – závisí na návrhu experimentu. Dále funkce generuje strukturovanou proměnnou „settings“, která sdružuje všechna potřebná data pro nastavení algoritmu AnYa. Existují dvě rozdílné verze pro klasifikaci a aproximaci. Klasifikační verze je schopna vygenerovat datové soubory používané v Kapitole 5.1. Aproximační verze pokrývá generaci dat pro experimenty v Kapitole 5.2.

Funkce na rozšiřování vstupního datového prostoru

Funkce *DataUp* umožňuje rozšíření vektoru vstupních dat pro např. aproximační experimenty. Např. vektor dat (x_n vstupy, y výstup) $A = [x_1, x_2, y_1]$ rozšíří na $A_r = [{x_1}^2, x_1, x_1x_2, {x_2}^2, x_2, y_1]$, při volání funkce *DataUp(A, 1, 2, true, 2).*

Vstupy funkce

- Vstupní data určená k rozšíření „data1“
- Počet výstupů v datech „ycnt“
- Maximální mocnina rozšířených dat „power“
- Boolean pro povolení kombinací vstupů „comb“
- Maximální délka kombinací „maxcomb“

Výstupy funkce

- Rozšířená data „newData“

Popis funkce

V první části funkce rozšíří vstupní data ve vzorku o jejich mocniny na základě maximální hodnoty „power“. Poté, jeli tak žádáno rozšíří vstupní data vzorku o různé kombinace vstupů.

Funkce realizující metodu k-NN

Pro potřeby experimentu v Kapitole 5.1.1 bylo potřeba realizovat srovnávací klasifikační metodu k-NN. Proto byla vytvořena funkce *bank_knn_fnc*.

Vstupy funkce

- Soubor validačních a tréninkových dat „data“
- Parametr metody „k“

Výstupy funkce

- Přesnost klasifikace „acc“
- Čtyři hodnoty „fp“, „fn“, „tp“ a „tn“ reprezentující počet falešně pozitivních, falešně negativních, skutečně pozitivních a skutečně negativních výsledků klasifikace.

Popis funkce

Funkce využívá metody k-NN implementované v programu MATLAB. Pomocí souboru tréninkových dat vygeneruje model k-NN a poté provede validaci na validačním souboru dat. Výsledky validace vrátí.

Skript na spuštění experimentu

Všechny výše popsané funkce se sdružuje spouštěcí skript *Startup*. Jeho úkolem je na základě volby uživatele vygenerovat data pro zvolený experiment, provést experiment a zaznamenat data z průběhu experimentu. Volba uživatele probíhá změnou proměnných v inicializační části skriptu. Existují dvě varianty spouštěcího skriptu. Pro úlohu klasifikace je to *Startup_Class*, obdobně pro aproximaci *Startup_Approx*. Po provedení skriptu zůstane uložena strukturovaná proměnná „history“, kde lze najít veškerá dat k vyhodnocení experimentu.

Verze pro aproximaci

Verze skriptu pro aproximaci realizuje dva experimenty z Kapitoly 5.2. Příklad nastavení experimentu (Spustí aproximaci koncentrace CO_2 v laboratorní peci a použije k tomu uložená data z diplomové práce):

```
clear all;
close all;

%% Halvní nastavení experimentu
%   decision:   1 - Aproximace energetické účinnosti budovy
%               2 - Aproximace koncentrace CO2 v laboratorní peci
%   olddata:    true - pro použití dat z diplomové práce
%               false - pro vygenerování nových dat
%   noisetre:   true - použít šum v datech (pouze pro decision=2)
%               false - nepoužít šum v datech (pouze pro decision=2
%   noisetre:   true - šum v datech aplikován na tréninkové data
%               false - šum v datech aplikován na validační data
%   level:      int úroveň šumu
%%%

decision=2;
olddata=true;
noise=false;
noisetre=true;
level=1;
```

Verze pro klasifikaci

Verze skriptu pro aproximaci realizuje tři experimenty z Kapitoly 5.1. Příklad nastavení experimentu (Proběhne experiment klasifikace bankovek, vygenerují se nové tréninkové a validační data):

```
clear all;
close all;

%% Halvní nastavení experimentu
%   decision:   1 - Klasifikace nemocí
%               2 - Klasifikace znalostí
%               3 - Klasifikace bankovek
%               4 - Klasifikace bankovek - data obsahující šum
%   swi:        0 - Použit oba výstupy pro klasifikaci nemocí
%               1 - Použit první vystup pro klasifikaci nemocí
%               2 - Použit druhý vystup pro klasifikaci nemocí
%   olddata:    true - pro použití dat z diplomové práce
%               false - pro vygenerování nových dat
%   noiseold:   true - použití dat z diplomové práce pro generování šumu na
%                      datech (pouze pro decision = 4)
%               false - použití nove vygenerovaných dat pro generování šumu
%                      na datech (pouze pro decision = 4)
%   level:      int úroveň šumu
%   knn:        int nastavení k pro metodu k-NN (decision = 4 a decision = 3)
%%%

decision = 3;
swi=0;
olddata=true;
noiseold=true;
level=50;
knn=5;
```

Využití uložených dat pro opětovné provedení experimentu

Z důvodů reprodukovatelnosti výsledků je skript schopen načíst uloženou datovou strukturu „history“ a zopakovat experiment. Struktura musí mít název „history_old“ a být uložena jako soubor s příponou *.mat*. V kódu poté umístit název souboru k příslušné funkci *load*. Načítání souboru probíhá vždy na začátku skriptu po zvolení typu experimentu nastavením proměnné „olddata“ na hodnotu *true*.

5. OVĚŘENÍ METODY

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

5.1. 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. Ověřena byla základní funkčnost algoritmu tak, jak byl napsán – pokročilejší manipulací s pravidly tvorby závěrů a lepším preprocessingem dat by jistě bylo možné dosáhnout lepších výsledků.

5.1.1. 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 *enum {pravá, falešná}*. Vstupy reprezentují vlastnosti fotky získané její vlnkovou transformací.

Pro srovnání úspěšnosti klasifikace systémem AnYa byla zvolena metoda k-NN s použitou Euklidovskou metrikou a velikostí k=1, k=3 a k=5. Tato metoda byla realizována pomocí implementované funkce v programu MATLAB.
Vzhledem k binární povaze výstupu byla pro úspěšnost klasifikace použita tyto kritéria:

- Přesnost $\frac{TP+TN}{TP+TN+FP+FN}$
- Citlivost $\frac{TP}{TP+FN}$
- Specificita $\frac{TN}{TN+FP}$
- Prediktivní hodnota pozitivního testu $\frac{TP}{TP+FP}$
- Prediktivní hodnota negativního testu $\frac{TN}{TN+FN}$

Kde:

TP - zjištěné případy výskytu znaku odpovídají skutečnosti

TN - zjištěné případy nepřítomnosti znaku odpovídají skutečnosti – správné zamítnutí

FP - odpovídá falešnému poplachu

FN - odpovídá přehlédnutému výskytu

Datový soubor byl náhodně rozdělen na dvě stejné poloviny (686 záznamů), kdy jedna byla použita pro naučení algoritmu a druhá pro jeho validaci. Po vyhodnocení se

tyto datové soubory prohodily – validační soubor byl použit pro učení a obráceně – metoda „holdout“. Toto se opakovalo celkem desetkrát. Hodnoty pro vyhodnocení úspěšnosti klasifikace obou algoritmů byly průměrné hodnoty výše zmíněných kritérií pro všech deset iterací.

Klasifikace systémem AnYa probíhala metodou „vítěz bere vše“ a výstupem byla nejčastěji zastoupená hodnota výstupů vítězného Mračna. Hranice užitečnosti Mračen $\varepsilon_1 = 0.01$ a parametr překrytí $\Lambda = e^{-1}$. Pro potřeby algoritmu byla výstupní hodnota kódována jako *integer {-1,1}*. Vstupní data byla normalizována metodou „z-score“ (realizováno implementovanou funkcí v programu MATLAB).

Systém AnYa klasifikoval s průměrnou úspěšností 94,72%. Ve srovnání se základní metodou k-NN, která klasifikovala s průměrnou přesností mezi 99,81% (k=1) a 99,83% (k=5) je zřejmě, že systém AnYa je pro daný datový soubor značně horší klasifikátor. Z hodnot v Tabulce vyplývá, že systém AnYa byl úspěšnější v negativní klasifikaci vzorku, kdežto k-NN v pozitivní klasifikaci vzorku. Metoda k-NN pro k=1 a k=3 dokonce určila všechny pozitivní vzorky ve sto procentech případů.

Algoritmus		k-NN (EU) k=1	k-NN (EU) k=3	k-NN (EU) k=5	AnYa
Přesnost	Průměr [%]	99,81	99,83	99,83	94,72
	Max. [%]	100,00	100,00	100,00	97,23
	Min. [%]	99,56	99,71	99,56	91,25
Specificita	Průměr [%]	100,00	100,00	99,97	94,26
	Max. [%]	100,00	100,00	100,00	98,44
	Min. [%]	100,00	100,00	99,73	88,38
Citlivost	Průměr [%]	99,58	99,62	99,66	95,45
	Max. [%]	100,00	100,00	100,00	98,59
	Min. [%]	98,99	99,32	98,99	88,70
Prediktivní hodnoty pozitivního testu	Průměr [%]	100,00	100,00	99,97	92,63
	Max. [%]	100,00	100,00	100,00	98,06
	Min. [%]	100,00	100,00	99,66	84,86
Prediktivní hodnota negativního testu	Průměr [%]	99,66	99,70	99,73	96,42
	Max. [%]	100,00	100,00	100,00	98,97
	Min. [%]	99,24	99,45	99,24	91,35

Tabulka č. 2: Zaznamenané hodnoty metrik použitých pro posouzení přesnosti klasifikace bankovek

V průběhu klasifikace systém AnYa vytvořil průměrně 22 Mračen. Užitečnost žádného Mračna neklesla pod hranici ε_1.

Součástí experimentu je i zhodnocení šumu v datech na klasifikaci. Byl zvolen šum hodnoty v třídovém atributu (výstup). Tento druh šumu je příklad nesprávného zařazení dat. Metoda šumu je uniformní šum hodnoty v třídovém atributu (Uniform class noise [9]). *x*% datových vzorků je poškozeno tím, že se vezme hodnota výstupu vybraného vzorku a změní se za náhodně vybranou nestejnou hodnotu z ostatních datových vzorků. Při experimentu se měnil výstup datového vzorku reprezentující pravost bankovky z *pravé* na *falešnou* a obráceně.

Na původní datový soubor se aplikuje výše zmíněný šum, čímž se vytvoří nový datový souboru obsahující šum s indexy datových záznamů korespondujících s původním datovým záznamem. Experiment poté probíhá tak, že se vybírají data z obou souborů. Podle [9] je obvyklé, při šumu hodnoty v třídovém atributu, že se poškozený datový soubor používá na výběr tréninkové množiny dat a původní datový soubor pro validační soubor dat. Toto je znázorněno na Obrázku 11[9].

Obrázek č. 11: Tvorba validačních a tréninkových souborů dat pro posouzení vlivu šumu na klasifikační algoritmus [9]

Pro posouzení vlivu šumu byly použity stejné datové soubory jako ve srovnání s metodou k-NN. Data obsahující šum se nacházela v tréninkovém datovém souboru. Metrika pro vyhodnocení vlivu šumu ELA (Equalized loss of accuracy):

$$ELA_{x\%}[\%] = \frac{100 - Acc_{x\%}}{Acc_{0\%}} * 100 \tag{5.1}$$

Tato metrika se používá k porovnání dvou klasifikátorů [9], v našem případě mezi systémem AnYa a metodou k-NN. Dále byla použita metrika RLA (Relative loss of accuracy – Relativní ztráta přesnosti), která lépe odráží vliv šumu na klasifikátor:

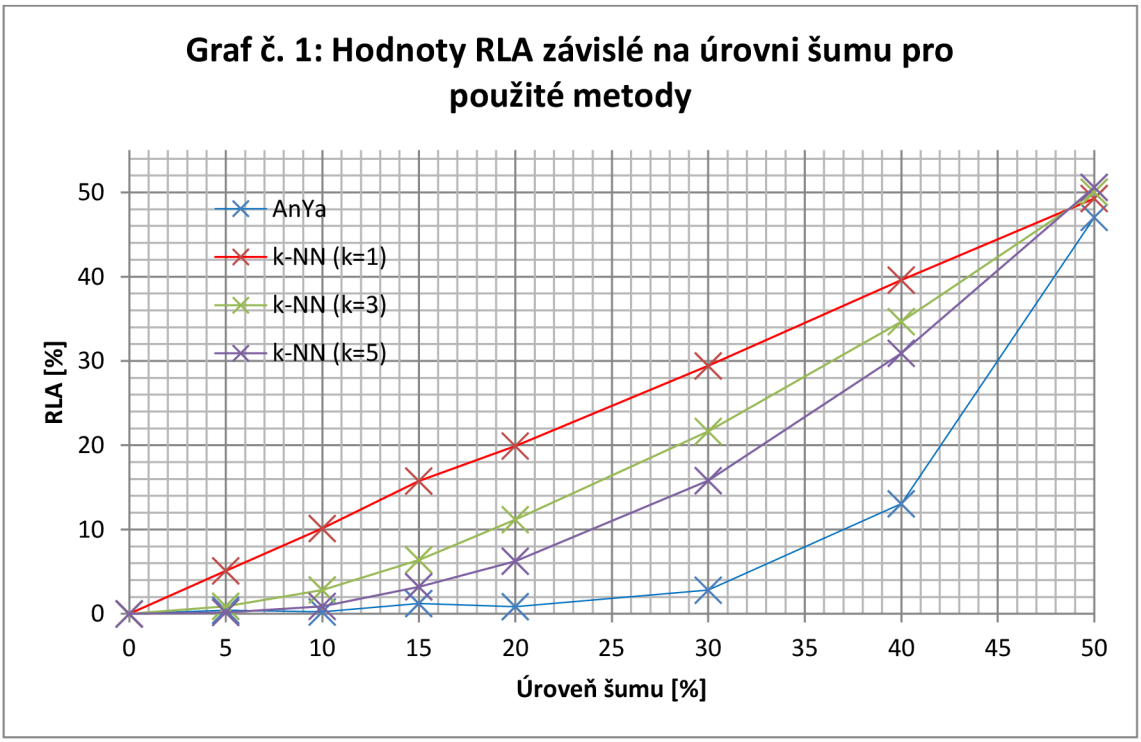
$$RLA_{x\%}[\%] = \frac{Acc_{0\%} - Acc_{x\%}}{Acc_{0\%}} * 100 \tag{5.2}$$

Tabulka č. 3 obsahuje data z experimentu zatíženého výše popsaným šumem. Úrovně použitého šumu činily 5%, 10%, 15%, 20%, 30%, 40% a 50%.

úroveň šumu [%]		0	5	10	15	20	30	40	50
AnYa	Acc [%]	94,72	94,32	94,51	93,56	93,91	92,06	82,35	50,16
	RLA [%]	0,00	0,42	0,22	1,22	0,86	2,81	13,06	47,04
	ELA [%]	5,57	6,00	5,80	6,80	6,43	8,38	18,63	52,62
k-NN (EU) k=1	Acc [%]	99,81	94,71	89,70	84,08	79,93	70,45	60,28	50,62
	RLA [%]	0,00	5,11	10,13	15,76	19,92	29,42	39,61	49,28
	ELA [%]	0,19	5,30	10,32	15,95	20,11	29,61	39,80	49,47
k-NN (EU) k=3	Acc [%]	99,83	98,93	97,00	93,44	88,67	78,22	65,22	49,88
	RLA [%]	0,00	0,90	2,83	6,40	11,18	21,65	34,67	50,04
	ELA [%]	0,17	1,07	3,01	6,57	11,35	21,82	34,84	50,21
k-NN (EU) k=5	Acc [%]	99,83	99,66	98,94	96,62	93,59	84,05	68,97	49,31
	RLA [%]	0,00	0,17	0,89	3,22	6,25	15,81	30,91	50,61
	ELA [%]	0,17	0,34	1,06	3,39	6,42	15,98	31,08	50,78

Tabulka č. 3: Srovnání vlivu šumu v tréninkových datech pro různé typy klasifikátorů

V Grafu č. 1 je názorně zobrazen vliv úrovně šumu v datech na přesnost klasifikace pro obě metody metrikou RLA.



Graf č. 1: Závislost Relativní ztráty přesnosti na úrovni šumu pro klasifikaci bankovek

Lze vidět, že systém AnYa se s daty obsahujícími šum vypořádal lépe, než metoda k-NN pro různé k. Do úrovně 30% šumu v datech systém AnYa klasifikoval velice přesně v porovnání s klasifikací bez šumu (do rozdílu v přesnosti 2,7%), kdežto přesnost klasifikace metody k-NN výrazně klesala. Tento výsledek je způsoben sdružováním

vstupních dat do Mračen, kdy jsou souvislosti mezi daty hledány s malým přispěním výstupní hodnoty (viz. Kapitola 3). Lze tedy říci, že systém AnYa je oproti metodě k-NN při klasifikaci poměrně odolný k šumu hodnoty v třídovém atributu.

5.1.2. Klasifikace onemocnění [7]

Pro úlohu klasifikace onemocnění systémem AnYa byl vybrán datový soubor ze studie [7]. Studie se zabývá klasifikací dvou onemocnění (Akutní zánět močového měchýře a akutní zánět ledvin) na základě odpovědí pacientů a jejich teploty. Datový soubor použitý pro klasifikaci se skládá ze záznamů reprezentujících každý jednoho pacienta. Vstupní hodnoty jsou:

- Teplota pacienta typu *real* <35,42>
- Výskyt nevolnosti {ano, ne}
- Bolest beder {ano, ne}
- Namáhavé močení {ano, ne}
- Mikce {ano, ne}
- Pálení/řezání močové trubice {ano, ne}

Výstupní hodnoty jsou diagnostikované nemoci {ano, ne}. Celkový počet záznamů v datovém souboru je 120 pacientů.

Ve studii [7] byly ke klasifikaci použity dvě hlavní metody a jejich variace - *Support vector machines* (SVM) a k-NN. Učení a validace byla realizována metodou „*holdout*“, tedy náhodným rozdělením datového souboru na dvě části, z nichž jedna byla použita jako tréninková množina záznamů a druhá jako ověřovací množina a podhodnocení se množiny záznamů prohodily. Ve studii se tento postup provádí 50 krát. Úspěšnost klasifikace byla hodnocena podle:

- Přesnost $\frac{TP+TN}{TP+TN+FP+FN}$
- Citlivost $\frac{TP}{TP+FN}$
- Specificita $\frac{TN}{TN+FP}$
- Prediktivní hodnoty pozitivního testu $\frac{TP}{TP+FP}$
- Prediktivní hodnota negativního testu $\frac{TN}{TN+FN}$

Kde:

TP - zjištěné případy výskytu znaku odpovídají skutečnosti

TN - zjištěné případy nepřítomnosti znaku odpovídají skutečnosti – správné zamítnutí

FP - odpovídá falešnému poplachu

FN - odpovídá přehlédnutému výskytu

Stejným způsobem byl testován i systém AnYa.

Pro potřeby systému AnYa byly odpovědi *{ano, ne}* kódovány jako *{1,-1}* a poté váhovány kvůli výhodnějšímu rozložení dat v prostoru (zjištěno experimentálně na základě analýzy tvorby Mračen), v případě výstupních veličin *{ano, ne}* kódováno jako *{-1,1}*. Při klasifikaci byla použita metoda „vítěz bere vše“, výstupem Mračen byla nejčastěji zastoupená hodnota výstupu Mračna. Parametr překrytí Λ byl zvolen e^{-1}. Hranice užitečnosti jednotlivých Mračen $\varepsilon_1 = 0.01$.

Při experimentu byla průměrná úspěšnost klasifikace systémem AnYa 97,20%. Systém vytvořil v průměru 11,9 Mračen a žádné z nich nepřekročilo spodní hranici užitečnosti. Průměrná specificita klasifikace byla 96,56% a průměrná citlivost 98,46%. Průměrná pravděpodobnost skutečně pozitivního výsledku byla 95,58%, průměrná pravděpodobnost skutečně negativního výsledku byla 98,61%. Srovnáním s výsledky studie [7] vyšel systém AnYa lépe, než použité metody k-NN, avšak nedosahoval 100% úspěšnosti klasifikace jako některé metody SVM (viz. Tabulka č. 4). Srovnání ostatních parametrů jako citlivost, specificita atd. nebylo možné, jelikož autor studie [7] neuvádí jejich průměrné hodnoty pro všech 50 testů. Lze je srovnat pouze pro nejúspěšnější metody SVM, které dosahují 100% ve všech hodnotách – systém AnYa tedy vyjde v porovnání s nimi jako horší varianta.

Metoda	Úspěšnost [%]	Max. Úspěšnost [%]	Min. Úspěšnost [%]
SVM-QP	100,00	100,00	100,00
SVM-SMO	100,00	100,00	100,00
SVM-LS	96,80	100,00	86,67
KNN-Euclidean	83,50	100,00	75,00
KNN-Cityblock	84,03	100,00	75,00
KNN-Cosine	89,59	100,00	75,00
KNN-Correlation	84,40	91,67	75,00
AnYa	97,20	100,00	79,71

Tabulka č. 4: Srovnání přesnosti klasifikace onemocnění mezi systémem AnYa a algoritmy použitými v [7]

5.1.3. Klasifikace znalostí studentů [6]

Datový soubor pro klasifikaci znalostí studentů byl převzat ze studie [6]. Jednalo se o data sbíraná systémem pro studenty, který na základě jejich akcí tvořil modely jednotlivých studentů (User modeling system – UMS). Data jsou tvořena pěti vstupy typu *real <0,1>*, reprezentujících výstup z UMS (stupeň času stráveným studiem cílové látky, stupeň opakování cílové látky, stupeň času stráveným studování souvisejících okruhů s cílovou látkou, výkon studentů na zkoušce z okruhů souvisejících s cílovou látkou a výkon studentů na zkoušce z cílové látky) a jedním výstupem typu *enum {Very low, Low, Middle, High}* reprezentujícím úroveň znalostí studentů. Celkový dostupný počet vzorků v datovém souboru byl 403.

Ve studii [5] bylo pro klasifikaci úrovně znalostí studentů použito sedm klasifikačních algoritmů:

- Bayesův klasifikátor
- k-NN klasifikátor (k={1,3,5,7})
    - S výpočtem vzdálenosti jednotlivých vzorků Euklidovskou metrikou (EU)
    - S výpočtem vzdálenosti jednotlivých vzorků metrikou Manhattan (MA)
    - S výpočtem vzdálenosti jednotlivých vzorků metrikou Minkovski (MI)
- Intuitivní znalostní klasifikátor (IKC) založený na metodě k-NN (k={1,3,5,7})
    - S výpočtem vzdálenosti jednotlivých vzorků Euklidovskou metrikou (EU)
    - S výpočtem vzdálenosti jednotlivých vzorků metrikou Manhattan (MA)
    - S výpočtem vzdálenosti jednotlivých vzorků metrikou Minkovski (MI)

Studií [6] navrhovaný klasifikátor (IKC) je kombinací genetických algoritmů a k-NN metody.

Pro porovnání výsledků z [6] a systémem AnYa byly navozený stejné podmínky experimentu. Z celkového datového souboru bylo náhodně vybráno 258 vzorků, které byly poté použity jako tréninková množina dat algoritmu. Zbylých 145 vzorků bylo použito pro validační množinu dat pro algoritmus. Tento postup byl opakován celkem 12 krát. Poté byl algoritmus vyhodnocen na základě průměrného počtu špatně zařazených vzorků a průměrné chyby zařazení v procentech a
průměrné úspěšnosti klasifikace. Pro potřeby systému AnYa byla výstupní hodnota
enum {Very low, Low, Middle, High} kódována jako *integer {1, 2, 3, 4}*.

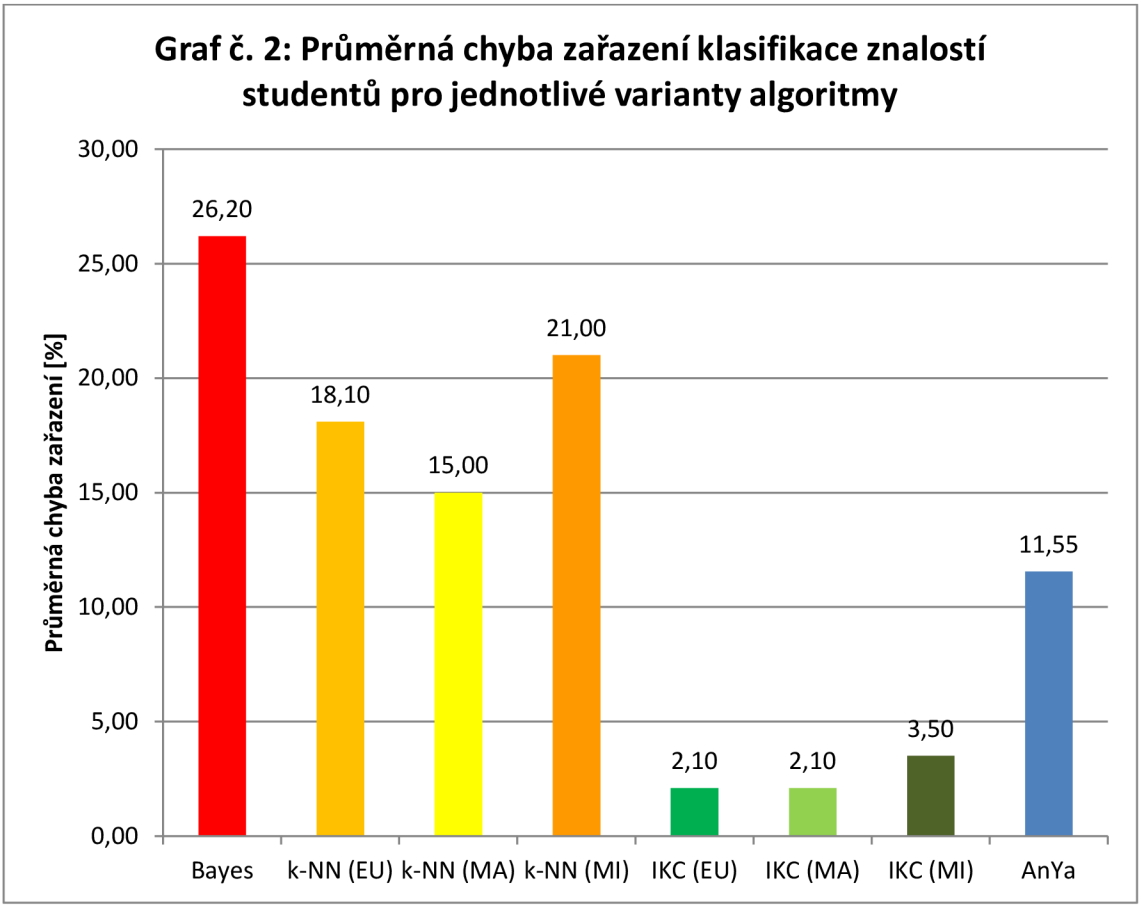
V systému AnYa byla pro rozhodovací mechanismus náležitosti datového vzorku jednotlivým Mračnům použita metoda „vítěz bere vše“. Parametr překrytí Λ byl zvolen e^{-1}. Hranice užitečnosti jednotlivých Mračen $\varepsilon_1 = 0.01$, jelikož byla předpokládána tvorba menšího celkového počtu Mračen.

Díky rozložení vstupních dat v prostoru systém AnYa nedosahoval dobrých výsledků (průměrná přesnost zařazení byla okolo 35%). Problém byl v míře tvorby Mračen, jelikož vstupní data se jevila algoritmu příliš blízká. Experimentálně bylo zjištěno, že preprocessingem vstupních dat – přidání váhy jednotlivým vstupům - se dosáhne lepšího rozložení dat a systém na ně bude lépe reagovat. S takto upravenými daty bylo dosaženo mnohem lepší míry úspěšnosti klasifikace systému AnYa.

Algoritmus	Bayesův klasifikátor	k-NN klasifikátor			IKC klasifikátor			systém AnYa
		EU	MA	MI	EU	MA	MI	
Průměrný počet chybně zařazených vzorků [-]	38,0	26,2	21,7	30,5	3,0	3,0	5,0	16,7
Průměrná chyba zařazení [%]	26,2	18,1	15,0	21,0	2,1	2,1	3,5	11,5
Průměrná přesnost zařazení [%]	73,8	81,9	85,0	79,0	97,9	97,9	96,5	88,5

Tabulka č. 5: Srovnání kvality klasifikace znalostí studentů mezi systémem AnYa a algoritmy použitými ve studii [6]

Ve výše popsaném experimentu klasifikoval systém AnYa validační data s přesností 88,45%. V průběhu klasifikace si systém vytvořil průměrně 59.1 Mračen, což naznačuje velké rozptýlení vstupních dat. Ve srovnání s výsledky klasifikačních algoritmů použitých ve studii [6] se systém AnYa umístil těsně před algoritmus k-NN užívající metriku Manhattan (85%). Jeho přesnost nedosahovala kvalit studií navrhovaných Intuitivních znalostních klasifikátorů, avšak byl výrazně úspěšnější než Bayesův klasifikátor (viz. Tabulka č. 5, Graf č. 2).



Graf č. 2: Průměrná chyba klasifikace znalostí systému. Úspěšnosti algoritmů jiných než AnYa použity ze studie [6]

5.2. Aproximace

Pro otestování algoritmu byly zvoleny dvě úlohy aproximace. První byl převzat experiment ze studie [3], která se zabývá systémy AnYa. Na tomto experimentu bylo ověřeno pochopení metody tím, že bylo dosaženo podobných výsledku (viz Kapitola 5.2.1). Druhý soubor dat byl vybrán pro ověření MIMO aproximace.

5.2.1. Aproximace koncentrace CO2 v laboratorní plynové peci [3]

Pro ověření pochopení konceptu byla provedena stejná aproximace jako v původní studii navrhující systém AnYa [3]. Systém je zde testován na datech získaných z laboratorní pece. Soubor dat se skládá z jednoho vstupu a jednoho výstupu. Aproximovaná funkce byla koncentrace CO2 v plynové peci. Podle různých studií [3] lze tuto funkci nejlépe aproximovat jako:

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

Proto byla data upravena. Vstupem jsou hodnoty $y(k-1)$ a $u(k-4)$, výstupem koncentrace CO2 v peci $y(k)$. Datový soubor obsahoval celkem 290 záznamů. Pro srovnání algoritmů byl zopakován experiment popsaný v [3]. Prvních 200 vzorků se použije jako tréninkový soubor algoritmu, zbylých 90 vzorků pro validační soubor. Použitá data ve studii byla normalizována.

Pro vyhodnocení přesnosti aproximace použitým algoritmem byl použit tzv. *non-dimensional error index* (NDEI), který je definován jako podíl střední kvadratické chyby (*root mean square error RMSE*) směrodatnou odchylkou dat a tudíž je nezávislý na použité normalizaci.

$$NDEI = \frac{RMSE}{\sigma} = \frac{\sqrt{\frac{1}{N}\sum_i (y_i - \hat{y}_i)^2}}{\sqrt{\frac{1}{N}\sum_i (y_i - \mu)^2}}, \qquad \mu = \frac{1}{N}\sum_i y_i, \qquad i = [i..N] \tag{5.3}$$

Kde
y_i - *skutečná hodnota výstupu*
$\hat{y}_i$ - *aproximovaná hodnota výstupu*

Ve studii [3] je systém AnYa srovnáván s několika dalšími algoritmy. Patří mezi ně modely ANFIS (Adaptive Neuro Fuzzy Interference System), Genfis2, DENFIS (Dynamic Evolving Neuro-Fuzzy Inference System) a eTS+ (evolving Takagi-Sugeno).

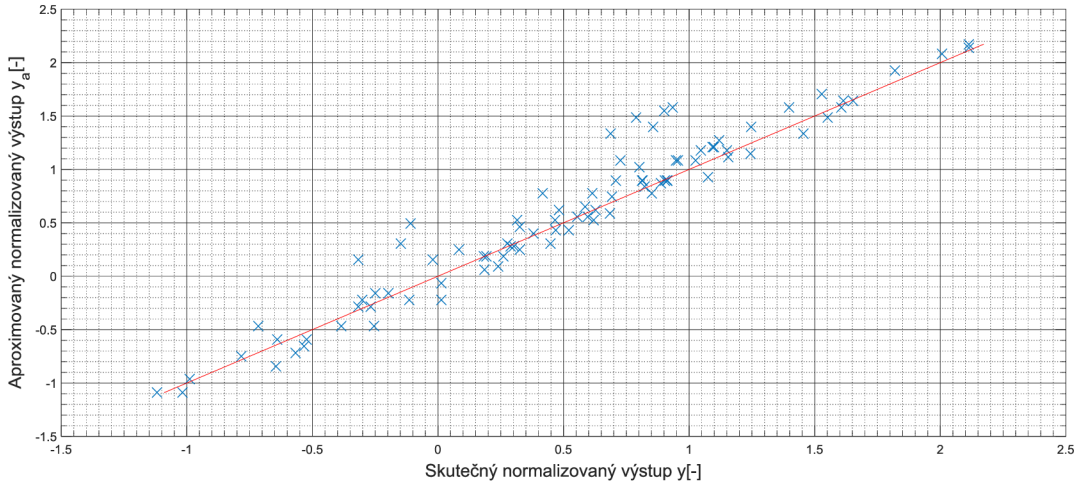
Pro učící část algoritmu AnYa použitém v této práci byly nastaveny parametry $\Omega = 10^9$, parametr překrytí $\Lambda = e^{-1}$ a spodní hranice užitečnosti Mračen $\varepsilon_1 = 0.01$. Vstupní data algoritmu byla normalizována pomocí metody z-score. Závěry byly tvořeny jako lineární kombinace vstupních hodnot stejně jako systém AnYa použitý ve studii [3]. Srovnání dosažených výsledků je v Tabulce č. 6.

Metoda	ANFIS	Genfis2	DENFIS	eTS+	AnYa [3]	AnYa
NDEI [-]	0,605	0,311	0,322	0,291	0,272	0,274
pravidla	25	3	10	7	7	4

Tabulka č. 6: Srovnání kvality aproximace mezi systémy použitými v [3] a systémem AnYa naprogramovaným v rámci této práce

Lze vidět, že systém AnYa vytvořený pro účely této práce dosahoval srovnatelného indexu NDEI jako systém AnYa vytvořený ze studie [3]. Rozdíl mezi počtem pravidel může být způsoben rozdílným nastavením obou systémů či jiným způsobem normalizace vstupních dat. V Grafu č. 2 lze vidět porovnání výstupů aproximovaných a skutečných.

Graf č. 3: Porovnání normalizovaných výstupů aproximovaných a skutečných pro aproximaci koncentrace CO_2



Graf č. 3: Porovnání aproximovaných a skutečných výstupů koncentrace CO_2

Na provedeném experimentu byl prověřen i vliv šumu na aproximaci systému AnYa.

Byla zvolena simulace šumu senzorů, tedy vstupů vzorků dat vstupujících do algoritmu. Zvolený typ šumu byl šum s normálním (Gaussovým) rozložením. Pro každý vstup byl vygenerován vektor náhodných hodnot šumu vynásobený požadovanou úrovní šumu o počtu hodnot rovných počtu vstupních vzorků. Tento vektor má střední hodnotu rovnou nule a směrodatnou odchylku rovnou požadované úrovni šumu (0.3 = 30%). Vzorku vstupu byla poté přičtena jeho hodnota vynásobená korespondující

hodnotou z vektoru náhodných hodnot šumu. Tím se docílilo hodnot vstupů, které obsahují šum s nulovou střední hodnotou a směrodatnou odchylkou v požadované úrovni šumu.

$$\hat{x}_i = x_i + x_i \cdot G_i \cdot s \tag{5.4}$$

Kde

G *je vektor náhodných hodnot* $\langle -1,1 \rangle$ *se střední hodnotou* 0

x_i *skutečná je hodnota vstupu*

$\hat{x}$ *je hodnota upraveného vstupu s přidaným šumem*

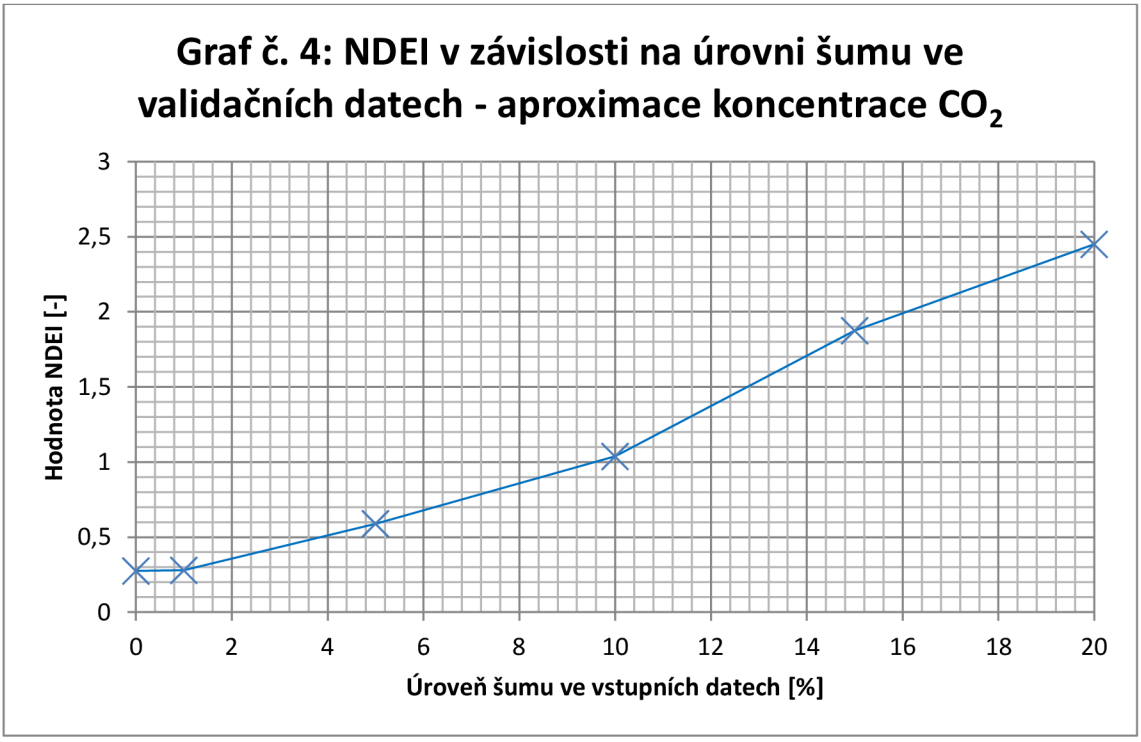
s *je úroveň šumu* $\langle 0,1 \rangle$

Nejprve byla data obsahující šum ve vstupech obsažena ve validačním souboru dat. Tato situace reprezentuje aproximaci výstupu již naučeného offline systému AnYa na základě upravených vstupů. Postupně byla aplikována úroveň šumu 1%, 5%, 10%, 15% a 20%. Pro dosažené hodnoty aproximace byl spočítán index NDEI. Výsledky byly srovnány s experimentem bez šum obsahujících vstupních dat (viz Tabulka č. 7).

Úroveň šumu [%]	0	1	5	10	15	20
NDEI [-]	0,274	0,2788	0,5893	1,0372	1,8748	2,4494

Tabulka č. 7: Zjištěné hodnoty indexu NDEI pro jiné úrovně šumu ve validačních datech aproximace CO_2

Z Tabulky č. 7 lze vyčíst strmě klesající přesnost aproximace (zvyšující se hodnota indexu NDEI) výstupní hodnoty v závislosti na úrovni šumu ve vstupních datech. Hodnoty NDEI vyšší než 1 už znamenají velice špatnou aproximaci (viz Graf č. 3).



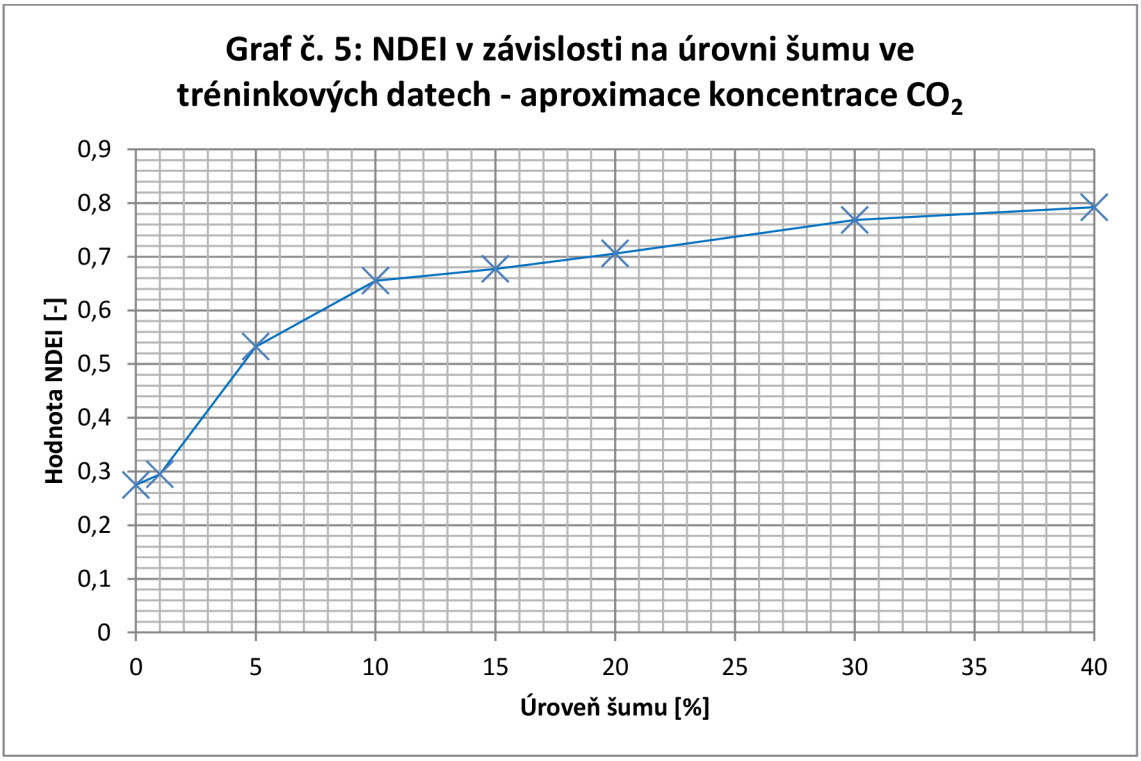
Graf č. 4: Závislost indexu NDEI na úrovni šumu ve validačních datech – aproximace koncentrace CO_2

Výrazný vliv šumu na přesnost je způsoben zejména použitou metodou tvorby závěrů. Výstup je díky použité vážené metodě nejmenších čtverců skládán jako $y = \psi^T\theta$, kde $\psi = [\lambda^1 x_e^T, \lambda^2 x_e^T \ldots, \lambda^N x_e^T]^T$ (3.21). Objeví-li se odchylka ve vstupu, tak je její hodnota vynásobená $\theta\lambda$. Systém se při validaci již neučí a parametry zůstávají neměnné, nemohou tedy reagovat na šum.

Dále byl prozkoumán vliv šumu s normálním rozložením na vstupy souboru tréninkových dat. Jedná se o situaci, kdy se systém AnYa učí na degradovaných datech. Aplikované úrovně šumu čítaly 1 až 40% (viz. Tabulka č. 8). Pro vypočítané hodnoty výstupu aproximace byl spočítán index NDEI. Dosažené hodnoty NDEI jsou zobrazeny v Tabulce č. 8.

Úroveň šumu [%]	0	1	5	10	15	20	30	40
NDEI [-]	0,2747	0,2947	0,5327	0,6554	0,6774	0,7061	0,7685	0,7921
Počet vytvořených Mračen [-]	4	4	6	7	8	9	8	7

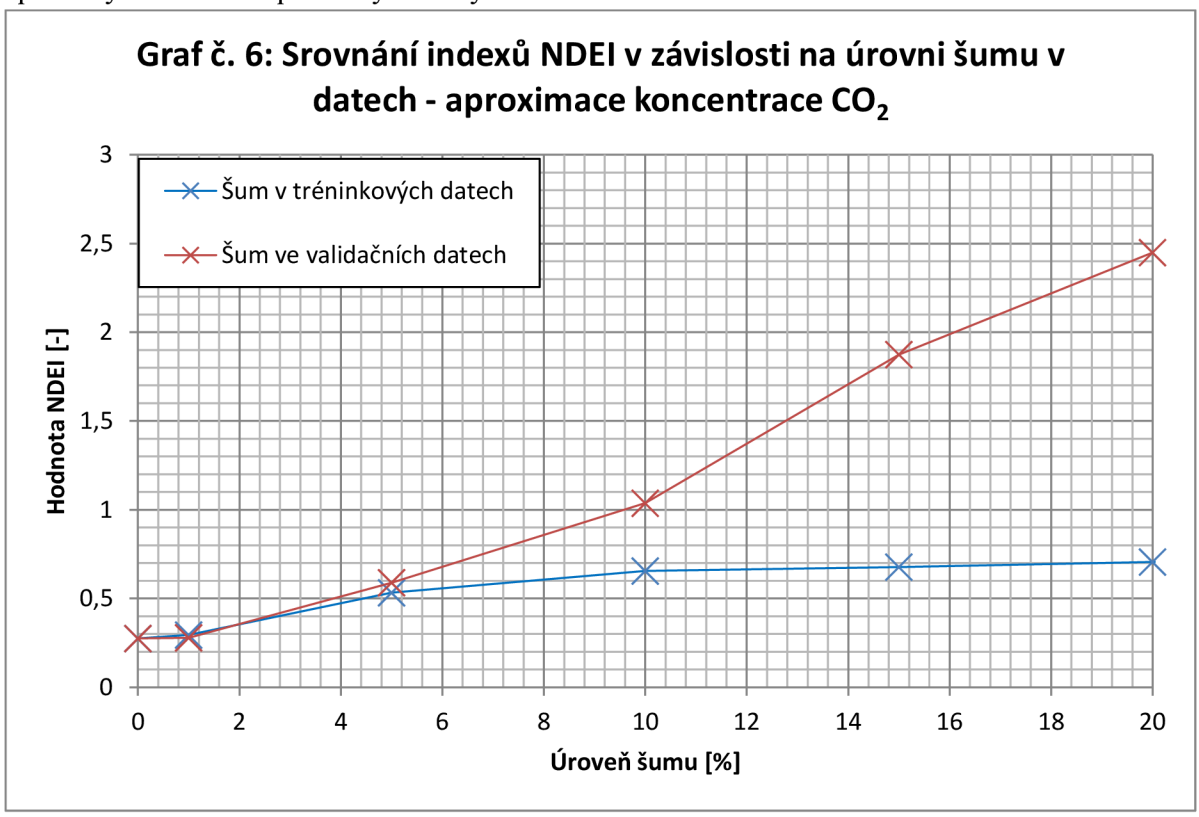
Tabulka č. 8: Zjištěné hodnoty indexu NDEI pro jiné úrovně šumu v tréninkových datech aproximace CO_2



Graf č. 5: Závislost indexu NDEI na úrovni šumu v tréninkových datech – aproximace koncentrace CO_2

V případě dat tréninkových obsahujících šum dopadla aproximace výstupu systémem AnYa lépe, než při přítomnosti šumu v datech validačních (viz Graf č. 6). Po prudkém indexu NDEI při úrovni šumu 0-10% se zhoršování přesnost klasifikace výrazně zpomalilo. Zajímavý je i počet vytvořených Mračen, kdy systém reaguje na šum v datech zvýšeným počtem pravidel (Mračen). To se děje až do úrovně šumu 20%. Nad

touto úrovní již začínají být vstupní data příliš rozptýlená ve vstupně-výstupním prostoru a systém AnYa v nich přestává nacházet souvislosti, což se projevuje opětovným klesáním počtu vytvořených mračen.



Graf č. 6: Srovnání závislosti indexu NDEI na úrovni šumu v tréninkových a validačních datech – aproximace koncentrace CO_2

5.2.2. Aproximace energetické účinnosti budovy [8]

Systém AnYa naprogramovaný pro účely této práce byl od začátku koncipován jako MIMO systém. Pro ověření, že je algoritmus schopen aproximovat funkci o více výstupech, byla vybrána data ze studie [8]. Jedná se o soubor 768 dat reprezentujících modely o standardním objemu $771{,}75\ m^3$, avšak rozdílných plochách a proporcích. Budovy byly popsány 8mi parametry, které reprezentovaly jejich rozložení (plocha, plocha stěn, plocha oken…). Výstupy reprezentovaly modelové hodnoty tepelného zatížení a chladícího výkonu.

Cílem studie [8] bylo aproximovat hodnoty výstupů jako funkci vstupních proměnných. K tomuto účelu byly použity dva algoritmy:

- Iterační metoda nejmenších čtverců (*Iteratively reweighted least squares – IRLS*)
- Metoda Náhodného lesa (*Random forrests - RF*)

Experiment spočíval v učení a validaci algoritmů metodou Křížové validace, konkrétně typ k-fold s $k = 10$. Pro vyhodnocení přesnosti aproximace algoritmů byla použita střední absolutní chyba (*mean absolute error - MAE*), střední kvadratická odchylka (*mean square error - MSE*) a střední relativní chyba v procentech (*mean relative error - MRE*).

$$MAE = \frac{1}{N}\sum_{i}|y_i - \hat{y}_i|, \quad i = [i..N] \tag{5.5}$$

$$MSE = \frac{1}{N}\sum_{i}(y_i - \hat{y}_i)^2, \quad i = [i..N] \tag{5.6}$$

$$MRE = 100 * \frac{1}{N}\sum_{i}\frac{|y_i - \hat{y}_i|}{y_i}, \quad i = [i..N] \tag{5.7}$$

Kde:

y_i *– skutečný výstup*

$\hat{y}_i$ *– predikovaný výstup*

N – počet validačních dat

Křížová validace proběhla celkem 100 krát. Ze všech dostupných hodnot chyb byla vypočítána jejich střední hodnota a směrodatná odchylka.

Stejným způsobem byl otestován i systém AnYa. Pro učící část byly algoritmu nastaveny parametry $\Omega = 10^2$, parametr překrytí $\Lambda = e^{-1}$ a spodní hranice užitečnosti Mračen $\varepsilon_1 = 0.01$. Závěry byly tvořeny jako lineární kombinace vstupních hodnot, podobně jako IRLS metoda použita v [8].

V průběhu experimentu systém AnYa vytvořil průměrně 33,22 Mračen. Mračna si po dobu života udržovala vysokou užitečnost, proto docházelo k promazávání struktury pouze sporadicky. Srovnání výsledků s metodami použitými v [8] je zobrazeno v Tabulce č. 9.

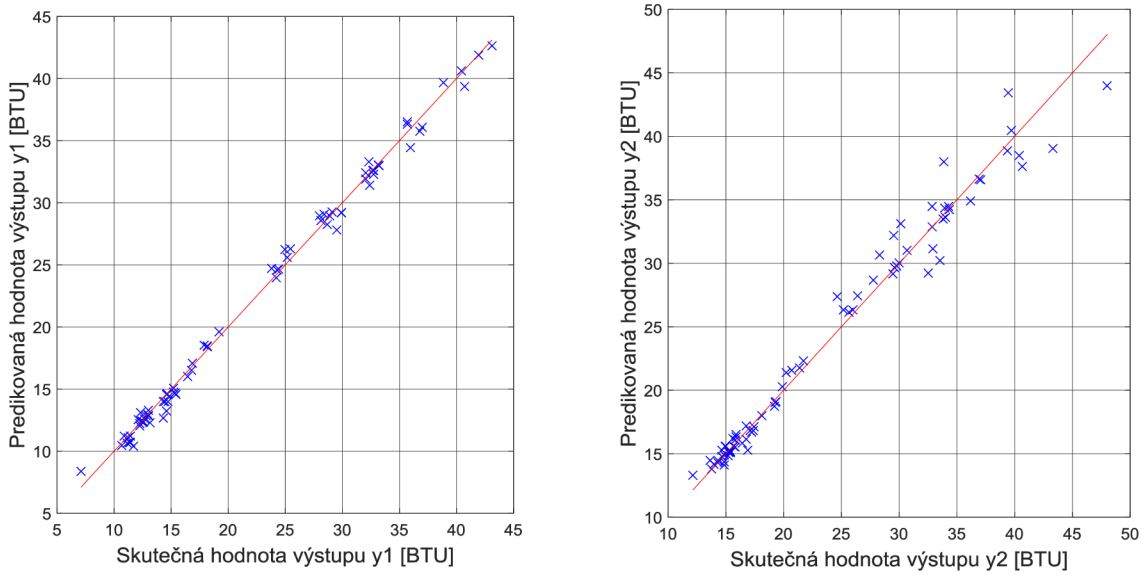
	výstup	IRLS	RF	AnYa
MAE [-]	y1	2,14 ± 0,24	0,51 ± 0,11	0,93 ± 0,25
	y2	2,21 ± 0,28	1,42 ± 0,25	1,39 ± 0,44
MSE [-]	y1	9,87 ± 2,41	1,03 ± 0,54	2,29 ± 2,59
	y2	11,46 ± 3,63	6,59 ± 1,56	5,44 ± 7,56
MRE [%]	y1	10,09 ± 1,01	2,18 ± 0,64	4,86 ± 1,85
	y2	9,41 ± 0,80	4,62 ± 0,70	5,89 ± 9,30

Tabulka č. 9: Srovnání kvality aproximace energetické účinnosti budovy pro systém AnYa a metody použité ve studii [8]

Systém AnYa nedosahoval přesnosti aproximace ve studii [8] použité metody Náhodných stromů. V případě aproximace druhého výstupu se přesnost aproximace průměrně blíží kvalitě RF algoritmu, avšak podle směrodatné odchylky lze říci, že přesnost v průběhu Křížové validace značně kolísala. To lze pozorovat zejména u MRE. V porovnání s IRLS dosahoval systém AnYa přibližně dvakrát lepších průměrných výsledků.

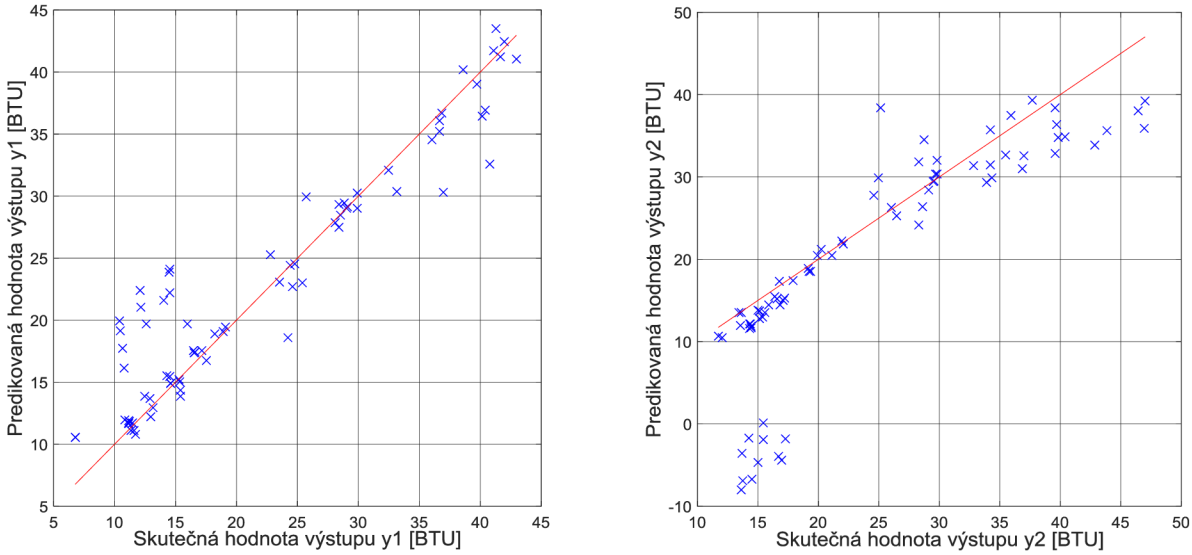
Metoda IRLS aproximovala oba výstupy s podobnou přesností, avšak metody RF a AnYa byly úspěšnější v aproximování prvního výstupu. Pro názornost je v Grafech č. 7 a 8 zobrazeno porovnání skutečných výstupů a predikovaných výstupů pro nejpřesnější a nejméně přesnou aproximaci systému AnYa (hodnoceno na základě součtu MRE).

Graf č. 7: Porovnání skutečných a aproximovaných hodnot výstupů aproximace energetické účinnosti budovy - Nejúspěšnější aproximace



Graf č. 7: Porovnání skutečné a aproximované energetické účinnosti budov – nejlepší výsledek

Graf č. 7: Porovnání skutečných a aproximovaných hodnot výstupů aproximace energetické účinnosti budovy - Nejhorší aproximace



Graf č. 8: Porovnání skutečné a aproximované energetické účinnosti budov – nejhorší výsledek

6. ZÁVĚR

V rámci diplomové práce byl představen nový fuzzy systém AnYa [1][3]. Jedná se o fuzzy systém s netradiční tvorbou předpokladů (antecedenty) fuzzy pravidel. Tvorba předpokladů probíhá přiřazováním dat do tzv. datových Mračen, která reprezentují neparametricky definované shluky dat. Rozdíl tvorby předpokladů použitím datových Mračen oproti klasickým fuzzy systémům typu Takagi-Sugeno či Mamdani spočívá v absenci nutnosti parametrizace vstupně-výstupního datového prostoru systému. Práce se v Kapitole 2. zabývá problematikou fuzzy logiky, představením klasických fuzzy systémů a nastíněním mechanismu tvorby závěrů klasickými fuzzy systémy, aby byl vysvětlen rozdíl mezi nimi a novým systémem AnYa. Samotný fuzzy systém AnYa je popsán v Kapitole 3, kde je nastíněn postup tvorby datových Mračen, jejich začlenění do fuzzy systému a následně i tvorba závěrů (konsekventy) za pomoci fuzzy vážené metody nejmenších čtverců.

V Kapitole 4. je popsaný fuzzy systém AnYa naprogramován v prostředí MATLAB. Práce se zde věnuje řešení systému, popisuje datovou strukturu programu a funkce použité k realizaci systému.

V poslední části práce je naprogramovaný systém AnYa otestován na vybraných příkladech klasifikace a aproximace funkce. Pro obě varianty bylo realizováno několik experimentů, které se opíraly o data z různých studií zabývajících se klasifikací či modelováním/aproximací ([3][6][7][8]).

Klasifikační experimenty zahrnovaly tři experimenty. V prvním bylo za úkol klasifikovat pravost bankovek na základě 4 atributů získaných analýzou obrazu ze senzoru. Pro klasifikaci byly použity metody k-NN s parametrem k={1,3,5} a systém AnYa. Jako úspěšnější se ukázala metoda k-NN, která byla schopna klasifikovat pravost bankovek s úspěšností 99,81% (k=1) až 99,83 (k=3). Systém AnYa klasifikoval s mnohem horší přesností 94,72% (viz Tabulka č. 2). S přidáním šumu hodnoty v třídovém atributu do dat se však ukázalo, že metoda AnYa je mnohem odolnější vůči šumu v datech (viz Tabulka č. 3), kdy přesnost metody k-NN prudce klesala, kdežto přesnost systému AnYa se začala znatelně měnit až od úrovně šumu hodnoty v třídovém atributu vyšší než ve dvaceti procentech datových záznamů (Graf č. 1).

Další experiment se týkal klasifikace onemocnění [7]. Jednalo se o klasifikaci dvou onemocnění (dva výstupy) na základě teploty pacienta a odpovědí na několik otázek týkajících se jeho zdravotního stavu (6 atributů). Experiment byl proveden způsobem popsaným ve studii [7], čímž bylo dosaženo možnosti porovnání výsledků s použitými metodami ve studii použitými. Byly to tři verze metody Support vector machines (SVM) a čtyři verze metody k-NN. Systém AnYa klasifikoval oba výstupy s přesností 97,2%. Oproti předchozímu experimentu byl systém AnYa v klasifikaci onemocnění

výrazně úspěšnější než všechny použité metody k-NN (přesnost 83,5%-89,6%). Nedosahoval ovšem 100% přesnosti klasifikace jako dvě ze tří metod SVM (viz Tabulka č. 4).

Poslední klasifikační úloha se zabývala klasifikací znalostí studentů [6]. Klasifikované datové záznamy vznikaly modelováním studentů systémem UMS (User modeling systém) z informačního systému pro studenty (čas strávený nad učivem apod.). Na základě těchto dat byla klasifikována úroveň znalostí studentů. Ve studii [6] jsou ke klasifikaci použity varianty metody k-NN a autory navrhovaný Intuitivní znalostní klasifikátor (IKC) založený na metodě k-NN. Pro porovnání byl uváděn Bayesův klasifikátor. Metody k-NN klasifikovaly s přesností 79%-85%, kdežto autory navrhovaná metoda IKC klasifikovala s přesností až 97,9%. Systém AnYa dosáhl přesnosti 88,5% což je mírně lepší výsledek než metody k-NN, avšak zdaleka nedosahuje přesnosti metody IKC. Všechny použité metody byly výrazně úspěšnější, než Bayesův klasifikátor, který dosáhl přesnosti 73,8% (viz Tabulka č. 5).

Z výsledků experimentů zabývajících se klasifikací dat lze usoudit, že systém AnYa je podobně úspěšný v klasifikaci dat, jako metoda k-NN avšak je výrazně odolnější k šumu hodnoty v třídovém atributu.

Experimenty použité ke zhodnocení aproximace funkce systémem AnYa se týkaly aproximace koncentrace CO_2 v laboratorní peci a aproximace energetické účinnosti budovy. První z nich byl použit pro ověření pochopení konceptu fuzzy systému AnYa, jelikož byl původně navřen původními autory systému AnYa, tudíž bylo umožněno porovnat výsledky naprogramovaného systému AnYa v rámci diplomové práce a systému AnYa vytvořeným autory studie [3]. Aproximovaná funkce měla dva vstupy a jeden výstup. Pro porovnání kvality aproximace byl autory použit index NDEI (*non-dimensional error index*), který je definován jako podíl střední kvadratické chyby a směrodatné odchylky skutečných výstupních dat. Autoři ve studii [3] porovnávali systém AnYa s dalšími fuzzy systémy jako ANFIS, DENFIS, eTS+ a Genfis2. V Tabulce č. 6 jsou srovnány výsledky aproximace všech metod. Systém AnYa použitý ve studii [6] dosahoval téměř totožných výsledků (NDEI=0,272), jako systém AnYa naprogramovaný v rámci diplomové práce (NDEI=0,274). Výsledky se lišily pouze v počtu vytvořených pravidel (Mračen), což lze vysvětlit jinou normalizací dat, rozdílným nastavením systému a mírně odlišnou verzí naprogramování systému.

Na použitých datech byl také ověřen vliv šumu na aproximaci funkce systému AnYa. Nejprve byl aplikován šum na validační data (veškeré vstupní hodnoty validačních dat byly obohaceny o šum s normálním rozložením podle rovnice 5.4). Pro vyhodnocení vlivu šumu byl použit index NDEI. Aplikovaný šum byl v úrovních 1%-20%. Ve výsledcích lze pozorovat výrazný vliv šumu validačních dat na aproximaci funkce systémem AnYa. Toto je nejspíše způsobeno zejména použitím vážené metody

nejmenších čtverců na tvorbu závěru fuzzy systému AnYa. Zaznamenané výsledky jsou v Tabulce č. 7 a v Grafu č. 4.

Vliv šumu s normálním rozložením v datech tréninkového souboru byl popsán v další části experimentu. Systém AnYa se učil na degradovaných datech s různou úrovní šumu a jeho vliv byl hodnocen změnou indexu NDEI. V tomto případě se projevila větší odolnost systémů AnYa na šum v tréninkových datech. Aplikován byl šum v rozmezí úrovní 1% až 40% (viz Tabulka č. 8 a Graf č. 5) a hodnota indexu NDEI nepřekročila 0,8 ani při úrovní 40%, kdežto při šumu aplikovaném na validační data systém překročil tuto hodnotu indexu NDEI již při úrovni šumu 10%. Srovnání vlivu šumu ve validačních a tréninkových datech se zobrazeno na Grafu č. 6.

Poslední experiment se věnoval aproximaci energetické účinnosti budov[8]. Soubor dat obsahoval vzorky s osmi vstupy reprezentujícími různé parametry modelů budov a dvěma výstupy – hodnota tepelného zatížení a hodnota chladícího výkonu. Pro aproximaci byly v původní studii [8] použity metody IRLS (Iterační metoda nejmenších čtverců) a RF (metoda Náhodného lesa). K vyhodnocení kvality aproximace byla použita střední absolutní chyba (*mean absolute error – MAE* (5.5)), střední kvadratická odchylka (*mean square error – MSE* (5.6)) a střední relativní chyba (*mean relative error – MRE* (5.7)). Experiment probíhal metodou Křížové validace, hodnoty chyb byly zprůměrovány. V Tabulce č. 9 lze vidět srovnání systému AnYa s metodami IRLS a RF. Systém AnYa aproximoval obě hodnoty výstupů průměrně dvakrát lépe ($MRE_{y1} = 4{,}89\%, MRE_{y2} = 5{,}89\%$), než metoda IRLS ($MRE_{y1} = 10{,}09\%, MRE_{y2} = 9{,}41\%$) a téměř se vyrovnal metodě RF v aproximaci druhého výstupu ($MRE_{y1} = 2{,}18\%, MRE_{y2} = 4{,}62\%$). Ovšem vyšší směrodatná odchylka výsledků naznačuje vyšší nestálost přesnosti aproximace systémem AnYa (viz Tabulka č. 9).

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. DOI: 10.1109/is.2010.5548369. ISBN 9781424451647.

[3] ANGELOV, Plamen a Ronald YAGER. A new type of simplified fuzzy rule-based system. In: *International Journal of General Systems*. 2012, **41**(2), s. 163-185. DOI: 10.1080/03081079.2011.634807.

[4] JURA, Pavel. *Základy fuzzy logiky pro řízení a modelování*. Brno: nakladatelství VUTIUM, 2003.

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

[6] KAHRAMAN, H. T., S. SAGIROGLU a I. COLAK. Developing intuitive knowledge classifier and modeling of users' domain dependent data in web. *Knowledge Based Systems*. **2013**(vol. 37), 283-295.

[7] SEYYID, Ahmed Medjahed, Ait Saadi TAMAZOUZT a Abdelkader BENYETTOU. Urinary System Diseases Diagnosis Using Machine Learning Techniques. *International Journal of Intelligent Systems and Applications* [online]. 2015, vol. 7, no. 5, s. 1-7. ISSN 2074904X.

[8] TSANAS, Athanasios a Angeliki XIFARA. Accurate quantitative estimation of energy performance of residential buildings using statistical machine learning tools. *Energy and Buildings* [online]. 2012, vol. 49, s. 560-567. ISSN 0378-7788, 0378-7788.

[9] Noisy Data in Data Mining. *SCI2S* [online]. [cit. 2016-05-15]. Dostupné z: http://sci2s.ugr.es/noisydata

Seznam příloh

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