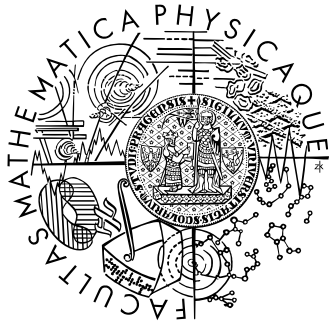
Univerzita Karlova v Praze

Matematicko-fyzikální fakulta

DIPLOMOVÁ PRÁCE



Bc. Miroslav Jakubík

RBF-sítě s dynamickou architekturou

Katedra teoretické informatiky a matematické logiky

Vedoucí diplomové práce: doc. RNDr. Iveta Mrázová, CSc.

Studijní program: Informatika, Teoretická informatika

2010

Chcel by som poďakovať všetkým, ktorí mi pomohli s touto prácou. V prvom rade vedúcej mojej práce doc. RNDr. Ivete Mrázovej, CSc. za trpezlivosť a množstvo dobré mierených rád. Ďalej by som chcel poďakovať za podporu celej mojej rodine a priateľom.

Prehlasujem, že som svoju diplomovú prácu napísal samostatne a výhradne s použitím citovaných prameňov. Súhlasím so zapožičiavaním práce.

V Prahe dňa Bc. Miroslav Jakubík

Obsah

Názov práce: RBF-sítě s dynamickou architekturou
Autor: Bc. Miroslav Jakubík
Katedra: Katedra teoretické informatiky a matematické logiky
Vedúca diplomovej práce: doc. RNDr. Iveta Mrázová, CSc.
e-mail vedúcej: Iveta.Mrazova@mff.cuni.cz

Abstrakt: V tejto diplomovej práci som zrekapituloval viacero metód vhodných pre klastrovanie vstupných dát. Predstavil som dva dobré známe klastrovacie algoritmy, a to konkrétne K-means algoritmus a Fuzzy C-means (FCM) algoritmus. Uviedol som niekoľko metód vhodných pre odhad optimálneho počtu klastrov. Ďalej som predstavil Kohonenové mapy a dva modely Kohonenových máp s adaptívnou topológiou, konkrétne Kohonenové mapy s rastúcou mriežkou a model rastúcich neurónových plynov. Ako posledný som predstavil pomerne nový model radiálne bázických neurónových sieti. Pre tento typ neurónových sieti som uviedol viacero učiacich algoritmov. V závere práce som aplikoval jednotlivé klastrovacie metódy na reálne dáta popisujúce vzájomný obchod štátov sveta.

Kľúčové slová: klastrovanie, K-means algoritmus, Fuzzy C-means klastrovanie, Kohonenové siete, radiálne bázické neurónové siete

Title: RBF-networks with a dynamic architecture
Author: Bc. Miroslav Jakubík
Department: Department of Theoretical Computer Science and Mathematical Logic
Supervisor: doc. RNDr. Iveta Mrázová, CSc.
Supervisor's e-mail address: Iveta.Mrazova@mff.cuni.cz

Abstract: In this master thesis I recapitulated several methods for clustering input data. Two well known clustering algorithms, concretely K-means algorithm and Fuzzy C-means (FCM) algorithm, were described in the submitted work. I presented several methods, which could help estimate the optimal number of clusters. Further, I described Kohonen maps and two models of Kohonen's maps with dynamically changing structure, namely Kohonen map with growing grid and the model of growing neural gas. At last I described quite new model of radial basis function neural networks. I presented several learning algorithms for this model of neural networks. In the end of this work I made some clustering experiments with real data. This data describes the international trade among states of the whole world.

Keywords: clustering, K-means algorithm, Fuzzy C-means clustering, Kohonen networks, radial basis neural networks

1. Úvod

Hlavnú tému tejto diplomovej práce tvorí problematika klastrovania dát. V dnešnej dobe je v každom vednom obore k dispozícií obrovské množstvo dát. Skúmanie týchto dát, objavovanie určitých vnútorných štruktúr a vzájomných závislostí, je predmetom stáleho záujmu. Uvedieme preto viacero nielen overených a dobre preskúmaných klastrovacích algoritmov, ale si predstavíme aj pomerne nový model neurónových sieti. Táto diplomová práca je rozčlenená do deviatich kapitol nasledovným spôsobom.

V druhej kapitole nájdeme stručný úvod do problematiky klastrovania a uvedieme mechanizmus, ktorý nám umožní pracovať so vstupnými dátami.

V tretej kapitole si predstavíme K-means algoritmus. Tento algoritmus vyžaduje od užívateľa zadanie požadovaného počtu klastrov, do ktorých chce vstupné dáta rozdeliť. Z tohto dôvodu v tejto kapitole spomenieme určité metódy, ktoré môžu pomôcť pri hľadaní optimálneho počtu klastrov.

Vo štvrtej kapitole si predstavíme FCM algoritmus. Podobne ako K-means algoritmus aj tento algoritmus vyžaduje znalosť počtu klastrov pred samotným spustením algoritmu. Uvedieme si niekoľko kritérií validity klastrov, ktoré hodnotia kvalitu rozdelenia vstupných dát do klastrov.

V piatej kapitole si predstavíme Kohonenové mapy, ktoré sú jedným z hlavným predstaviteľom neurónových sieti typu SOM – Self-Organizing Map. Ďalej si v tejto kapitole predstavíme dve varianty Kohonenových máp s adaptívnou topológiou. Konkrétne Kohonenové mapy s rastúcou mriežkou a model rastúcich neurónových plynov.

V šiestej kapitole si predstavíme jeden z najmladších modelov neurónových sieti, a to RBF (Radial Basis Function) sieť. Podrobne sa zameriame na architektúru týchto sieti a uvedieme význam RBF jednotiek. Uvedieme si viacero algoritmov učenia pre tento typ neurónových sieti ako napríklad hybridné učenie. Predstavíme si algoritmus GGAP-RBF (A Generalized Growing and Pruning), ktorý umožňuje v priebehu učenia siete meniť počet RBF jednotiek. Uvedieme algoritmus kontextového klastrovania, ktorý pomáha riešiť problém mnohorozmernosti vstupných dát. Ďalej spomenieme citlivostnú analýzu, ktorá sa využíva pri samotnom návrhu architektúry RBF sieti.

V siedmej kapitole prevedieme testovanie niektorých spomenutých algoritmov. V prvej časti budeme testovať tieto algoritmy na umelo vytvorených dátach. V druhej časti tejto kapitoly budeme pracovať s reálnymi dátami. Na základe objemu uskutočnených obchodných transakcií medzi štátmi budeme skúmať prepojenie ekonomík jednotlivých štátov sveta.

V ôsmej kapitole si predstavíme program Klastrovacie techniky, ktorý je súčasťou tejto diplomovej práce. Tento program umožňuje testovanie uvedených klastrovacích metód.

V deviatej kapitole nájdeme zhrnutie tejto diplomovej práce. Zrekapitulujeme témy, ktorým sme sa venovali.

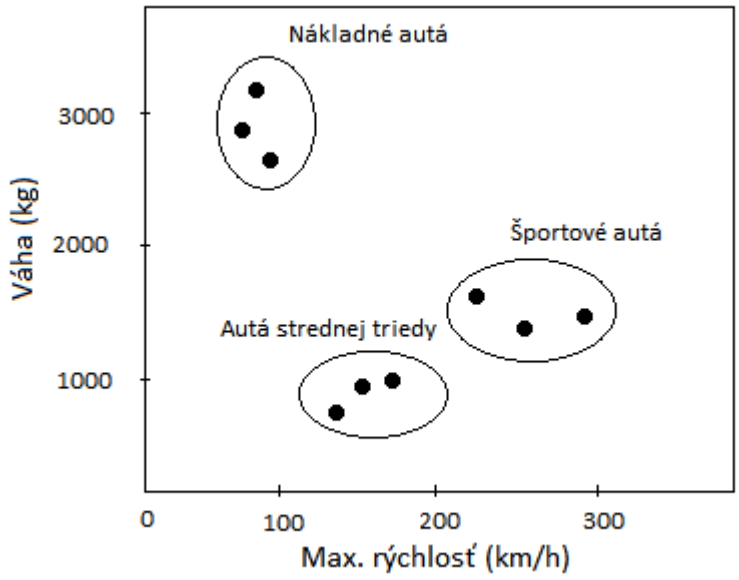
Kapitolu číslo desať tvorí zoznam literatúry, z ktorej bolo čerpané pri vypracovávaní tejto diplomovej práce.

2. Klastrovanie

Pod pojmom klastrovanie rozumejme proces zaraďovania konkrétnych, či abstraktných objektov do navzájom disjunktných skupín (nazývaných aj ako klastre) na základe pozorovania ich vlastnosti. Získané klastre nám pomôžu objaviť vnútornú štruktúru medzi spracovávanými objektmi.

Klastrom budeme nazývať skupinu objektov, ktoré sú si navzájom čo najviac podobné a zároveň sa tieto objekty čo najviac odlišujú od objektov umiestnených vo zvyšných klastroch.

Po klastrovacích algoritmoch požadujeme, aby výsledné rozdelenie objektov do klastrov bolo čo najviac jednoznačné. Teda, aby získane klastre boli jasne ohraničené a vzájomne sa neprekrývali. Existujú však aj algoritmy, ktoré povoľujú vzájomné sa prekrývanie jednotlivých klastrov, príkladom je Fuzzy c-means algoritmus (4).



Obrázok č. 1: Jednoduchý príklad klastrovania. Jednotlivé klastre sú tvorené rôznymi kategóriami vozidiel.

V dnešnej dobe je už známe množstvo klastrovacích algoritmov, niektoré najznámejšie z nich si predstavíme v ďalších kapitolách tejto práce. Avšak v tejto práci sa zameriame len na metódy typu učenia bez učiteľa (unsupervised learning), teda nemáme žiadnu informáciu o existencii a počiatočnom rozmiestnení skupín.

Rozdeľovanie objektov do klastrov sa riadi podľa istých kritérií, ktoré nám určujú kvalitu tohto rozdelenia. Keďže v oblasti dobývania znalostí pracujeme so vstupnými dátami, ktoré sú reprezentované pomocou vektorov, naším kritériom podobnosti bude vzdialenosť dvoch vektorov v priestore. Čím bude vzdialenosť vektorov menšia, tým sú si viac podobné, a naopak, čím je ich vzdialenosť väčšia, tým viac sa od seba odlišujú. Vzdialenosť dvoch vektorov môžeme počítať pomocou niektorej z nižšie uvedených metrík, pričom ich použitie je závislé od konkrétneho typu zadaného problému.

Nech $\vec{p}, \vec{q}$ sú dva n-rozmerné vstupné vektory $p = [p_1, p_2, \dots, p_n]$, $\vec{q} = [q_1, q_2, \dots, q_n]$. Vzdialenosť týchto dvoch vektorov môžeme spočítať v jednotlivých metrikách pomocou nasledujúcich vzorcov:

- Euklidova vzdialenosť

$$d_E(\vec{p}, \vec{q}\,) = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2} \tag{2.1}$$

- Manhattanska vzdialenosť

$$d_M(\vec{p}, \vec{q}\,) = \sum_{i=1}^{n} |p_i - q_i| \tag{2.2}$$

- Kosínova vzdialenosť

$$d_S(\vec{p}, \vec{q}\,) = \frac{\sum_{i=1}^{n} p_i . q_i}{\sqrt{\sum_{i=1}^{n} {p_i}^2 . \sum_{i=1}^{n} {q_i}^2}} \tag{2.3}$$

- Čebyševova vzdialenosť

$$d_C(\vec{p}, \vec{q}) = max_{i=1,\dots,n} |p_i - q_i| \tag{2.4}$$

Problematika klastrovania je už v dnešnej dobe pomerne dobré preskúmaná a nachádza si široké uplatnenie v mnohých oblastiach. Napríklad v oblasti predaja (analýza nákupného košíka) sa vytvárajú zoznamy produktov, ktoré zákazníci väčšinou nakupujú v rámci jedného nákupu. Ďalej je snahou rozdeliť zákazníkov do skupín, pričom zákazníci v rámci jednej skupiny sa zaujímajú o podobný sortiment produktov, prípadne nakupujú v približne rovnakom čase. Takéto informácie sú dôležité z marketingového pohľadu a umožňujú vytvárať atraktívnejšie ponuky pre zákazníkov.

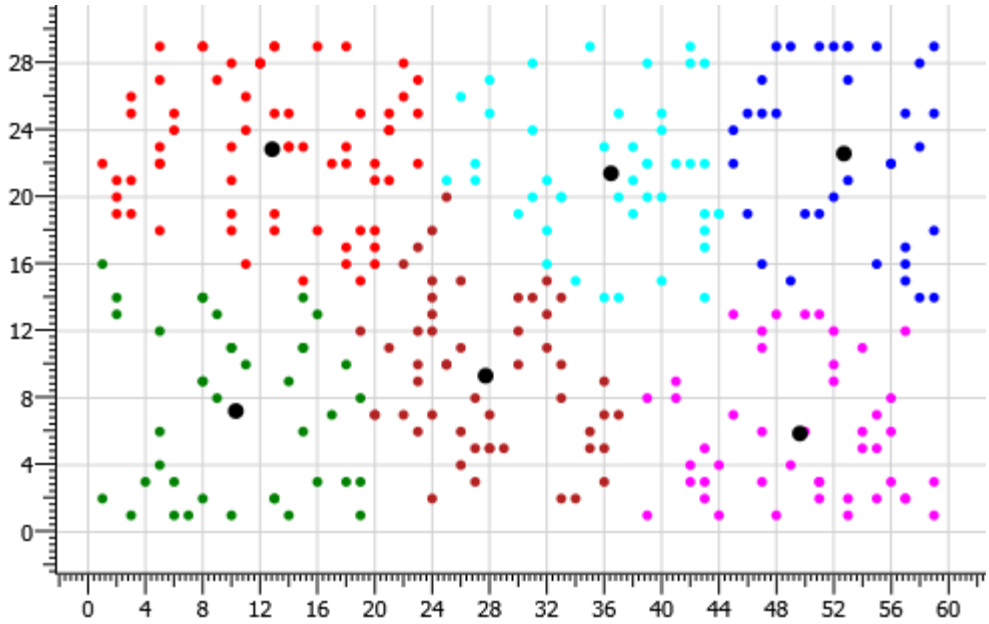
Ďalšia oblasť, ktorá sa nás takmer bezprostredne týka je poisťovníctvo. V oblasti poisťovníctva rozdeľujeme zákazníkov do skupín podľa toho aká miera rizika je s nimi spojená. Rizikovosť jednotlivých zákazníkov sa posudzuje podľa rôznych faktorov ako napríklad výška príjmu a typ zamestnania.

V závere tejto diplomovej práce pomocou klastrovania sa budeme snažiť rozdeliť štáty sveta do skupín podľa určitých ukazateľov stavu ich ekonomiky. Skutočne nie je ťažké nájsť využitie klastrovania v rôznych oblastiach.

3. K-means clustering (K-priemerová metóda)

Tento klastrovací algoritmus (R.O. Duda a P.E. Hart, 1973) rozdeľuje množinu vstupných vektorov do K navzájom disjunktných podmnožín (zhlukov). Jedná sa o pomerne jednoduchý algoritmus, vďaka čomu je veľmi často využívaný.

Nech $X = \{\vec{x}_i,\ \vec{x}_i \in R^n, i = 1, \dots, P\}$ je množina P vstupných vektorov, ktoré chceme rozdeliť do K zhlukov $v_k, k = 1, \dots, K$. Každý zhluk v_k je reprezentovaný vektorom $\vec{c}_k$, ktorý je umiestnený do ťažiska zhluku v_k. Vektory $\vec{c}_k, k = 1, \dots, K$ budeme nazývať ako centroidy príslušných zhlukov. Predpokladá sa $k \ll P$.



Obrázok č. 2: Ukážka rozdelenia vstupných dát do 6 klastrov pomocou K-means algoritmu. Centroidy sú reprezentované čiernymi bodmi. Jednotlivé klastre sú farebne odlíšené.

3.1. Chybová funkcia pre K-means algoritmus

Cieľom tohto algoritmu je minimalizovať hodnotu chybovej funkcie, teda minimalizovať súčet vzdialenosti medzi jednotlivými vstupnými vektormi a príslušnými centroidami.

$$E_{KM} = \sum_{k=1}^{K} \sum_{\vec{x} \in v_k} \|\vec{x} - \vec{c}_k\|^2 \tag{3.1}$$

$\|\vec{x} - \vec{c}_k\|^2$ určuje vzdialenosť medzi vektorom $\vec{x}$ a centroidom $\vec{c}_k$v zvolenej metrike. Hodnota tejto funkcie klesá s postupným pridávaním ďalších zhlukov. Tento pokles však po prekročení optimálneho počtu zhlukov začína byť len nepatrný.

Ako je uvedené v ([29], [28], [42]) na začiatku algoritmu užívateľ určí požadovaný počet zhlukov K. Vo vstupnom priestore náhodne vygenerujeme K vektorov a tie prehlásime za centroidy $\vec{c}_k, k = 1, \dots, K$. Pre každý vstupný vektor $\vec{x}_i \in X$ nájdeme centroid $\vec{c}_k$, ktorý je najbližšie k tomuto vektoru podľa vzťahu:

$$c_k = \arg\min_k \|\vec{x}_i - \vec{c}_k\|^2, \quad k \in \{1, .., K\} \tag{3.2}$$

Následne tento vstupný vektor $\vec{x}_i$ zaradíme do zhluku v_k reprezentovaného centroidom $\vec{c}_k$. Po zaradení všetkých vstupných vektorov pristúpime k prepočítaniu pozícií všetkých centroidov, a to v zmysle, že centroid $\vec{c}_k$ posunieme do ťažiska príslušného zhluku v_k. Tento postup opakujeme dovtedy, pokiaľ sa menia pozície centroidov.

3.2. Algoritmus K-means clustering

1. Inicializácia:
   Náhodne vygeneruj K navzájom rôznych centroidov.

2. Priradenie do zhluku:
   Nech $\vec{c}_k$ reprezentuje centroid, ktorý je najbližšie k vstupnému vektoru $\vec{x}_i$. Pre každý vstupný vektor $\vec{x}_i$, $i = 1, \dots, N$ urči najbližší centroid $\vec{c}_k$ a priraď tento vektor $\vec{x}_i$ do príslušného zhluku v_k. Index najbližšieho centroidu $\vec{c}_k$ k vektoru $\vec{x}_i$ spočítaj podľa predpisu:

$$c_k = \arg\min_k \|\vec{x}_i - \vec{c}_k\|^2, \quad k \in \{1, .., K\} \tag{3.3}$$

3. Aktualizácia pozícií centroidov:
   Vypočítaj nové pozície centroidov podľa predpisu:

$$\vec{c}_k = \frac{1}{n_k} \sum_{\vec{x} \in v_k} \vec{x}, \quad k = 1, \ldots, K \tag{3.4}$$

   n_k určuje počet vektorov, ktoré sú už priradené do zhluku s centroidom $\vec{c}_k$.

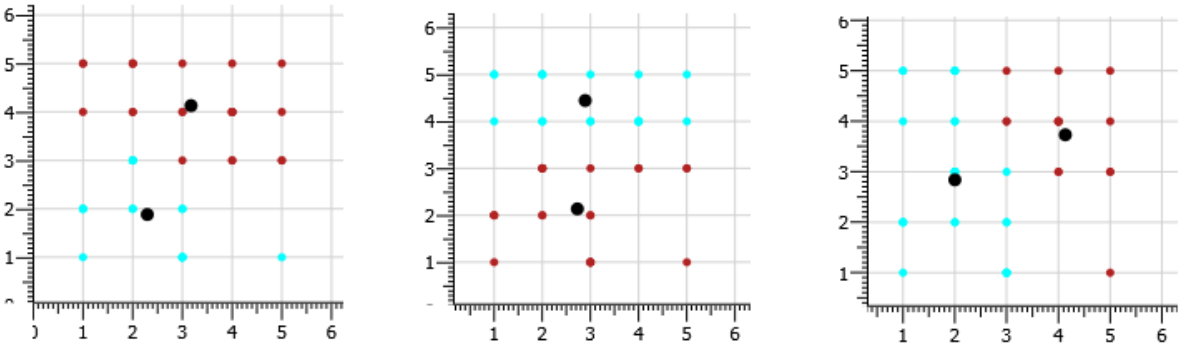
4. Ukončenie algoritmu:
   Pokračuj na krok algoritmu číslo 2. Opakuj pokiaľ sú pozorované zmeny jednotlivých pozícií centroidov.

Inicializačný krok tohto algoritmu možno nasledovne pozmeniť. Generovanie náhodných umiestnení centroidov jednotlivých zhlukov úplne vypustíme a za centroidy zvolíme priamo niektoré zo vstupných vektorov.

3.3. Vlastnosti K-means algoritmu

Algoritmus jednoznačne určí príslušnosť každého vstupného vektoru ku konkrétnemu zhluku, nie je teda možné, aby jeden vektor zároveň patril do viacerých zhlukov.

Medzi hlavné výhody tohto algoritmu patrí jeho jednoduchosť a rýchlosť. Čo nám umožňuje pracovať s väčším objemom dát, a tiež opakovane spúšťať tento algoritmus. Opakované spustenie algoritmu na rovnakých dátach nám však nezaručuje, že dospejeme k rovnakým výsledkom. Obrázok č. 3 ilustruje rôzne výsledky K-means algoritmu na rovnakých vstupných dátach.



Obrázok č. 3: Opakované spustenie K-means algoritmu na rovnaných dátach môže viesť k rozličnému rozloženiu klastrov vo vstupnom priestore. Čierne body predstavujú umiestnenie jednotlivých centroidov príslušných klastrov.

Nevýhodou K-means algoritmu určite je, že často nenájde optimálne riešenie, pretože sa zastaví v nejakom lokálnom optime (avšak často už po niekoľkých iteráciách). Toto môžeme čiastočne odstrániť viacnásobným spustením algoritmu na rovnakých dátach, avšak s iným náhodným rozložením centroidov.

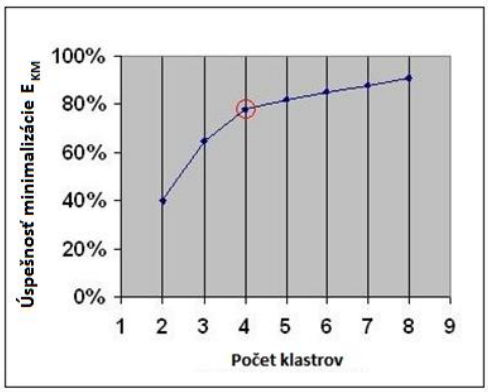
Výkonnosť tohto algoritmu veľmi závisí na inicializačnej fáze. Keďže užívateľ musí dopredu určiť počet klastrov, je potrebné odhadnúť optimálny počet zhlukov a optimálne rozmiestnenie ich centroidov vo vstupnom priestore. Pri zvolení menšieho počtu zhlukov niektoré zhluky nebudú objavené, ale budú zahrnuté v iných zhlukoch. Pri voľbe väčšieho počtu zhlukov by sa pri optimálnom rozložení centroidov mali objaviť všetky dostupné zhluky plus naviac nejaké zhluky obsahujúce len centroidy.

Veľmi dôležitou otázkou teda je ako určiť optimálne rozloženie centroidov. Jednou z možností je experimentovanie, čiže viacnásobné spustenie algoritmu vždy s iným náhodným rozmiestnením centroidov a vyberie sa najlepší variant. Ďalšou možnosťou je postupné umiestňovanie centroidov. Prvý centroid umiestnim náhodne do priestoru, druhý centroid umiestnim na takú pozíciu, ktorá je čo najviac vzdialená od prvého centroidu. Podobne postupujem s umiestňovaním ďalších centroidov, teda j-tý centroid umiestnim tak, aby bol čo najvzdialenejší od $j-1$ predchádzajúcich centroidov.

Nie je známe všeobecné pravidlo, ktoré by nám jednoznačne určilo optimálny počet zhlukov. Ako je uvedené v ([50]), veľmi jednoduchým pravidlom je pravidlo hrubého odhadu, vyjadrené predpisom:

$$K \approx \sqrt{\frac{P}{2}}, P \text{ je počet vstupných vektorov} \tag{3.5}$$

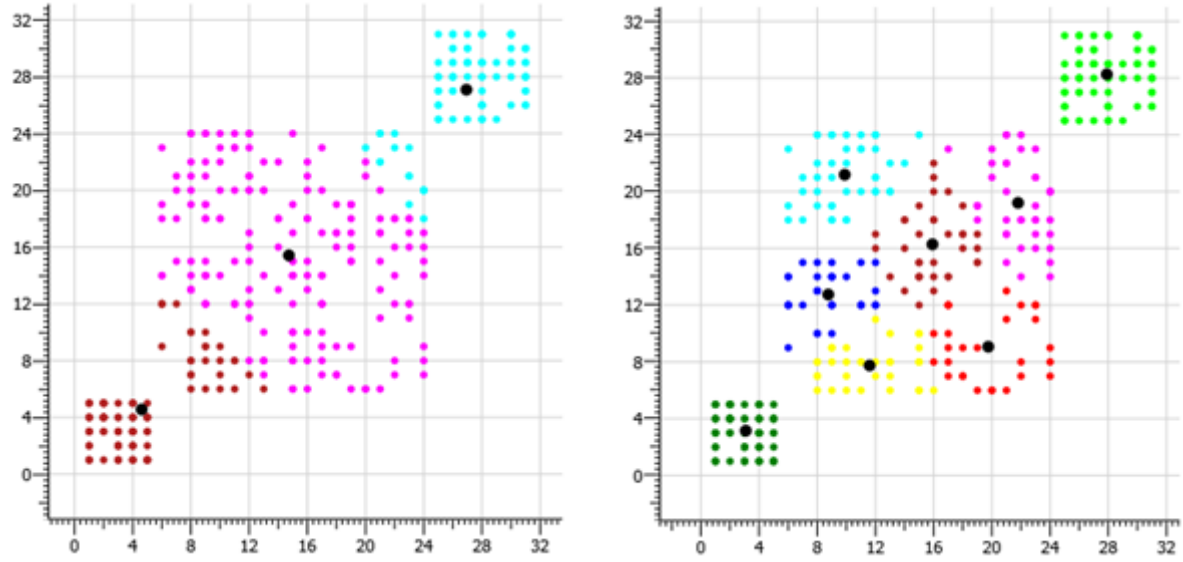
Ďalšou možnosťou je uhlové kritérium, v anglickej literatúre označované ako Elbow criterion. Táto metóda funguje v princípe nasledovne. Postupným zväčšovaním počtu klastrov sa zväčšuje množstvo získaných informácií o vstupných dátach a zároveň sa týmto zmenšuje hodnota chybovej funkcie. Tato zmena chybovej funkcie je pomerne značná, kým sa nedosiahneme optimálny počet klastrov. S ďalším nárastom klastrov však táto zmena nastáva stále pomalšie. My chceme určiť taký počet klastrov, že pridanie ďalšieho klastra už neprinesie výrazne väčšie množstvo informácií o vstupných dátach. Tento počet možno určiť z grafu, aj keď nie vždy jednoznačne. Vytvoríme si graf závislosti úspešnosti minimalizácie chybovej funkcie (3.1) na počte klastrov a hľadáme taký bod grafu, v ktorom je najväčší zlom na krivke (najväčšia uhlová zmena).



Obrázok č. 4: Uhlové kritérium vyjadruje závislosť úspešnosti minimalizácie chybovej funkcie na počte klastrov

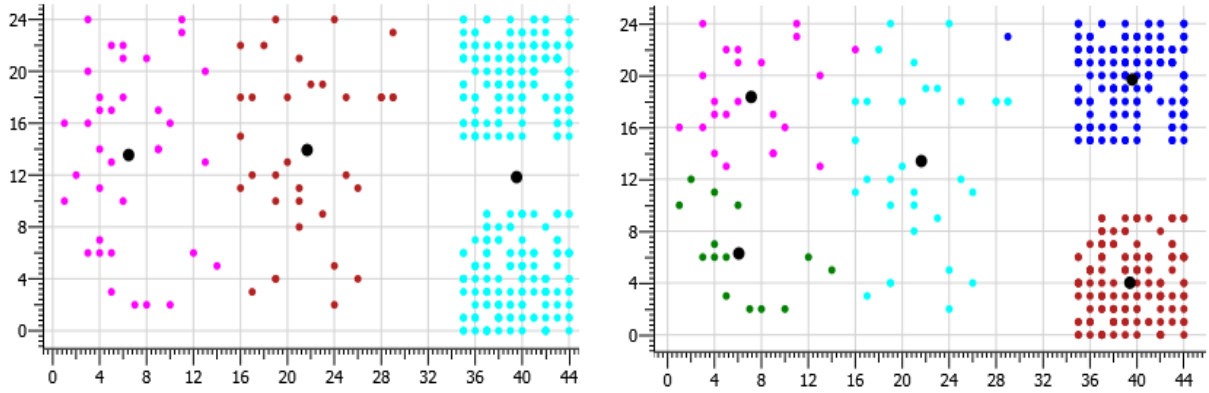
K-means algoritmus vytvára kompaktné klastre guľového tvaru. Bohužiaľ však existujú vstupné dáta, pre ktoré tento algortimus nenájde správne zhluky. Možným riešením môže byť zväčšenie počtu klastrov a ich následne zlučovanie, prípadne použiť inú vhodnejšiu klastrovaciu metódu. Nasledujúce obrázky 5 - 7 znázorňujú niektoré prípady vstupných dát, na ktorých použitie K-means algortimu nemusí viesť k správnemu rozdeleniu vstupných dát do klastrov.

Problémove môžu byť dáta, ktoré vytvárajú klastre rozličnej veľkosti. V takomto prípade môže dôjsť k nesprávnemu zaradeniu vstupných vektorov z klastrov väčších rozmerov do klastrov menších rozmerov. Obrázok č. 5 zachytáva práve takýto prípad.



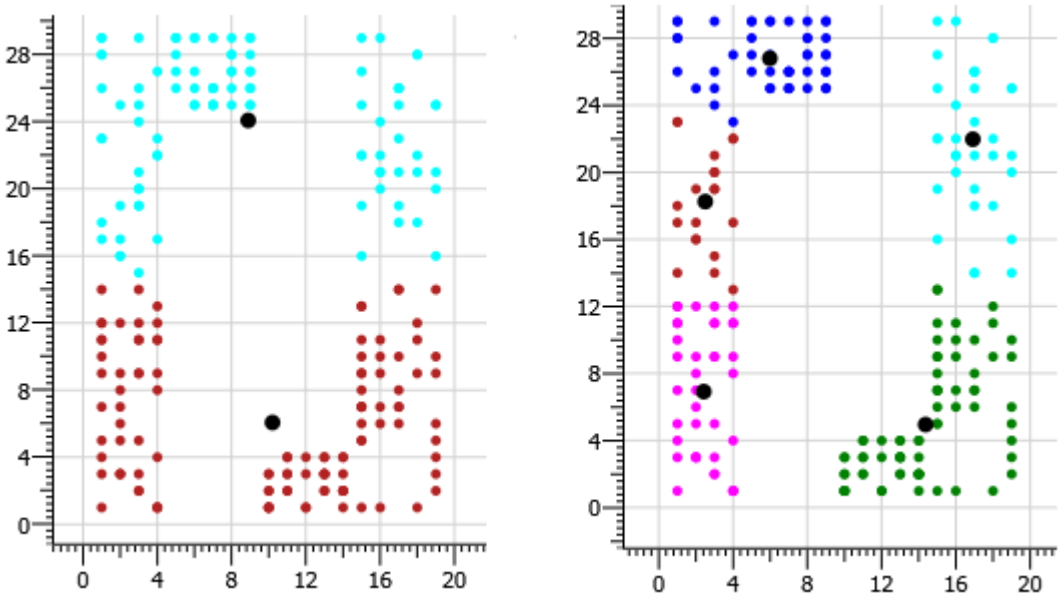
Obrázok č. 5. Problémy K-means algoritmu s klastrami rôznej veľkosti. Vľavo vidíme nesprávne rozdelenie vstupných dát do 3 klastrov. Vpravo vidíme rozdelenie vstupných dát do 8 klastrov, pričom klastre menšej veľkosti sú už určené správne.

Problémy s rozdelením vstupných dát do klastrov sa môžu vyskytnúť aj pri klastroch rôznej hustoty. Obrázok č. 6 zachytáva tento prípad.



Obrázok č. 6: Problémy K-means algoritmu s klastrami rôznej hustoty. Vľavo vidíme nesprávne rozdelenie vstupných dát do 3 klastrov. Vpravo vidíme rozdelenie vstupných dát do 5 klastrov, pričom klastre väčšej hustoty sú už určené správne.

Ďalšie problémy s rozdelením vstupných dát do klastrov sa často vyskytujú, ak chceme rozdeliť vstupné dáta do klastrov neguľových tvarov. Obrázok č. 7 zachytáva takýto prípad.



Obrázok č. 7: Problémy K-means algoritmu s klastrami rôznych tvarov. Vľavo vidíme nesprávne rozdelenie vstupných dát do 2 klastrov. Vpravo vidíme rozdelenie vstupných dát do 5 klastrov, pričom jednotlivé klastre už neobsahujú vstupné vzory z inej množiny.

4. FCM algoritmus

Fuzzy c-means (FCM) je klastrovacia metóda podobná K-means algoritmu (J. C. Dunn, 1973, Bezdek, 1981). Rozdeľuje množinu vstupných vektorov do K zhlukov. Avšak zásadným rozdielom oproti spomínanému algoritmu je, že vstupný vektor môže patriť do dvoch alebo viacerých klastrov zároveň. Inak povedané, zhluky už nemusia tvoriť navzájom disjunktné podmnožiny.

Nech $X = \{\vec{x}_i,\ \vec{x}_i \in R^n, i = 1, \dots, P\}$ je množina P vstupných vektorov, ktoré chceme rozdeliť do K zhlukov $v_k,\ k = 1, \dots, K$. Zhluk v_k je reprezentovaný centroidom $\vec{c}_k, k = 1, \dots, K$. Predpokladá sa $k \ll P$. Keďže vstupné vektory môžu patriť súčasne do viacerých zhlukov, má každý vstupný vektor $\vec{x}_i, i = 1, \dots, P$ priradený stupeň členstva u_{ij} k j-tému zhluku, pričom $0 \leq u_{ij} \leq 1$. Stupeň príslušnosti vstupných vektorov do jednotlivých zhlukov je určený členskou maticou $U = \left(u_{ij}\right)_{1 \leq i \leq P, 1 \leq j \leq K}$. Počet riadkov matice je určený počtom vstupných vektorov P a počet stĺpcov je určený počtom klastrov K. Parameter m je fuzzyfikačný parameter a platí $1 \leq m < \infty$.

Pre každý vstupný vektor $\vec{x}_i, i = 1, \dots, P$ platí, že súčet všetkých príslušných hodnôt u_{ij} k tomuto vektoru $\vec{x}_i$ cez všetky klastre je rovný jednej, $\forall i | \sum_{j=1}^{K} u_{ij} = 1$. Pre každý zhluk $v_k, k = 1, \dots, K$ platí, že nie je prázdny ani plný, $\forall j | 0 < \sum_{i=1}^{P} u_{ij} < P$.

4.1. Chybová funkcia pre FCM algoritmus

Cieľom tohto algoritmu je minimalizovať hodnotu chybovej funkcie, teda minimalizovať vzdialenosti medzi jednotlivými vstupnými vektormi a príslušnými centroidami.

$$E_{FCM} = \sum_{i=1}^{P} \sum_{k=1}^{K} u_{ik}^{m} \|\vec{x}_i - \vec{c}_k\|^2, \qquad 1 \leq m < \infty, \tag{4.1}$$

$\|\vec{x}_i - \vec{c}_k\|^2$ určuje vzdialenosť medzi vektorom $\vec{x}_i$ a centroidom $\vec{c}_k$ v zvolenej metrike.

Na začiatku algoritmu užívateľ zvolí hodnoty vstupných parametrov. To znamená požadovaný počet klastrov K, hodnotu fuzzyfikačného parametra m a hodnotu ukončovacieho kritéria ε. Náhodne vygenerujeme členskú maticu $U^{(0)}$. Kým nie je splnená ukončovacia podmienka, opakovane prepočítavame pozície centroidov a hodnoty prvkov členskej matice podľa nasledujúcich vzťahov:

$$\vec{c}_j(t) = \frac{\sum_{i=1}^{P} u_{ij}^m(t)\vec{x}_i}{\sum_{i=1}^{P} u_{ij}^m(t)} \quad j = 1, \dots, K \tag{4.2}$$

$$u_{ij}(t+1) = \frac{\left(\frac{1}{\left\|\vec{x}_i - \vec{c}_j(t)\right\|^2}\right)^{\frac{1}{m-1}}}{\sum_{k=1}^{K}\left(\frac{1}{\|\vec{x}_i - \vec{c}_k(t)\|^2}\right)^{\frac{1}{m-1}}} \quad i = 1, \dots P, j = 1, \dots, K \tag{4.3}$$

Podmienkou k ukončeniu algoritmu je, že maximálna hodnota zmien pozícií centroidov jednotlivých zhlukov medzi jednotlivými iteráciami algoritmu klesne aspoň na hodnotu ukončovacieho kritéria ε ([35],[3]).

4.2. Fuzzy c-means algoritmus

1. Inicializácia:
   Zvoľ parametre K, m, ε a t. Náhodne zvoľ maticu $U^{(0)}$, pre ktorej prvky platí $0 \leq u_{ij} \leq 1$.

2. Aktualizácia pozícií centroidov:
   Urč nové centrá fuzzy zhlukov podľa predpisu:

$$\vec{c}_j(t) = \frac{\sum_{i=1}^{P} u_{ij}^m(t)\vec{x}_i}{\sum_{i=1}^{P} u_{ij}^m(t)} \quad j = 1, \dots, K \tag{4.4}$$

3. Aktualizácia členskej matice:
   Vypočítaj novú členskú maticu $U^{(t+1)}$ podľa predpisu:

$$u_{ij}(t+1) = \frac{\left(\frac{1}{\left\|\vec{x}_i - \vec{c}_j(t)\right\|^2}\right)^{\frac{1}{m-1}}}{\sum_{k=1}^{K}\left(\frac{1}{\|\vec{x}_i - \vec{c}_k(t)\|^2}\right)^{\frac{1}{m-1}}} \quad i = 1, \dots P, j = 1, \dots, K \tag{4.5}$$

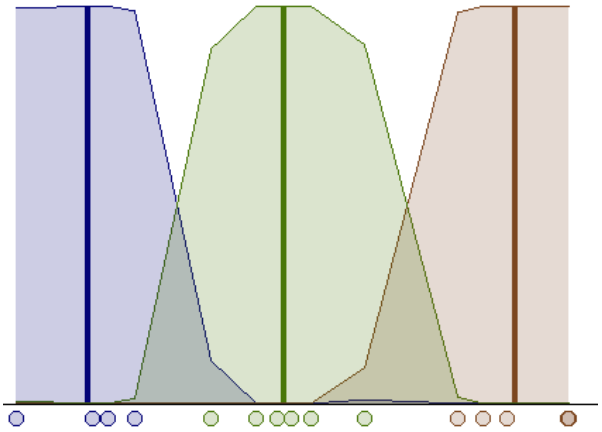
4. Ukončenie algoritmu:
   Vyhodnoť ukončovaciu podmienku.

$$\Delta = \left\|U^{(t+1)} - U^{(t)}\right\| = max_{ij}\left|u_{ij}(t+1) - u_{ij}(t)\right| \tag{4.6}$$

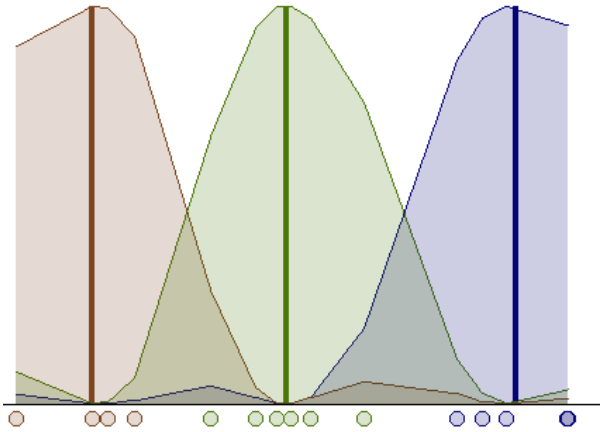
   Ak $\Delta > \varepsilon$, tak nastav $t = t + 1$ a prejdi na krok číslo 2.
   Ak $\Delta \leq \varepsilon$, tak ukonči algoritmus.
   Hodnotu ukončovacie kritéria volíme z intervalu $0 \leq \varepsilon \leq 1$.

Fuzzyfikačný parameter m má dôležitý vplyv na výkonnosť algoritmu. Určuje ako veľmi sa mení príslušnosť vstupných vektorov ku klastrom vzhľadom k ich vzdialenosti od centroidov jednotlivých zhlukov. Bežne sa v literatúre môžeme stretnúť s hodnotou tohto parametra rovnou dva. Avšak pomocou experimentov ([35]), v ktorých sa hľadala optimálna kombinácia hodnoty fuzzyfikačného parametra a počtu klastrov, sa preukázalo, že optimálna hodnota m sa môže líšiť od spomínanej m = 2.0. Na rôznych množinách vstupných dát sa testovali hodnoty parametru $m = \{1.1, 1.5, 2.0, 2.2, 2.7\}$.

Pri hodnotách fuzzyfikačného parametra blízkych jednotke je príslušnosť vstupných vektorov do viacerých klastrov minimálna. Pri hodnote $m = 1$ algoritmus degraduje na K-means klastrovanie. Naopak pri vyšších hodnotách m vstupné vektory náležia súčasne do viacerých klastrov výraznejšou mierou. Ako vidieť na obrázkoch 8, 9 s narastajúcou hodnotou fuzzyfikačného parametra m nadobúdajú oblasti príslušnosti vstupných vektorov do jednotlivých klastrov ostrejšie tvary.



Obrázok č. 8: Rozdelenie vstupných dát do 3 klastrov pomocou FCM algoritmu, hodnota fuzzyfikačného parametra $m = 1.5$ ([49]).



Obrázok č. 9: Rozdelenie vstupných dát do 3 klastrov pomocou FCM algoritmu, hodnota fuzzyfikačného parametra $m = 2$ ([49]).

4.3. Kritéria pre validitu klastrov

Pri mnohorozmerných vstupných dátach býva ťažké vhodne vizualizovať výsledné klastrovanie. Často sa stáva, že od pohľadu nie je možné posúdiť, ako kvalitné rozdelenie dát do klastrov sme získali. Aby sme boli schopný rozhodnúť, ktoré z výsledných klastrovaní je to najkvalitnejšie, potrebujeme nejaký mechanizmus, ktorý nám umožní ich vzájomné porovnanie. Z tohto dôvodu bolo definovaných viacero kritérií validity klastrov hodnotiacich kvalitu rozdelenia vstupných dát do klastrov . Pri fuzzy zhlukovaní sa najčastejšie využívajú nasledovné validačné kritéria ([45]):

Koeficient členskej funkcie

Toto kritérium ([6],[43]) definujeme predpisom:

$$F_K(U) = \frac{\sum_{i=1}^{P}\sum_{j=1}^{K} u_{ij}^2}{P} \tag{4.7}$$

Toto kritérium naznačuje priemerný počet zdieľaných členstiev medzi dvojicami fuzzy podmnožín v U. Koeficient členskej funkcie môže nadobúdať hodnoty z intervalu

$\frac{1}{K} \leq F_K(U) \leq 1$. Optimálne rozdelenie do klastrov je také, keď funkcia $F_K(U)$ nadobúda maximum. Zároveň vieme pomocou tohto koeficientu určiť optimálny počet klastrov, pri ktorom klastrovanie dosiahne najvyššej kvality. Optimálny počet klastrov dostaneme vyriešením vzťahu:

$$\max_{2\leq K\leq P-1}\{\max_U[F_K(U)]\} \tag{4.8}$$

Entropia členskej funkcie

J.Bezdek navrhol toto kritérium ([4], [5], [6]) a je definované predpisom:

$$H_K(U) = -\frac{\sum_{i=1}^{P}\sum_{j=1}^{K} u_{ij} log_a(u_{ij})}{P} \qquad a \in (1,\infty) \tag{4.9}$$

Entropia členskej funkcie popisuje mieru neurčitosti v rozdelení do fuzzy klastrov, pričom môže nadobúdať hodnoty z intervalu $0 \leq H_K \leq log_a K$. Keďže pri klastrovaní chceme túto mieru neurčitosti minimalizovať, tak optimálne rozdelenie do klastrov je také, keď funkcia $H_K(U)$ nadobúda minimum. Optimálny počet klastrov v tomto prípade vypočítame vyriešením vzťahu:

$$\min_{2\leq K\leq P-1}\{\min_U[H_K(U)]\} \tag{4.10}$$

Entropia $H_K(U)$ je viac citlivá ako koeficient $F_K(U)$ na lokálne zmeny. Medzi týmito kritériami platia nasledujúce vzťahy:

$$F_K = 1 \Leftrightarrow H_K = 0 \tag{4.11}$$

$$F_K = \frac{1}{K} \Leftrightarrow H_K = log_a K \tag{4.12}$$

$$\frac{1}{K} \leq F_K \leq 1, \quad 0 \leq H_K \leq log_a K \tag{4.13}$$

Nevýhodou zmienených kritérií je ich monotónna závislosť na počte klastrov K. Koeficient členskej funkcie môžeme modifikovať, tak aby došlo k redukcii tejto závislosti ([2]). Definujme modifikovanú verziu $F'_K(U)$ nasledujúcim spôsobom:

$$F'_K(U) = 1 - \frac{K}{K-1}\big(1 - F_K(U)\big) \tag{4.14}$$

Pre hodnoty, ktoré môže toto modifikované kritérium nadobudnúť platí $0 \leq F'_K(U) \leq 1$. Optimálny počet klastrov získame vyriešením vzťahu:

$$\max_{2\leq K\leq P-1}\{\max_U[F'_K(U)]\} \tag{4.15}$$

Windhamov proporčný index

Toto kritérium navrhol Windham ([46]) a definujeme ho predpisom:

$$W_K(U) = -\sum_{i=1}^{P} ln\left[\sum_{j=1}^{\lfloor \mu_i^{-1} \rfloor} (-1)^{j+1}\binom{K}{j}(1 - j.\mu_i)^{K-1}\right] \tag{4.16}$$
$$\mu_i = \max_{1\leq k\leq K}\{u_{ki}\}$$

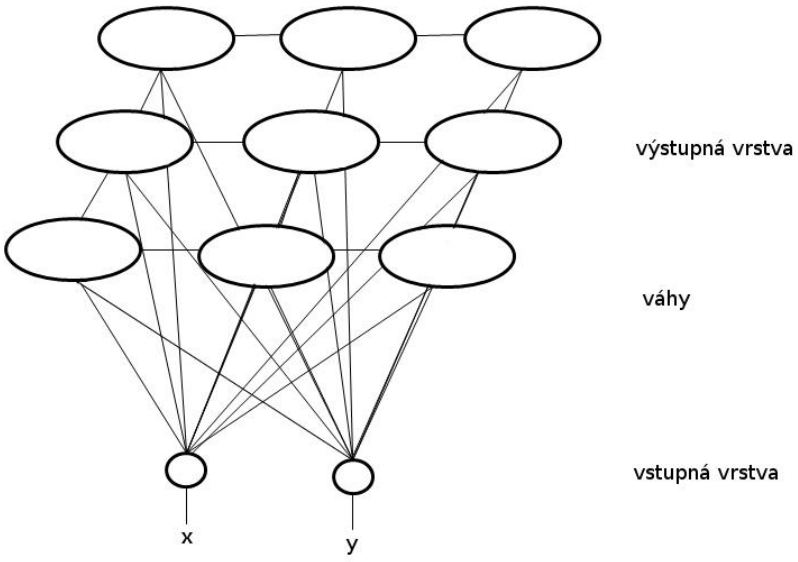
Windham tvrdí, že je prirodzené zamerať sa na maximálne hodnoty prvkov uvedených v stĺpcoch členskej matice U. Veľké hodnoty maximálnych prvkov totižto predznamenávajú jasnú príslušnosť niektorých vstupných vektorov do nejakých zhlukov. To nám signalizuje identifikáciu určitej vnútornej štruktúry vo vstupnom priestore. Optimálne rozdelenie do klastrov je také, keď funkcia $W_K(U)$ nadobúda maximum. Optimálny počet klastrov v tomto prípade vypočítame vyriešením vzťahu:

$$\max_{2\leq K\leq P-1}\{\max_U[W_K(U)]\} \tag{4.17}$$

5. Kohonenové mapy

Medzi základné typy neurónových sieti patria samo organizačné neurónové siete, tzv. SOM – Self-Organizing Map. Už ako názov napomína, jedná sa o siete, ktorých učiaci proces prebieha bez informácií o požadovaných výstupoch od učiteľa, tzv. Unsupervised learning. To znamená, že učiaci algoritmus nemá možnosť overiť správnosť výsledkov siete pre zadané trénovacie vzory. SOM sa využívajú na mapovanie vstupného príznakového priestoru, v ktorom sa hľadajú oblasti so vzormi, ktoré majú navzájom podobné vlastnosti a budú vo výsledku vytvárať zhluky. Popritom by sa malo zachovávať topologické usporiadanie vstupných vzorov.

Kohonenové mapy navrhnuté T. Kohonenom v roku 1982 sú jedným z hlavným predstaviteľom neurónových sieti typu SOM. Základná myšlienka pre ich vznik pochádza z neurobiológie, kde sa zistilo, že väčšina štruktúr neurónov v mozgu má lineárnu alebo rovinnú topografickú anatómiu napriek tomu, že zmyslové vnemy bývajú mnohorozmerné ([22]). Toto je dôležitá vlastnosť, ktorá sa preniesla aj do Kohonenových máp. Použitie Kohonenových máp je teda veľmi výhodné, ak chceme vizualizovať mnohorozmerné dáta do nízkodimenzionálneho priestoru.



Obrázok č. 10: Príklad Kohonenovej mapy s usporiadaním neurónov do mriežky so štvorcovým okolím a s dvoma vstupmi.

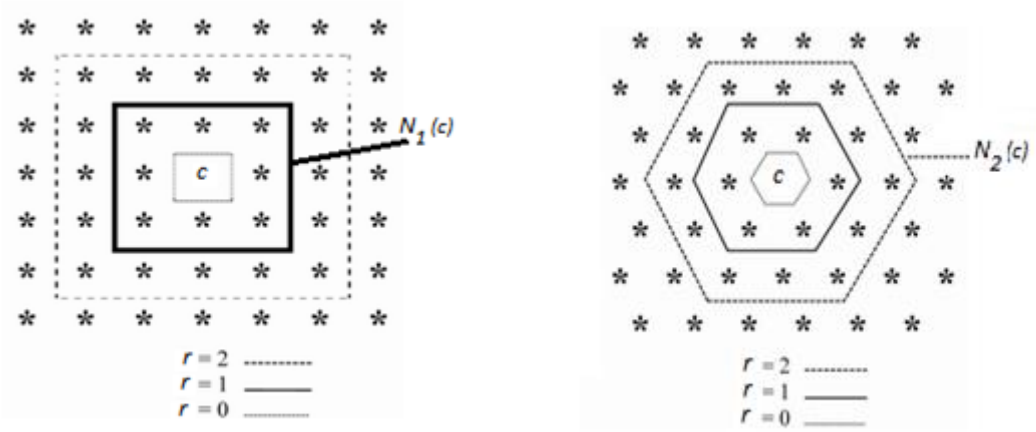
5.1.1. Topológia Kohonenovej mapy

Jedná sa o dvojvrstvovú neurónovú sieť (vstupná a výstupná vrstva), pričom pre každý vstupný neurón platí, že je spojený synaptickou váhou s každým neurónom vo výstupnej vrstve. Neuróny vo výstupnej vrstve sú usporiadané do určitej topologickej štruktúry. Najčastejšie sa používa dvojrozmerná mriežka, prípadne jednorozmerná rada. Voľba konkrétnej topologickej štruktúry je dôležitá v procese učenia, pretože nám zároveň určuje, ktoré neuróny spolu susedia. Pod pojmom okolie neurónu c budeme rozumieť množinu neurónov $N_r(c)$ spĺňajúcich podmienku, že ich topologická vzdialenosť na topologickej mriežke od neurónu c je menšia alebo rovná r.

$$N_r(c) = \{j, d(j,c) \leq r\} \tag{5.1}$$

Vzdialenosť neurónov $d(j,c)$ je práve závislá na topologickej štruktúre výstupných neurónov. Vypočítame ju ako počet hrán na najkratšej ceste z neurónu c do neurónu j. Veľkosť okolia sa v procese učenia s časom mení. Na začiatku algoritmu sa zvolí veľký polomer okolia r, napr. polovica veľkosti mriežky. Postupne sa veľkosť okolia s narastajúcim počtom krokov učenia zmenšuje. Na konci učiaceho procesu by malo okolie zahŕňať už len samotný neurón c. Tento pokles veľkosti okolia spôsobuje, že na predložený vstupný vektor sa postupne adaptuje stále menšie množstvo neurónov, čím sa zvyšuje presnosť výslednej Kohonenovej mapy.

V blízkosti okrajov mapy (rada, mriežka, ...) okolie neurónov nie je symetrické, z tohto dôvodu bývajú okrajové časti mierne kontrahované smerom do vnútra mapy. Toto môžeme odstrániť tak, že algoritmicky prepojíme odpovedajúce si okraje mapy do slučky.



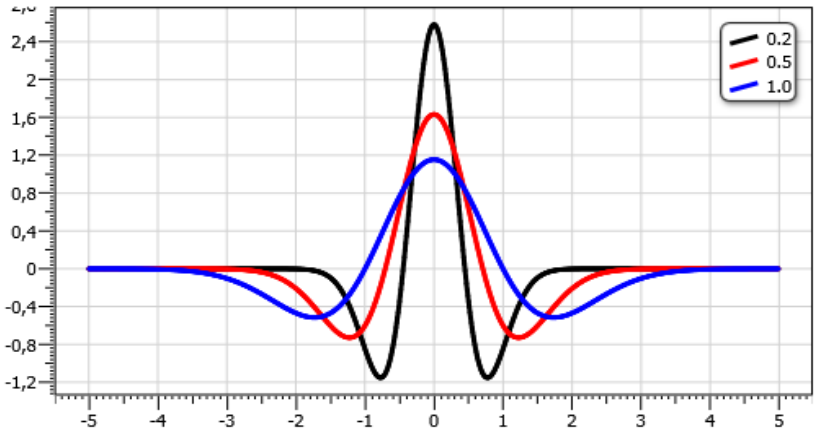
Obrázok č. 11: Príklad okolia neurónu c s hodnotu polomeru $r = \{1, 2\}$ vo štvorcovej a hexagonálnej topografickej štruktúre výstupných neurónov

5.2. Funkcia laterálnej interakcie

Určuje mieru a druh vzájomnej interakcie (inhibícia, excitácia) medzi neurónmi v topologickej mriežke v závislosti na topologickej vzdialenosti neurónov c a j. Obrázok č. 12 znázorňuje graf takejto funkcie, ktorá je typu mexického klobúka. Pre funkciu tohto typu platí, že laterálna interakcia má excitačný vplyv na blízke neuróny a v menšej miere inhibičný vplyv pre navzájom vzdialenejšie neuróny. Pre veľmi vzdialené neuróny má slabý excitačný vplyv, a to z dôvodu, aby sa z takýchto neurónov nestali nevyužité („mŕtve“) neuróny. Funkciu laterálnej interakcie budeme označovať $\phi(c,j)$ a je definovaná predpisom:

$$\phi(c,j) = \frac{2}{\sqrt{3\sigma}\pi^{\frac{1}{4}}}\left(1 - \frac{d(j,c)^2}{\sigma^2}\right) e^{-\frac{d(j,c)^2}{2\sigma^2}}, \tag{5.2}$$

kde σ je šírka funkcie.



Obrázok č. 12: Funkcia laterálnej interakcie typu mexického klobúka s rôznou hodnotou parametra šírky $\sigma, \sigma \in \{0.2, 0.5, 1.0\}$. Vyjadruje druh vzájomnej interakcie medzi neurónmi v topologickej mriežke v závislosti na topologickej vzdialenosti od víťazného neurónu.

Nech $X = \{\vec{x}_i, \vec{x}_i \in R^n, i = 1, \dots, P\}$ je množina P trénovacích vzorov. Parameter učenia η(t), $0 \leq \eta(t) \leq 1$ ovplyvňuje rýchlosť učenia siete. Na začiatku volíme vyššiu hodnotu blízku jednej, ktorá počas procesu učenia siete klesá k nule. Parameter učenia by mal spĺňovať požiadavky $\sum_{t=1}^{\infty} \eta(t) = \infty \wedge \sum_{t=1}^{\infty} \eta^2(t) < \infty$. Ďalej $\vec{w}_{ij}$, $0 \leq i \leq I-1, 0 \leq j \leq O-1$ určuje váhu synapsie zo vstupného neurónu i k výstupnému neurónu j, pričom I (input) určuje počet vstupných neurónov a O (output) určuje počet výstupných neurónov.

Na začiatku algoritmu učenia Kohonenovej mapy zvolíme počiatočne hodnoty parametrov, a to hodnotu parametra učenia $\eta(t)$, polomer r okolia neurónov vo výstupnej vrstve siete a váhy siete $\vec{w}_{ij}, 0 \leq i \leq I-1, 0 \leq j \leq O-1$ nastavíme na malé náhodné hodnoty. Náhodne vyberieme trénovací vzor $\vec{x} \in X$ a ten predložíme na vstup sieti. Tento vzor $\vec{x}$ porovnáme s váhovými vektormi $\vec{w}_j$ všetkých výstupných neurónov a vyberieme taký neurón c, ktorého váhový vektor je najbližšie (je najpodobnejší) trénovaciemu vzoru. Hovoríme, že tento neurón vyhral kompetíciu a označíme ho ako víťazný neurón c. Takýto princíp sa nazýva „víťaz berie všetko“ (“winner takes all”).

Ďalej chceme, aby víťazný neurón c a neuróny z jeho okolia $N_r(c)$ ešte lepšie reprezentovali predložený vzor $\vec{x}$. To dosiahneme posunutím týchto neurónov bližšie k predloženému vzoru $\vec{x}$. V tomto bode sa práve využije funkcia laterálnej interakcie (5.2), ktorá spôsobí, že neuróny blízko víťazného neurónu c sa posunú spolu s víťazným neurónom c bližšie k predloženému vzoru $\vec{x}$ (stanú sa citlivejšími na predložený vzor a vzory jemu podobné) a vzdialenejšie neuróny z okolia $N_r(c)$ sa naopak ešte viac vzdialia od víťazného neurónu c (ich reakcia bude utlmená). To, ktoré váhy sa budú aktualizovať záleží na veľkosti okolia víťazného neurónu $N_r(c)$.

S narastajúcim počtom iterácií klesá veľkosť okolia, a teda klesá aj počet neurónov, ktorých sa tento posun týka. Následne pokračujeme predložením ďalšieho trénovacieho vzoru $\vec{x} \in X$ na vstup sieti. Tento postup opakujeme, kým nedosiahneme požadovaný počet iterácií.

5.3. Algoritmus učenia Kohonenovej mapy

1. Inicializácia:
   Zvoľ náhodné malé hodnoty váh $\vec{w}_{ij}, 0 \leq i \leq I-1, 0 \leq j \leq O-1$ medzi I vstupnými a O výstupnými neurónmi. Ďalej zvoľ počiatočný polomer r okolia, parameter učenia $\eta(t)$ a funkciu laterálnej interakcie $\phi(c, j)$.
   Polomer okolia r volíme na začiatku veľký, napr. polovica veľkosti mriežky. Parameter učenia $\eta(t)$, $0 \leq \eta(t) \leq 1$ nastavíme na hodnotu blízko jednej.

2. Predloženie vzoru:
   Náhodne vyber trénovací vzor $\vec{x} \in X$ a ten predlož na vstup neurónovej sieti.

3. Výpočet vzdialenosti:
   Spočítaj vzdialenosti d_j medzi trénovacím vzorom $\vec{x}$ a váhovým vektorom $\vec{w}_j$ pre každý výstupný neurón $j, 0 \leq j \leq O-1$ podľa predpisu:

$$d_j = \sum_{i=0}^{P-1} \left(\vec{x}_i(t) - \vec{w}_j(t)\right)^2 \tag{5.3}$$

4. Výber víťazného neurónu:
   Nájdi najbližší výstupný neurón c, ktorý má minimálnu hodnotu d_j a označ ho ako víťaza.

$$c = \arg\min_j \left(d_j\right), \qquad 0 \leq j \leq O-1 \tag{5.4}$$

5. Aktualizácia váh:
   Aktualizuj váhy pre víťazný neurón c a pre všetky neuróny z okolia definovaného pomocou $N_c(r)$ podľa predpisu:

$$\vec{w}_j(t+1) = \begin{cases} \vec{w}_j(t) + \eta(t)\phi(c,j)\left(\vec{x}_i(t) - \vec{w}_j(t)\right), & j \in N_c(r) \\ \vec{w}_j(t), & j \notin N_c(r) \end{cases} \tag{5.5}$$

6. Ukončenie algoritmu:
   Ak bol dosiahnutý požadovaný počet iterácií, tak ukonči algoritmus.
   V opačnom prípade aktualizuj hodnoty parametra učenia $\eta(t)$ a polomeru r. Hodnotu parametra učenia aktualizuj podľa:

$$\eta(t) = 0.9\left(1 - \frac{t}{1000}\right) \tag{5.6}$$

   Veľkosť polomeru okolia by mala s narastajúcim časom t klesnúť až na nulu. Napríklad lineárne. Pokračuj na krok algoritmu číslo 2.

V piatom kroku algoritmu sa pomerne často za funkciu laterálnej interakcie (5.2) volí Gaussová funkcia so stredom v c, ktorá viac odpovedá biologickým interakciam. Aktualizácia váh v tomto prípade prebieha podľa predpisu:

$$\vec{w}_j(t+1) = \vec{w}_j(t) + \eta(t)\exp\left(-\frac{d(j,c)^2}{2\sigma^2(t)}\right)\left(\vec{x}_i(t) - \vec{w}_j(t)\right), \tag{5.7}$$

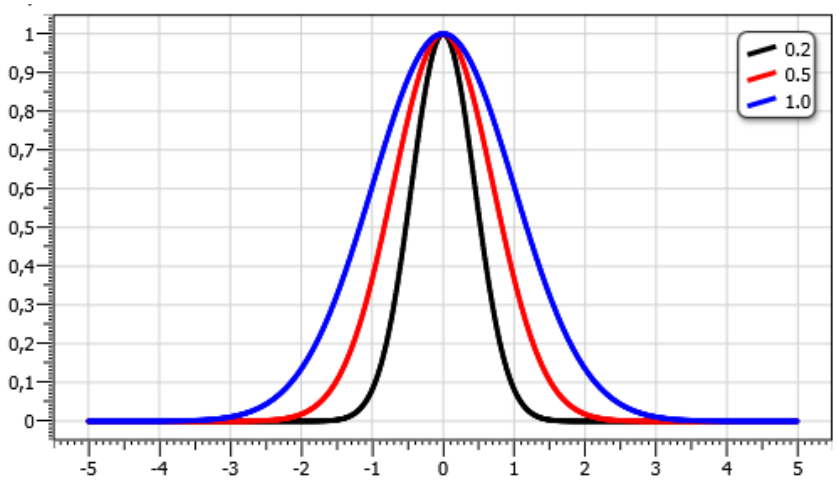
kde $\sigma(t)$ je šírka funkcie. A platí, že hodnoty parametrov $\sigma(t)$ a $\eta(t)$ sa s narastajúcim časom t zmenšujú. Časová náročnosť tohto riešenia je však vyššia, pretože v každom kroku prechádzame a aktualizujem všetky váhové vektory $\vec{w}_j$, $0 \leq j \leq O-1$ v sieti.

5.4. Gaussová funkcia

Gaussová funkcia je definovaná predpisom:

$$f(x) = a\exp\left(-\frac{(x-c)^2}{2\sigma^2}\right), \qquad r \in R, \sigma > 0 \tag{5.8}$$

Parameter c určuje pozíciu stredu Gaussovej funkcie, parameter σ určuje šírku Gaussovej funkcie. Parameter šírky σ ovplyvňuje strmosť tejto funkcie. Parameter a ovplyvňuje výšku vrcholu funkcie. Graf tejto funkcie pripomína tvar zvončeka.



Obrázok č. 13. Priebeh Gaussovej funkcie so stredom v bode 0 s rôznou hodnotou parametra šírky $\sigma, \sigma \in \{0.2, 0.5, 1.0\}$. Parameter $a = 1$. Za funkciu laterálnej interakcie sa často volí práve Gaussová funkcia.

Celý proces učenia Kohonenovej mapy sa dá rozdeliť na dve fázy, hrubé učenie a dolaďovanie ([22]).

Pod hrubým učením rozumejme napríklad prvých tisíc krokov učiaceho procesu. V tejto fáze majú parameter učenia $\eta(t)$ a veľkosť okolia $N_r(c)$ veľké hodnoty. Parameter učenia $\eta(t)$ začína s hodnotou blízkou jednotke a postupne klesá. Experimentálne sa

preukázalo, že nie je dôležité, či tento parameter klesá lineárne, exponenciálne, alebo iným spôsobom. Ako rozumná voľba pre tento parameter sa javí použiť už spomínaný predpis (5.6). Parameter učenia $\eta(t)$ by mal počas hrubého učenia klesnúť na hodnotu približne 0.02. V ďalšej fáze pokračuje pokles tohto parametru k nule.

Hodnota polomeru r okolia neurónov by sa mala počas fázy hrubého učenia meniť nasledovne. Na začiatku môžeme zvoliť za polomer r polovicu rozmeru mapy. Tento parameter môže lineárne klesať až na hodnotu jedna. V dolaďovacej fáze môžu byť ešte zo začiatku do okolia $N_r(c)$ zahrnuté najbližšie neuróny, pričom hodnota polomeru r klesne až na nulu a ďalej sa bude už upravovať len víťazný neurón c. Počas hrubého učenia dôjde k rozmiestneniu výstupných neurónov približne do oblastí, kde sa vyskytujú vstupné vzory. Pozície neurónov sa v ďalšej fáze mena už len pomerne málo.

Výsledná presnosť Kohonenovej mapy záleží na počte vykonaných iterácií učiaceho algoritmu, predovšetkým v záverečnej dolaďovacej fáze. Ta by mala byť primerane dlhá. V publikáciách ([22], [23]) sa uvádza, že počet iterácií by mal byť najmenej päťstokrát väčší ako je počet výstupných neurónov v sieti. Bežne sa necháva učiaci algoritmus vykonať až do 100 tisíc iterácií, ale vždy záleží na konkrétnom druhu úlohy. Napríklad pri úlohe rozpoznávania rečí 10 tisíc iterácií môže byť dostatočné množstvo. V prípade, že nemáme k dispozícii dostatočný počet trénovacích vzorov, musíme tieto vzory opakovane predkladať na vstup siete.

Uvedený algoritmus učenia Kohonenovej mapy (5.3) nemusí vždy fungovať podľa našich predstáv. Problém môže nastať v prípade, že jeden alebo malá skupina neurónov často víťazia v kompetícii. Takáto situácia môže vyzerať napríklad nasledovne. Vstupné dáta nech sú rozdelené do dvoch izolovaných od seba dostatočne vzdialených oblasti. Inicializačné rozmiestnenie výstupných neurónov sa volí náhodne, takže sa môže stať, že všetky tieto neuróny sa vyskytnú v jednej z týchto oblasti. Keď predložíme vzor z druhej oblasti, tento vzor pritiahne jeden neurón z prvej oblasti bližšie k sebe. Potom tento neurón už bude vždy víťaziť v kompetícii pre vstupy z druhej oblasti a táto oblasť bude reprezentovaná len jedným neurónom.

Odstrániť tento problém môžeme nasledujúcim spôsobom. Pre každý výstupný neurón si budeme pamätať počet jeho doposiaľ dosiahnutých víťazstiev v kompetícii. Ak v kompetícii zvíťazí neurón, ktorého počet víťazstiev je príliš veľký, vyradíme tento neurón na určitú dobu z kompetície a necháme zvíťaziť na miesto neho nejaký iný výstupný neurón. Tento postup sa bežne nazýva ako princíp svedomia.

5.5. Aplikácie Kohonenových máp

Aplikácií Kohonenových máp je veľmi veľa. Jednou z najznámejších aplikácií je rozpoznávanie hovorenej reči. Na vstup predložíme nejaký zvukový záznam a ako výstup dostaneme textový prepis tohto zvukového záznamu. Základným impulzom pre vznik takéhoto prístroja bolo odstránenie ručného písania textu. Človek jednoducho diktuje text a stroj ho za neho zapisuje. Nevýhodou tohto stroja je naviazanosť na konkrétnu osobu, s ktorou sa učil rozpoznávať jednotlivé písmena a slová.

Rozsiahly zoznam možných aplikácií Kohonenových máp je uvedený v ([22]). Spomeniem niekoľko z nich, ktoré má najviac zaujali. Jednou z oblastí je robotika, kde sa Kohonenové mapy využívajú pri navigácii robotov a na ich samotnú kontrolu ([17]). Jednou z hlavných úloh robotiky je umiestnenie ramena robota do požadovanej polohy ([27]). Aby sa roboti mohli pohybovať v priestore musia spracovávať množstvo informácií zo svojho okolia ako napríklad obraz. A práve na spracovanie obrazu sa opäť môžu využiť Kohonenové mapy. Medzi úlohy z tejto oblasti patrí napríklad digitálne zakódovanie obrazu ([1]), kompresia obrazu ([25]) a segmentácia obrazu ([21]).

V medicíne sa s Kohonenovými mapami môžeme stretnúť pri analýze signálov z elektrokardiogramu ([31]), pri analýze zvukov, ktoré vydávajú pľúca ([19]), pri analýze výsledkov hladiny glukózy v krvi získaných pomocou glukometra ([24]) a pri množstve ďalších iných aplikácií z rôznych oblastí.

5.6. Varianty Kohonenových máp s adaptatívnou topológiou

Klasické Kohonenové mapy reprezentujeme pomocou dvojvrstvovej neurónovej siete (vstupná a výstupná vrstva). Každý vstupný neurón je spojený s každým neurónom vo výstupnej vrstve. Neuróny vo výstupnej vrstve sú usporiadané do určitej topologickej štruktúry.

Tvar tejto topologickej štruktúry musí byť užívateľom zadaný dopredu ešte pred začiatkom algoritmu učenia Kohonenovej mapy (4.2). Tvar a veľkosť zvolenej topologickej štruktúry v ďalšom priebehu algoritmu učenia zostávajú nemenné. Práve toto je veľmi obmedzujúce pri použití Kohonenových máp. Kohonenové mapy sa využívajú na mapovanie mnohorozmerného vstupného priestoru do nízkodimenzionálneho priestoru a toto mapovanie je teda už dopredu obmedzované veľkosťou a tvarom zvolenej topologickej štruktúry výstupných neurónov. Z tohto dôvodu sa začali vyvíjať nové varianty Kohonenových máp, ktoré umožňujú počas učiaceho procesu meniť nielen pozíciu jednotlivých neurónov, ale aj topologickú štruktúru siete.

V ďalšom texte si predstavíme dve varianty tohto typu modelu, a to Kohonenové mapy s rastúcou mriežkou (5.6.1) a model rastúcich neurónových plynov (5.6.3).

5.6.1. Kohonenové mapy s rastúcou mriežkou

V tomto modeli Kohonenových máp ([10]) budeme pracovať s usporiadaním neurónov vo vstupnom priestore do mriežky so štvorcovým okolím o rozmeroch $k \times m$. Označme túto mriežku ako A, $A = (a_{ij}),\ 1 \leq i \leq k, 1 \leq j \leq m$.

Nech $X = \{\vec{x}_i,\ \vec{x}_i \in R^n, i = 1, \dots, P\}$ je množina P trénovacích vzorov. Počet neurónov označíme O. Ďalej $\vec{w}_j,\ 0 \leq j \leq O - 1$ je váhový vektor neurónu j určujúci jeho pozíciu vo vstupnom priestore. Premenná $\tau_j, 0 \leq j \leq O - 1$ udržiava informáciu o doposiaľ dosiahnutom počte víťazstiev v kompetícii pre každý neurón. Premenná λ_r (r-rast) ovplyvňuje rýchlosť zväčšovania počtu neurónov vo vstupnom priestore. Parameter učenia η_0 má konštantnú hodnotu počas celého algoritmu.

Na začiatku algoritmu učenia sa začína s mriežkou malých rozmerov (napr. 2×2). Hodnoty všetkých premenných τ_j nastavíme na nulu, $\tau_j = 0|\forall j$. Kým nie je splnená ukončovacia podmienka algoritmu, opakujeme nasledujúci postup.

Náhodne zvolíme trénovací vzor $\vec{x} \in X$ a ten predložíme na vstup sieti. V zvolenej metrike vzor $\vec{x}$ porovnáme s váhovými vektormi $\vec{w}_j$ všetkých neurónov a vyberieme taký neurón c, ktorého váhový vektor je najbližšie (je najpodobnejší) trénovaciemu vzoru. Hovoríme, že tento neurón vyhral kompetíciu a označíme ho ako víťazný neurón c. Aktualizujeme hodnoty všetkých váhových vektorov $\vec{w}_j$ podľa vzťahu

$$\vec{w}_j = \vec{w}_j + \eta_0 \exp\left(-\frac{d(j,c)^2}{2\sigma^2}\right)(\vec{x} - \vec{w}_j) \quad \forall j \in A \tag{5.9}$$

Za funkciu laterálnej interakcie je v tomto prípade zvolená už spomínaná Gaussová funkcia (5.4). Zvýšime počet víťazstiev τ_c o jedna víťaznému neurónu c. V tomto okamihu dochádza k rozhodnutiu, či sa pridajú neuróny do vstupného priestoru, alebo sa pokračuje predložením ďalšieho trénovacieho vzoru.

Ak počet doposiaľ vykonaných iterácií učiaceho algoritmu prevýšil počet $k \times m \times \lambda_r$, dôjde k pridaniu nových neurónov nasledujúcim spôsobom. Podľa najväčšej hodnoty $\tau_j,\ 0 \le j \le O-1$ nájdeme výstupný neurón q s najväčším počtom víťazstiev v kompetícii. Práve do okolia neurónu q budeme vkladať nové neuróny, čím zvýšime hustotu neurónov v danej oblasti a dosiahneme tým presnejšie a rovnomernejšie rozdelenie neurónov vo vstupnom priestore. Nájdeme k neurónu q susedný neurón f taký, že váhové vektory týchto neurónov sú si najviac vzdialené, a teda najmenej podobné. Hľadáme len medzi priamymi susedmi, tí môžu byť maximálne štyria.

Do topologickej mriežky vložíme riadok (prípadne stĺpec) nových neurónov, a to medzi riadky (prípadne stĺpce), ktoré obsahujú neuróny q a f. Bez ujmy na všeobecnosti môžeme predpokladať, že neuróny q a f sa nachádzajú na r-tom riadku mriežky A. Konkrétne nech $q = a_{rs}, f = a_{rs+1}$. Medzi stĺpce s a $s+1$ vložíme nový stĺpec s' s k novými neurónmi. Váhové vektory pre nové neuróny sa vytvoria podľa vzťahu:

$$\vec{w}_{rs'} = \frac{\vec{w}_{rs} + \vec{w}_{rs+1}}{2} \qquad 1 \le r \le k \tag{5.10}$$

Zvýšime počet stĺpcov mriežky m a všetky premenné τ_j opäť nastavíme na nulu, $\tau_j = 0|\forall j$. Ak nebola splnená ukončovacia podmienka algoritmu, pokračujeme s predložením ďalšieho trénovacieho vzoru $\vec{x} \in X$.

Po skončení algoritmu (rastovej fázy) sa môže previesť ešte dolaďovacia fáza, počas ktorej dôjde k poslednému jemnejšiemu posunu neurónov. V tejto fáze hodnota parametra učenia η_1 klesá s narastajúcim časom. Parameter λ_d (d-doladenie) vyjadruje, koľko sa priemerne vykoná aktualizačných krokov váhových vektorov každého neurónu v dolaďovacej fáze.

Adaptácia váhových vektorov prebieha podľa vzťahu (5.9), pričom priebeh parametra učenia je určený predpisom:

$$\eta(t) = \eta_0 \left(\frac{\eta_1}{\eta_0}\right)^{\frac{t}{t_{\max}}} \qquad t = 1, \dots, t_{\max} \tag{5.11}$$

$$t_{\max} = k \times m \times \lambda_d \tag{5.12}$$

Ukončovacia podmienka algoritmu je potrebná, aby sme zabránili nekontrolovanému zväčšovaniu počtu neurónov vo vstupnom priestore. Možnosti je viacero, medzi najjednoduchšie patrí napríklad určenie maximálneho počtu neurónov, ktoré sa môžu objaviť vo vstupnom priestore. Pri voľbe iných podmienok treba dávať pozor, aby došlo k ich naplneniu.

5.6.2. Algoritmus učenia Kohonenovej mapy s rastúcou mriežkou

1. Inicializácia:
   Začni s mriežkou malých rozmerov $k \times m$ (napr. 2×2). Inicializuj zložky váhových vektorov $\vec{w}_j$ malými náhodnými hodnotami, $0 \leq j \leq O - 1$, kde O je počet neurónov (v tomto prípade $O = 4$). Množinu neurónov označíme ako A. Nastav hodnoty parametrov $\tau_j = 0,\ 0 \leq j \leq O - 1$.

2. Predloženie trénovacieho vzoru:
   Náhodne vyber trénovací vzor $\vec{x} \in X$ a ten predlož na vstup sieti.

3. Výber víťazného neurónu:
   Nájdi neurón c, ktorého váhový vektor je najbližšie k predloženému trénovaciemu vzoru $\vec{x}$ a označ ho ako víťaza.

   $$c = \arg\min_j \left(\left\|\vec{x} - \vec{w}_j\right\|\right), \qquad \forall j \in A \tag{5.13}$$

   Zvýš hodnotu parametra τ_c označujúceho počet víťazstiev pre neurón c, $\tau_c = \tau_c + 1$.

4. Aktualizácia váh:
   V tomto prípade je za funkciu laterálnej interakcie zvolená Gaussová funkcia (5.4). Aktualizuj váhy podľa predpisu:

$$\vec{w}_j = \vec{w}_j + \eta_0 \exp\left(-\frac{d(j,c)^2}{2\sigma^2}\right)\left(\vec{x} - \vec{w}_j\right) \quad \forall j \in A \tag{5.14}$$

   Pre výpočet vzdialenosti $d(j,c)$ medzi neurónmi je zvolená Manhattanska metrika definovaná predpisom:

$$d_M\left(\vec{a},\vec{b}\right) = \sum_{i=1}^{n} |a_i - b_i| \tag{5.15}$$

   $\vec{a},\vec{b}$ sú n-prvkové vektory.

   Hodnoty parametra učenia η_0 a parametra šírky σ sú konštantné počas celého algoritmu.

5. Rozhodnutie o pridaní neurónov:
   Ak počet doposiaľ vykonaných iterácií algoritmu je rovný $k \times m \times \lambda_r$, pokračuj ďalším krokom algoritmu. V opačnom prípade pokračuj na krok algoritmu číslo 2.

6. Určenie najčastejšie vyhrávajúceho neurónu:
   Najčastejšie vyhrávajúci neurón q nájdi podľa najväčšej hodnoty τ.

$$q = \arg\max_j \left(\tau_j\right) \quad \forall j \in A \tag{5.16}$$

7. Určenie najvzdialenejšieho susedného neurónu:
   Definujme množinu priamych susedov neurónu q predpisom:

$$N_1(q) = \{j, d(q,j) = 1\} \tag{5.17}$$

   Z množiny $N_1(q)$ vyber susedný neurón f, ktorého váhový vektor $\vec{w}_f$ je najvzdialenejší od váhového vektora $\vec{w}_q$.

$$f = \arg\min_{j'} \left(\left\|\vec{x} - \vec{w}_{j'}\right\|\right), \ \forall j' \in N_1(q) \tag{5.18}$$

   Medzi riadky (prípadne stĺpce) obsahujúce neuróny q a f sa bude vkladať riadok (prípadne stĺpec) nových neurónov.

8. Pridanie nových neurónov:
   Bez ujmy na všeobecnosti predpokladajme, že neuróny q a f sa nachádzajú na r-tom riadku mriežky A, konkrétne nech $q = a_{rs}, f = a_{rs+1}$. Medzi stĺpce s a $s+1$ vlož nový stĺpec s' s k novými neurónmi. Váhové vektory pre nové neuróny vytvor podľa predpisu:

$$\vec{w}_{rs'} = \frac{\vec{w}_{rs} + \vec{w}_{rs+1}}{2} \qquad 1 \leq r \leq k \tag{5.19}$$

   Aktualizuj počet stĺpcov $m = m + 1$.
   V prípade vkladania riadku nových neurónov do topologickej mriežky postupujeme podobne.

9. Ukončenie algoritmu:
   Nastav hodnoty premenných τ_j na nulu, $\tau_j = 0 | \forall j$. Ak nie je splnená ukončovacia podmienka pokračuj na krok algoritmu číslo 2.

Nevýhodou tejto varianty Kohonenových máp je, že sa namiesto jedného neurónu vkladá do mriežky naraz väčší počet neurónov. Vždy sa vloží do mriežky celý riadok (prípadne stĺpec) nových neurónov, pričom sa môže stať, že nie všetky vložené neuróny budú plne využité. Tento spôsob vkladania neurónov je zvolený z dôvodu zachovania tvaru topológie neurónov vo vstupnom priestore.

5.6.3. Model rastúcich neurónových plynov

V tomto modeli ([9]) nie sú kladené žiadne požiadavky na tvar alebo veľkosť topologickej mriežky. Množinu neurónov opäť označme ako A. Samotná topológia sa vytvaruje v priebehu učenie danej siete. Počas učenia dochádza k postupnému pridávaniu jednotlivých neurónov do vstupného priestoru. Tento model je založený na princípe kompetitívného Hebbovského učenia.

Nech $X = \{\vec{x}_i,\ \vec{x}_i \in R^n, i = 1, \dots, P\}$ je množina P trénovacích vzorov. Počet neurónov označíme O. Niektoré neuróny sú navzájom spojené hranou, množinu týchto hrán označme ako $E, E \subset O \times O$. Každá hrana má priradený parameter $vek \in N \cup \{0\}$ vyjadrujúci vek tejto hrany. Ak vek nejakej hrany presiahne hodnotu parametra $vek_{\max}$ určujúceho maximálny povolený vek, tak sa táto hrana odstráni. Pre každý víťazný neurón c, si udržiavame hodnotu lokálnej chyby $Error_c$ vyjadrujúcu kvadrát

vzdialenosti váhového vektora $\vec{w}_c$ víťazného neurónu od predloženého trénovacieho vzoru $\vec{x}$. Parametre α a β ovplyvňujú zmenu lokálnych chýb jednotlivých neurónov, pričom platí $0 < \alpha, \beta < 1$. Parameter λ určuje počet iterácií algoritmu, ktoré sa musia vykonať, kým dôjde k pridaniu ďalšieho neurónu. Parametre ϵ_b a ϵ_n ovplyvňujú veľkosť priblíženia víťazného neurónu a jeho priamych susedov k predloženému trénovaciemu vzoru $\vec{x}$. Pre tieto parametre platí $1 > \epsilon_b \gg \epsilon_n$. Ďalej $\vec{w}_j$, $0 \leq j \leq O - 1$ je váhový vektor neurónu j určujúci jeho pozíciu vo vstupnom priestore.

Algoritmus začína len s dvoma neurónmi a, b, ktoré náhodne rozmiestnime vo vstupnom priestore. Zatiaľ nie sú žiadne neuróny v sieti spojené hranou, $E = \emptyset$. Kým nie je splnená ukončovacia podmienka algoritmu, opakujeme nasledujúci postup.

Náhodne zvolíme trénovací vzor $\vec{x} \in X$ a ten predložíme na vstup sieti. V zvolenej metrike vzor $\vec{x}$ porovnáme s váhovými vektormi $\vec{w}_j$ všetkých neurónov a vyberieme také neuróny c_1 a c_2, ktorých váhové vektory sú dva najbližšie (najpodobnejšie) trénovaciemu vzoru. Najbližšiemu neurónu c_1 nastavíme hodnotu lokálnej chyby $Error(c_1)$ podľa vzťahu:

$$Error(c_1) = Error(c_1) + \left\|\vec{w}_{c_1} - \vec{x}\right\|^2 \tag{5.20}$$

Všetkým hranám vychádzajúcim z c_1 zvýšime vek $vek = vek + 1$. Aktualizujeme hodnoty váhových vektorov víťazného neurónu c_1 a jeho priamych susedov podľa vzťahov:

$$\begin{aligned} \vec{w}_{c_1} &= \vec{w}_{c_1} + \epsilon_b\left(\vec{x} - \vec{w}_{c_1}\right) \\ \vec{w}_j &= \vec{w}_j + \epsilon_n\left(\vec{x} - \vec{w}_j\right) \qquad \forall j \in N_1(c_1) \end{aligned} \tag{5.21}$$

Ak medzi neurónmi c_1 a c_2 nevedie hrana, tak ju vytvoríme. V opačnom prípade nastavíme jej vek na nulu. V tomto okamihu dochádza k rozhodnutiu, či sa pridá nový neurón do vstupného priestoru, alebo sa pokračuje predložením ďalšieho trénovacieho vzoru $\vec{x} \in X$.

Ak počet doposiaľ vykonaných iterácií učiaceho algoritmu je násobkom parametra λ, dôjde k pridaniu nového neurónu nasledujúcim spôsobom. Nájdi neurón q s najväčšou lokálnou chybou a z pomedzi priamych susedov neurónu q nájdi neurón f opäť s najväčšou lokálnou chybou. Do stredu hrany spájajúcej neuróny q a f vlož nový neurón r. Pre tento neurón vytvor váhový vektor w_r podľa vzťahu:

$$\vec{w}_r = \frac{\vec{w}_q + \vec{w}_f}{2} \tag{5.22}$$

Odstráň hranu medzi neurónmi q, f a pridaj dve nové hrany medzi neuróny q, r a r, f. Neurónom q a f aktualizuj hodnotu lokálnej chyby vynásobením parametrom α. Novému neurónu r priraď hodnotu lokálnej chyby neurónu q. Následne aktualizuj hodnoty lokálnych chýb všetkých neurónov vynásobením parametrom β. Ak nebola splnená ukončovacia podmienka algoritmu, pokračujeme s ďalším trrénovacím vzorom $\vec{x} \in X$.

Ukončovacia podmienka algoritmu je potrebná, aby sme zabránili nekontrolovanému zväčšovaniu počtu neurónov vo vstupnom priestore. Opäť môžeme použiť napríklad podmienku určujúcu maximálny počet neurónov, ktoré sa môžu objaviť vo vstupnom priestore. Pri voľbe iných podmienok treba dávať pozor, aby došlo k ich naplneniu.

5.6.4. Algoritmus učenia rastúcich neurónových plynov

1. Inicializácia:
   Náhodne umiestni dva neuróny a, b vo vstupnom priestore. Množinu neurónov značíme ako A. Množina hrán je prázdna $E = \emptyset$.

2. Predloženie vzoru:
   Náhodne vyber trénovací vzor $\vec{x} \in X$ a ten predlož na vstupy neurónovej sieti.

3. Výber víťazných neurónov:
   Nájdi neurón c_1, ktorého váhový vektor je najbližší k predloženému trénovaciemu vzoru $\vec{x}$ a označ ho ako víťaza. Ďalej nájdi neurón c_2, ktorého váhový vektor je druhý najbližší k predloženému trénovaciemu vzoru $\vec{x}$.

$$c_1 = \arg\min_j \left\|\vec{w}_j - \vec{x}\right\| \quad \forall j \in A$$
$$c_2 = \arg\min_j \left\|\vec{w}_j - \vec{x}\right\| \quad \forall j \in A \setminus \{c_1\} \tag{5.23}$$

4. Nastavenie veku hrán:
   Definujme množinu priamych susedov neurónu c_1 predpisom:

$$N_1(c_1) = \{j, d(c_1, j) = 1\} \tag{5.24}$$

   Všetkým hranám vychádzajúcim z neurónu c_1 zvýš vek.

$$vek_{(c_1,j)} = vek_{(c_1,j)} + 1 \qquad \forall j \in N_1(c_1) \tag{5.25}$$

5. Priradenie lokálnej chyby $Error(c_1)$ víťaznému neurónu c_1:
   Víťaznému neurónu c_1 aktualizuj hodnotu lokálnej chyby podľa predpisu:

$$Error(c_1) = Error(c_1) + \left\|\vec{w}_{c_1} - \vec{x}\right\|^2 \tag{5.26}$$

6. Aktualizácia váh:
   Váhový vektor víťazného neurónu aktualizujeme podľa predpisu:

$$\vec{w}_{c_1} = \vec{w}_{c_1} + \epsilon_b(\vec{x} - \vec{w}_{c_1}) \tag{5.27}$$

   Váhové vektory priamych susedov víťazného neurónu aktualizujeme podľa predpisu:

$$\vec{w}_j = \vec{w}_j + \epsilon_n\left(\vec{x} - \vec{w}_j\right) \qquad \forall j \in N_1(c_1) \tag{5.28}$$

7. Aktualizácia hrán:
   Ak existuje hrana medzi neurónmi c_1 a c_2, tak nastav jej vek na nulu.

$$vek_{(C_1,C_2)} = 0 \tag{5.29}$$

   Ak takáto hrana neexistuje, tak ju vytvor. E= $E \cup \{(c_1, c_2)\}$. Ďalej odstráň všetky hrany, ktoré majú vek väčší ako $vek_{\max}$. Ak po odstránení týchto hrán vznikne neurón, z ktorého nevychádza žiadna hrana, tak tento neurón taktiež odstráň.

8. Rozhodnutie o pridaní neurónov:
   Ak počet doposiaľ vykonaných iterácií algoritmu je násobkom parametra λ, pokračuj ďalším krokom algoritmu. V opačnom prípade pokračuj na krok algoritmu číslo 2.

9. Určenie neurónov, medzi ktoré sa vloží nový neurón:
   Nájdi neurón q s najväčšou lokálnou chybou

$$q = \arg\max_j Error(j) \qquad \forall j \in A \tag{5.30}$$

   Z množiny najbližších susedov neurónu q nájdi neurón s najväčšou lokálnou chybou

$$f = \arg\max_{j'} Error(j') \qquad \forall j' \in N_1(q) \tag{5.31}$$

10. Vloženie nového neurónu:
    Do stredu hrany spájajúcej neuróny q a f vlož nový neurón r. $A = A \cup \{r\}$. Váhový vektor pre nový neurón vytvor podľa predpisu:

$$\vec{w}_r = \frac{\vec{w}_q + \vec{w}_f}{2} \tag{5.32}$$

    Zmaž hranu spájajúcu neuróny q, f a vytvor dve nové hrany spájajúce neuróny q, r a
    r, f.

$$E = E \cup \{(q,r),(r,f)\}$$
$$E = E \setminus \{(q,f)\} \tag{5.33}$$

    Neurónom q a f uprav hodnotu lokálnej chyby podľa predpisu:

$$Error(q) = \alpha Error(q)$$
$$Error(f) = \alpha Error(f) \tag{5.34}$$

    Novému neurónu r nastav rovnakú lokálnu chybu ako má neurón q po aktualizacií svojej lokálnej chyby.

$$Error(r) = Error(q) \tag{5.35}$$

11. Aktualizácia hodnôt lokálnych chýb:
    Všetkým neurónom uprav hodnoty lokálnych chýb podľa predpisu:

$$Error(j) = \beta Error(j) \quad 0 < \beta < 1, \ \forall j \in A \tag{5.36}$$

12. Ukončenie algoritmu:
    Ak nie je splnená ukončovacia podmienka pokračuj na krok algoritmu číslo 2.

Keďže hrany počas učiaceho procesu starnú a niektoré sa aj odstránia, dochádza vo vstupnom priestore k javu, že topologická štruktúra môže mať rôznu dimenzionalitu v rôznych oblastiach vstupného priestoru. Toto môže pomerne dosť zneprehľadniť prípadnú vizualizáciu výsledkov.

6. RBF neurónové siete

RBF (Radial Basis Function) neurónová sieť je viacvrstvová neurónová sieť s dopredným šírením signálu. Tento model neurónových sieti patrí k najmladším modelom neurónových sieti. V osemdesiatych rokoch sa v oblasti numerickej matematiky začali zaoberať interpoláciou a aproximáciou dát s využitím radiálnych bázických funkcií ([37]). Využiť tieto funkcie k vytvoreniu nového modelu neurónových sieti navrhol Broomhead a Lowe ([7]) v roku 1988. K ďalšiemu rozvoju RBF sieti prispeli Moody a Darken ([30]).

Radiálnu bázickú funkciu ([42]) si môžeme predstaviť ako funkciu určenú nejakým významným bodom (nazývaný ako stred funkcie), ktorá pre rovnako vzdialené argumenty od stredu dáva rovnaké funkčné hodnoty. V dvojrozmernom priestore, pokiaľ je vzdialenosť argumentov meraná v Euklidovej metrike (2.1), tvoria množiny rovnakých funkčných hodnôt kružnice. Z tohto dôvodu tieto funkcie označujeme ako radiálne.

RBF siete často dosahujú lepšie výsledky ako klasické neurónové siete, a to aj vďaka použitiu radiálnych bázických funkcií, ktoré viac odpovedajú recepčným poliam skutočných neurónov.

6.1. Architektúra RBF siete

RBF sieť je trojvrstvová neurónová sieť, pričom každá vrstva plní rôzny účel. Vstupná vrstva je tvorená I neurónmi a slúži len k prenosu vstupných dát do druhej skrytej vrstvy. Skrytá vrstva je nelineárna a je tvorená H RBF jednotkami (6.2), ktoré realizujú jednotlivé radiálne bázické funkcie. RBF jednotky sa nazývajú aj ako skryté jednotky a priestor, ktorý pokrývajú sa nazýva skrytý priestor. Výstupná vrstva je lineárna a je tvorená O výstupnými neurónmi.

Každý vstupný neurón je spojený s každou RBF jednotkou v skrytej vrstve, pričom spoj medzi i-tým vstupným neurónom a h-tou RBF jednotkou má priradenú váhu $c_{ih}, i = 1, \dots, I,\ h = 1, \dots, H$. Táto váha c_{ih} vyjadruje i-tú súradnicu stredu $\vec{c}_h$ h-tej RBF jednotky.

Každá RBF jednotka je spojená synaptickou váhou s každým neurónom vo výstupnej vrstve, presnejšie w_{hj} je hodnota váhy medzi h-tou RBF jednotkou a j-tým výstupným neurónom.

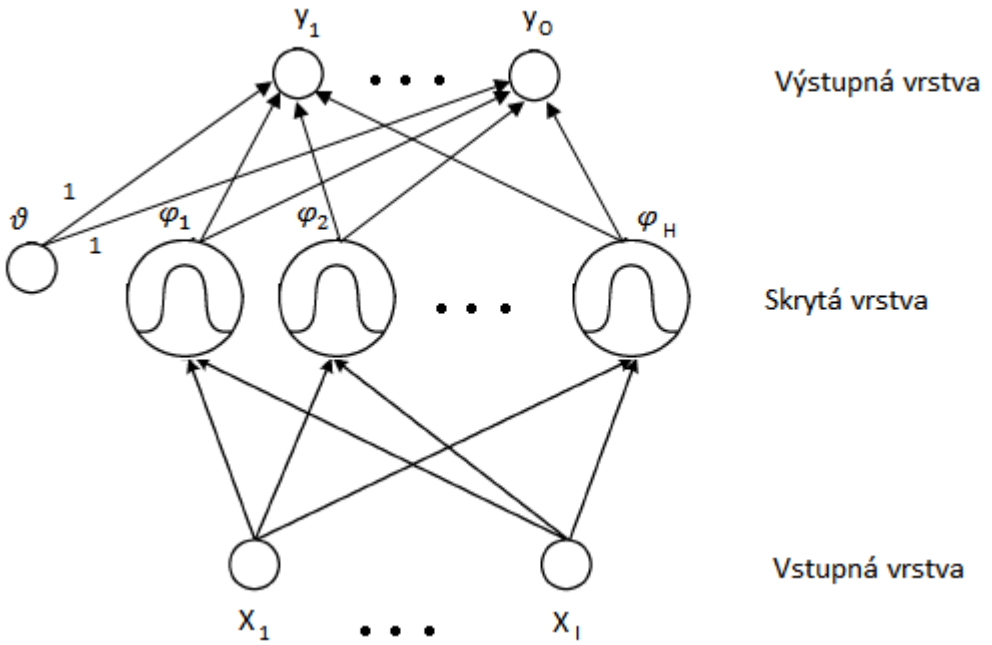
Do skrytej vrstvy ešte pridáme jeden špeciálny neurón, ktorý bude reprezentovať prah ϑ. Tento neurón má konštantnú výstupnú hodnotu, a to rovnú jednotke. Tento neurón podobne ako RBF jednotky je spojený synaptickou váhou s každým neurónom vo výstupnej vrstve.

RBF sieť prevádza dve rôzne transformácie. Prvá transformácia je nelineárna zo vstupného priestoru R^I do skrytého priestoru R^H a je prevádzaná RBF jednotkami. Druhá transformácia je už lineárna zo skrytého priestoru R^H do výstupného priestoru R^O a je realizovaná neurónmi vo výstupnej vrstve.

Celý proces návrhu architektúry RBF siete môžeme rozdeliť na dve základné fázy ([35]). V prvej fáze ide o vývoj RBF jednotiek a v druhej fáze o optimalizáciu hodnôt váh neurónov vo výstupnej vrstve. Prvá fáza má rozhodujúci vplyv na výslednú efektivitu fungovania RBF siete a je oveľa náročnejšia ako druhá optimalizačná fáza. Je to z dôvodu, že v prvej fáze sa musia riešiť nasledujúce problémy:

- určiť typ a počet RBF jednotiek je základným problémom pri návrhu architektúry RBF siete
- určiť rozmiestnenie RBF jednotiek vo vstupnom priestore má dôležitý vplyv na funkčnosť RBF siete
- optimalizácia parametrov jednotlivých RBF jednotiek je tiež podstatná pre správnu funkčnosť RBF siete

Cover vo svojej práci ([8]) dospel k výsledku, že je vhodné voliť väčší počet skrytých RBF jednotiek ako je dimenzia vstupu. Na základe svojich výsledkov sformuloval Coverov teorém (6.3).



Obrázok č. 14: Architektúra RBF siete tvorenej I vstupnými neurónmi, H RBF jednotkami a O výstupnými neurónmi. V skrytej vrstve je naviac neurón reprezentujúci prah ϑ.

6.2. RBF jednotka

RBF jednotka realizuje určenú radiálnu bázickú funkciu a je podobná perceptrónu ([42]). Má I reálnych vstupov, na ktoré sa predkladajú vstupné vektory $\vec{x} = (x_1, x_2, \dots, x_I)$. RBF jednotka si udržiava informáciu o pozícii stredu $\vec{c} = (c_1, c_2, \dots, c_I)$, pričom každý vstup jednotky x_i má priradenú váhu c_i. Táto váha vyjadruje i-tú súradnicu stredu $\vec{c}$ tejto RBF jednotky. Ďalej má RBF jednotka jeden reálny výstup y vyjadrujúci aktivitu jednotky. Navyše môže mať RBF jednotka ešte parameter b vyjadrujúci šírku. Obrázok č. 15 znázorňuje architektúru RBF jednotky.

Vnútorný potenciál ξ RBF jednotky vyjadruje vzdialenosť predloženého vzoru $\vec{x}$ od stredu $\vec{c}$ RBF jednotky. Ak je zadaný aj parameter šírky b, tak výslednú vzdialenosť predelíme týmto parametrom.

$$\xi = \frac{\|\vec{x} - \vec{c}\|}{b} \tag{6.1}$$

Vzdialenosť vektorov $\|\vec{x} - \vec{c}\|$ sa počíta vo zvolenej metrike. V teórii RBF sieti sa používa hlavne Euklidova metrika (2.1).

Parameter šírky b určuje veľkosť radiálnej oblasti okolo stredu $\vec{c}$ danej RBF jednotky, v ktorej má táto RBF jednotka významné výstupné hodnoty. Hodnota tohto parametra by nemala byť príliš veľká, aby nedošlo k veľkému prekrývaniu radiálnych oblastí RBF jednotiek. Chybovú funkciu závislú na parametroch šírky môžeme vyjadriť predpisom ([42]):

$$E(b_1, b_2, \dots, b_H) =$$

$$\frac{1}{2}\sum_{k=1}^{H}\left(\sum_{l=1}^{H}\exp\left(-\left(\frac{\|c_l - c_k\|}{b_k}\right)^2\right)\left(\frac{\|c_l - c_k\|}{b_k}\right)^2 - P\right)^2 \tag{6.2}$$

P určuje mieru prekrývania sa radiálnych oblastí jednotlivých RBF jednotiek. Chceme minimalizovať hodnotu tejto chybovej funkcie, v praxi sa však využívajú jednoduchšie heuristiky k určeniu hodnôt šírok. Napríklad sa šírka nastaví úmerne priemeru vzdialenosti niekoľko najbližších susedov danej RBF jednotky.

Výstupnú hodnotu y RBF jednotky spočítame ako hodnotu prenosovej funkcie φ v bode ξ.

$$y = \varphi(\xi) \tag{6.3}$$

Za prenosovú funkciu φ sa môže zvoliť napríklad niektorá z nižšie uvedených funkcií:

- Lineárna funkcia

$$\varphi(r) = r, \qquad r \in R \tag{6.4}$$

- Multi-kvadratická funkcia

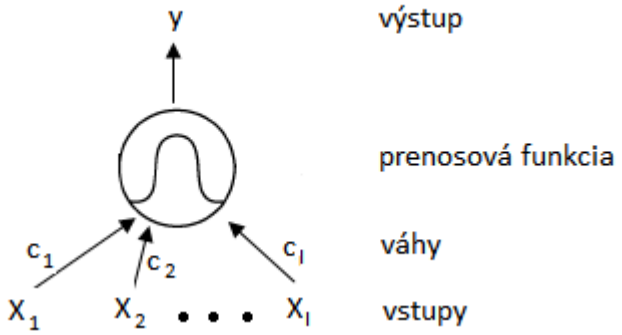
$$\varphi(r) = \sqrt{(r^2 + c^2)}, \qquad r \in R, c > 0 \tag{6.5}$$

- Inverzná multi-kvadratická funkcia

$$\varphi(r) = \frac{1}{\sqrt{(r^2 + c^2)}}, \qquad r \in R, c > 0 \tag{6.6}$$

- Gaussová funkcia

$$\varphi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right), \qquad r \in R, \sigma > 0 \tag{6.7}$$



Obrázok č. 15: Architektúra RBF jednotky

6.3. Coverov teorém

Problém klasifikácie vzorov, ktorý je nelineárne transformovaný do mnohorozmerného priestoru, bude v tomto priestore lineárne separovateľný pravdepodobnejšie ako v pôvodnom menej rozmernom priestore. (Cover, 1965)

Nech $X = \{\vec{x}_i,\ \vec{x}_i \in R^n, i = 1, \dots, P\}$ je množina P vstupných vzorov, pričom každý z týchto vzorov patrí do jednej z množín X_1, X_2. Pre každý vzor $\vec{x}, \vec{x} \in X$ definujeme vektorovú funkciu predpisom

$$\varphi(\vec{x}) = \left(\varphi_1(\vec{x}), \varphi_2(\vec{x}), \dots, \varphi_H(\vec{x})\right) \tag{6.8}$$

Táto funkcia $\varphi(\vec{x})$ namapuje vzor $\vec{x}$ s n-rozmerného vstupného priestoru do nového H-rozmerného priestoru. Funkcie $\varphi_h(\vec{x}), h = 1, \dots, H$ nazývame skryté funkcie, pretože ich účel je podobný skrytým jednotkám v dopredných RBF neurónových sieťach. Podobne priestor, ktorý pokrývajú tieto skryté funkcie nazývame skrytý priestor.

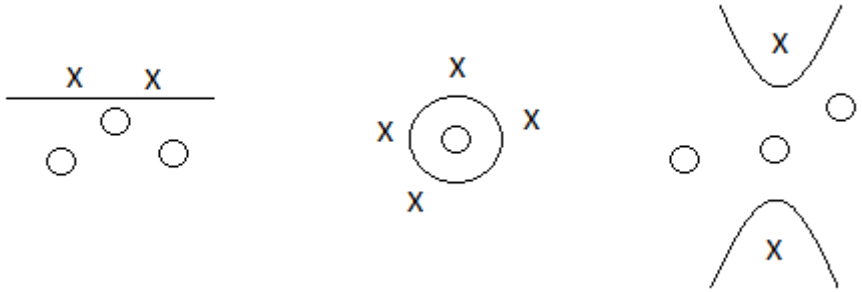
Dichotómiu $\{X_1, X_2\}$ množiny X nazveme φ-separovateľnú, ak existuje vektor $\vec{u}$,

$\vec{u} = (u_1, u_2, \dots, u_H)$ taký, že platí:

$$\begin{aligned} \vec{u}\varphi(\vec{x}) > 0, \quad & \vec{x} \in X_1 \\ \vec{u}\varphi(\vec{x}) < 0, \quad & \vec{x} \in X_2 \end{aligned} \tag{6.9}$$

Deliacu nadrovinu v skrytom priestore potom definujeme rovnosťou:

$$\vec{u}\varphi(\vec{x}) = 0 \tag{6.10}$$



Obrázok č. 16: Príklad rôznych φ-separovateľných dichotómií v 2 dvojrozmernom priestore. V poradí zľava: lineárne, sféricky a kvadraticky separovateľná dichotómia ([11]).

Predpokladajme, že všetky možné dichotómie množiny X sú rovnako pravdepodobné. Označme $Prop(P,H)$ pravdepodobnosť, že náhodne vybraná dichotomia je φ-separovateľná. Táto pravdepodobnosť je daná predpisom ([8]):

$$Prop(P,H) = \left(\frac{1}{2}\right)^{P-1} \sum_{m=0}^{H-1} \binom{P-1}{m} \tag{6.11}$$

Z predchádzajúceho vzťahu vyplýva, že čím väčšia dimenzionalita H skrytého priestoru sa zvolí, tým je hodnota pravdepodobnosti $Prop(P,H)$ bližšie k jednotke.

Niekedy však postačuje samotná nelinearita, aby sa pôvodný lineárne neseparovateľný problém zmenil bez zvyšovania dimenzionality priestoru na lineárne separovateľný problém.

Učiaci proces RBF sieti

Definujme trénovaciu množinu T, ktorej prvky sú usporiadané dvojice tvaru (vstupný vzor, požadovaný výstup). Vstupný vzor označuje vektor $\vec{x} = (x_1, x_2, \dots, x_I)$, ktorý predkladáme na vstupy neurónovej sieti. Požadovaný výstup je vektor $\vec{d} = (d_1, d_2, \dots, d_O)$ a predstavuje požadované výstupné hodnoty výstupných neurónov. Skutočný výstup pre daný vstupný vzor je vektor $\vec{y} = (y_1, y_2, \dots, y_O)$ a predstavuje skutočné hodnoty výstupných neurónov. Trénovaciu množinu obsahujúcu P trénovacích vzorov definujeme predpisom:

$$T = \{[\vec{x}_1, \vec{d}_1], \dots, [\vec{x}_P, \vec{d}_P]\}$$

Učiaci proces RBF sieti môžeme rozdeliť na dve rôzne úlohy ([11]). Prvá úloha je spojená s rozmiestnením a nastavením parametrov RBF jednotiek a v druhej úlohe ide o nastavenie hodnôt synaptických váh neurónov vo výstupnej vrstve. Proces vývoja RBF jednotiek je oveľa pomalší oproti úlohe nastavenia synaptických váh výstupných neurónov. Dôležitým faktom je, že rôzne vrstvy RBF siete vykonávajú rôzne úlohy. Z tohto dôvodu je rozumné rozdeliť optimalizáciu skrytej a výstupnej vrstvy do samostatných úloh, pričom na každú úlohu použijeme iný postup. RBF jednotky najčastejšie realizujú Gaussovú radiálnu bázickú funkciu so stredom v $\vec{c}$ definovanú predpisom:

$$\varphi(\vec{x}) = \exp\left(-\frac{\|\vec{x} - \vec{c}\|^2}{2\sigma^2}\right),$$

kde σ je šírka funkcie a určuje strmosť Gaussovej funkcie.

Celkový výstup RBF siete spočítame pomocou vzorca:

$$y_j(\vec{x}) = \sum_{h=1}^{H} \vec{w}_{hj}\, \varphi_h(\vec{x}), \qquad j = 1, \dots, O \tag{6.14}$$

$y_j(\vec{x})$ je výstupná hodnota j-tého neurónu vo výstupnej vrstve na predložený vstupný vzor $\vec{x}$.

Avšak v práve uvedenom vzorci nikde nefiguruje prah ϑ. Neurón reprezentujúci prah ϑ môžeme taktiež chápať ako extra RBF jednotku realizujúcu konštantnú funkciu $\varphi_0(\vec{x}) = 1$ pre každý predložený vzor $\vec{x}$. Čiže rozšírime množinu RBF jednotiek o novú RBF jednotku realizujúcu „bázickú funkciu“ $\varphi_0(\vec{x}) = 1$. Takže po zahrnutí tohto nového faktu celkový výstup RBF siete spočítame pomocou aktualizovaného vzorca:

$$y_j(\vec{x}) = \sum_{h=0}^{H} \vec{w}_{hj}\, \varphi_h(\vec{x}), \qquad j = 1, \dots, O \tag{6.15}$$

Pre RBF siete bolo vyvinutých niekoľko učiacich algoritmov. Niektoré z nich si predstavíme v nasledujúcom texte.

6.4. Náhodná voľba stredov RBF jednotiek

Jedná sa o pomerne jednoduchý algoritmus ([11]). V prvok kroku náhodne rozmiestnime pozície H stredov jednotlivých RBF jednotiek vo vstupnom priestore, a to tak, že náhodne vyberieme H rôznych vzorov $\vec{x}$ z trénovacej množiny T a tie prehlásime za stredy $\vec{c}$ jednotlivých RBF jednotiek.

Za radiálne bázické funkcie zvolíme Gaussovú funkciu, ktorej šírka σ je pevne určená vzhľadom k rozmiestneniu stredov RBF jednotiek. Radiálne bázické funkcie so stredom umiestnením v $\vec{c}_h, h = 1, \dots, H$ vyjadríme predpisom:

$$G(\|\vec{x} - \vec{c}_h\|^2) = \exp\left(-\frac{h}{d_{\max}^2}\|\vec{x} - \vec{c}_h\|^2\right), \qquad h = 1, \dots, H \tag{6.16}$$

$d_{\max}$ vyjadruje maximálnu vzdialenosť medzi zvolenými stredmi.

Šírka σ všetkých Gaussových radiálnych bázických funkcií je pevne určená, a to predpisom:

$$\sigma = \frac{d_{\max}}{\sqrt{2H}} \tag{6.17}$$

Prípadne sa ešte môže hodnota šírky σ jednotlivým funkciám pozmeniť. V oblastiach vstupného priestoru s nižšou hustotou výskytu vstupných dát môžeme hodnotu šírky zväčšiť.

6.4.1. Pseudo-inverzná metóda

V druhom kroku nám ešte zostáva nastaviť správne hodnoty synaptických váh medzi RBF jednotkami a neurónmi vo výstupnej vrstve. Toto môžeme vykonať priamo pomocou pseudo-inverznej metódy ([7]). Čiže chceme riešiť nasledujúci vzťah.

$$w = G^{+}d \tag{6.18}$$

G^{+} je označenie pseudo-inverznej matice k matici $G = \{g_{ih}\}$, ktorej prvky sú definované predpisom:

$$g_{ih} = \exp\left(-\frac{H}{d_{\max}^2}\|\vec{x}_i - \vec{c}_h\|^2\right), \qquad i = 1, \dots P, h = 1, \dots, H \tag{6.19}$$

$\vec{x}_i$ je i-tý vzor z trénovacej množiny.

Pseudo-inverznú maticu spočítame pomocou metódy SVD (Singular Value Decomposition) rozkladu.

Ak G je reálna matica rozmerov $N \times M$, potom existujú ortogonálne matice U, V

$U = \{\vec{u}_1, \vec{u}_2, \dots, \vec{u}_N\}$, V= $\{\vec{v}_1, \vec{v}_2, \dots, \vec{v}_M\}$ také, že

$$U^T G V = \mathrm{diag}(\sigma_1, \sigma_2, \dots, \sigma_K), \qquad K = \min(M, N) \tag{6.20}$$

pričom $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_K > 0$.

Čísla $\sigma_1, \sigma_2, \dots, \sigma_K$ nazývame singulárne hodnoty matice matice G. Pseudo-inverznú maticu G^{+} rozmerov $M \times N$ k matici G definujeme predpisom:

$$G^{+} = V \Sigma^{+} U^T, \tag{6.21}$$

kde Σ^{+} je matica rozmerov $N \times N$ definovaná predpisom:

$$\Sigma^{+} = \mathrm{diag}\left(\frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \dots, \frac{1}{\sigma_K}, 0, \dots, 0\right) \tag{6.22}$$

Tento postup návrhu a učenia RBF siete pomocou náhodnej voľby stredov RBF jednotiek je vhodný v prípade, že máme dostupnú rozsiahlu množinu trénovacích vzorov. V opačnom prípade nebude dosahovať RBF sieť požadovanú výkonnosť.

6.5. Hybridný učiaci proces

Nevýhodou predchádzajúcej metódy bola požiadavka na rozsiahlu množinu trénovacích vzorov. To však nemôžeme vždy splniť. Možnosťou ako sa tomuto problému vyhnúť je použiť hybridný učiaci ([11]) proces pozostávajúci s dvoch úplne odlišných štýlov učenia neurónových sieti. Jedná sa o:

1. Samo-organizačné učenie (self-organizing learning), ktorého účelom je odhadnúť vhodné rozmiestnenie stredov jednotlivých RBF jednotiek.
2. Učenie s učiteľom (supervised learning), ktorého účelom je nastaviť správne hodnoty synaptických váh medzi RBF jednotkami a neurónmi vo výstupnej vrstve.

Samo-organizačné učenie

Potrebujeme nejakú metódu pomocou, ktorej rozdelíme vstupné vzory do zhlukov. Zároveň po týchto zhlukoch požadujeme, aby boli čo možno najviac homogénne. Každý zhluk je reprezentovaný vektorom umiestneným do ťažiska zhluku, ktorý nazývame centroid. Následne do týchto centroidov príslušných zhlukov umiestnime stredy jednotlivých RBF jednotiek. Vhodným algoritmom, ktorý spĺňa tieto požiadavky napríklad K-means klastrovací algoritmus (3). Tento algoritmus rozdelí vstupný priestor do dopredu určeného počtu klastrov. Výsledné klastre K-means algoritmu sú kompaktné a navzájom disjunktné. Ďalším dôvodom pre voľbu tohto algoritmu je jeho rýchlosť.

K-means algoritmus popri identifikácií stredov jednotlivých RBF jednotiek zároveň nastaví šírky týchto RBF jednotiek**.** Ďalej nám zostáva nastaviť hodnoty synaptických váh. V tomto prípade budeme túto úlohu riešiť pomocou metódy najmenších štvorcov (least-mean-square algorithm).

Učenie s učiteľom

V tejto časti algoritmu chceme nastaviť príslušné hodnoty synaptických váh medzi RBF jednotkami a neurónmi vo výstupnej vrstve pomocou metódy najmenších štvorcov. Túto metódu budeme označovať ako LMS (least-mean-square algorithm)**.** LMS algoritmus patrí medzi gradientné algoritmy a vyznačuje sa predovšetkým svojou rýchlosťou.

Nech $T = \{[\vec{x}^1, \vec{d}^1], \dots, [\vec{x}^p, \vec{d}^p]\}$ je množina P trénovacích vzorov. Pre každý predložený vzor $\vec{x}^p$ na vstup RBF sieti dáva skrytá vrstva RBF jednotiek výstup $\varphi(\vec{x}^p) = (\varphi_1(\vec{x}^p), \varphi_2(\vec{x}^p), \dots, \varphi_H(\vec{x}^p))$, kde H je počet RBF jednotiek. Na vstup LMS

algoritmu budeme predkladať práve tieto vektory, ktoré vzniknú ako výstup skrytej vrstvy tvorenej RBF jednotkami. Vytvoríme novú trochu pozmenenú trénovaciu množinu

$T' = \{[\varphi(\vec{x}^1), \vec{d}_1], \dots, [\varphi(\vec{x}^p), \vec{d}_P]\}$, ktorej prvky budeme predkladať na vstup LMS algoritmu. $\eta(t)$ určuje hodnotu parametra učenia a platí $0 \leq \eta(t) \leq 1$.

6.5.1. Least-mean-square algoritmus

1. Inicializácia:
   Zvoľ malé náhodné hodnoty synaptických váh $w_{hj}, h = 1, \dots, H, j = 1, \dots, O$, pričom
   H určuje počet skrytých RBF jednotiek, O určuje počet neurónov vo výstupnej vrstve.

2. Predloženie trénovacieho vzoru:
   Predlož na vstup nový trénovací vzor tvaru $[\varphi(\vec{x}^p), \vec{d}^p]$ ([vstup, požadovaný výstup]).

3. Výpočet skutočného výstupu:
   Pre zadaný vstupný vzor $\varphi(\vec{x}^p)$ spočítame hodnotu reálneho výstupu RBF siete $\vec{y}^p$ podľa:

$$y_j^p = \sum_{h=0}^{H} w_{hj} \varphi_h(\vec{x}^p), \qquad j = 1, \dots, O \tag{6.23}$$

   y_j^p je skutočný výstup j-tého výstupného neurónu na predložený vstupný vzor $\varphi(\vec{x}^p)$.

4. Výpočet chybovej hodnoty:
   Spočítaj odchýlku siete $\vec{e}^p$ medzi požadovaným výstupom $\vec{d}^p$ a skutočným výstupom $\vec{y}^p$ siete na predložený vstupný vzor $\varphi(x_j^p)$ podľa predpisu:

$$\vec{e}^p = \vec{d}^p - \vec{y}^p \tag{6.24}$$

5. Aktualizácia váh:
   Aktualizuj synaptické váhy podľa predpisu:

$$w_{hj}(t+1) = w_{hj}(t) + \eta(t)\varphi(\vec{x}^p)\vec{e}^p, \quad h = 1, \dots, H, j = 1, \dots, O \tag{6.25}$$

6. Ukončenie algoritmu:
   Pokračuj na krok algoritmu číslo 2 a predlož sieti nový trénovací vzor.

K-means klastrovací algoritmus pre skryté RBF jednotky a algoritmus LMS pre výstupné neuróny môžu svoje výpočty prevádzať súčasne, a tým sa skráti čas potrebný na učiaci proces RBF siete.

6.6. Varianty RBF sieti s adaptatívnou topológiou

Pre RBF siete vzniklo viacero učiacich algoritmov, ktoré umožňujú počas samotného učenia siete meniť topológiu učenej RBF siete. Tieto algoritmy môžeme rozlišovať podľa dvoch faktorov:

- Aký druh aktualizačných operácií s topológiou RBF siete prevádzajú
- V ktorom okamihu učiaceho algoritmu sa tieto operácie vykonávajú

Metóda dopredného výberu (Forward selection) umožňuje len postupné pridávanie RBF jednotiek do skrytej vrstvy RBF siete. Na začiatku algoritmu sa začína s prázdnou množinou RBF jednotiek. Postupne sa do siete pridávajú také RBF jednotky, ktoré najviac prispievajú k znižovaniu výstupnej chyby RBF siete ([34],[33]).

Metóda spätnej eliminácie (Backward elimination) funguje obrátene oproti doprednému výberu. Na začiatku algoritmu je každému vstupnému vzoru priradená jedna RBF jednotka. Postupne sa odstraňujú také RBF jednotky, ktoré najmenej prispievajú k znižovaniu výstupnej chyby RBF siete ([12],[13]).

Metódy učenia RBF sieti, ktoré v sebe nemajú zahrnutú operáciu odstraňovania nevýhodných RBF jednotiek, často produkujú RBF siete zbytočne veľkých rozmerov. V prípade, že takáto operácia odstraňovania RBF jednotiek je implementovaná v učiacom algoritme, malo by sa kontrolovať splnenie kritéria pre odstraňovanie RBF jednotiek počas celého priebehu učiaceho algoritmu. Ak k odstraňovaniu nadbytočných RBF jednotiek zo siete dochádza až po predložení všetkých vstupných vzorov, tak RBF sieť počas učiaceho algoritmu nie je kompaktná ([40],[38]).

V mnohých reálnych aplikáciách týchto učiacich algoritmov totižto nie je možné počas ich priebehu určiť, či už boli predložené všetky vstupné vzory. Z tohto dôvodu býva

ťažké rozhodnúť, v ktorom okamihu učiaceho algoritmu sa má vykonať fáza odstraňovania RBF jednotiek.

V nasledujúcom texte si podrobnejšie predstavíme učiaci algoritmus GGAP-RBF (0). GGAP-RBF algoritmus umožňuje počas samotného učenia RBF siete nielen pridávanie, ale aj odoberanie jednotlivých RBF jednotiek.

6.7. GGAP-RBF učiaci algoritmus

GGAP-RBF (A Generalized Growing and Pruning) algoritmus ([14]) je sekvenčný učiaci algoritmus pre RBF neurónové siete. Tento algoritmus sa líši od ostatných učiacich algoritmov hlavne tým, že po každom predloženom vzore sa kontroluje, či nie je potrebné zvýšiť prípadne znížiť počet RBF jednotiek. Vďaka tomuto je RBF sieť kompaktná počas celého procesu učenia a výsledná sieť obsahuje zvyčajne oveľa menší počet skrytých RBF jednotiek. S tým súvisí aj rýchlosť algoritmu, menšia sieť sa naučí samozrejme rýchlejšie ako rozľahlé siete.

Pre každú RBF jednotku zavedieme nový pojem „významnosť“. Významnosť neurónu určuje mieru množstva informácií obsiahnutých v tomto neuróne o funkcii, ktorú sa sieť učí. Významnosť neurónu definujeme ako štatisticky priemerný príspevok tohto neurónu do celkového výstupu siete cez všetky doteraz predložené vzory.

Nová RBF jednotka môže byť pridaná do siete len vtedy, ak je to štatisticky významné pre všetky predložené vzory. A naopak, ak niektorá RBF jednotka má nízku hodnotu významnosti, tak takúto RBF jednotku odstránime z RBF siete.

Za radiálne bázické funkcie opäť zvolíme Gaussovú funkciu definovanú ako:

$$\varphi(\vec{x}) = \exp\left(-\frac{\|\vec{x} - \vec{c}\|^2}{2\sigma^2}\right) \tag{6.26}$$

Výstup RBF siete spočítame podľa predpisu:

$$f_1(\vec{x}) = \sum_{h=1}^{H} \vec{w}_h \, \varphi_h(\vec{x}) \tag{6.27}$$

Ak z RBF siete odstránime h-tú RBF jednotku, výstup RBF siete bude:

$$f_2(\vec{x}) = \sum_{k=1}^{h-1} \vec{w}_k \, \varphi_k(\vec{x}) + \sum_{k=h+1}^{H} \vec{w}_k \, \varphi_k(\vec{x}) \tag{6.28}$$

Definujme pre vstupný vzor $\vec{x}_p$ chybu $E(h,p)$ spôsobenú odstránením h-tej RBF jednotky zo siete predpisom:

$$E(h,p) = \left\| f_1(\vec{x}_p) - f_2(\vec{x}_p) \right\|_q = \|\vec{w}_h\|_q \varphi_h(\vec{x}_p), \qquad p = 1, \dots, P \tag{6.29}$$

$\|.\|_q$ znamená q-normu a vyjadruje L_q-vzdialenosť medzi dvoma bodmi v Euklidovom priestore.

Chybu $E_q(h)$ pre všetkých P vstupných vzorov spôsobenú odstránením h-tej RBF jednotky zo siete vyjadríme:

$$E_q(h) = \|(E(h,1), E(h,2), \dots, E(h,P))^T\|_q \tag{6.30}$$

Uvedený vzťah ďalej môžeme prepísať ako:

$$E_q(h) = \left(\frac{\sum_{p=1}^{P} E^q(h,p)}{P} \right)^{\frac{1}{q}} = \|\vec{w}_h\|_q \left(\frac{\sum_{p=1}^{P} \varphi_h^q(\vec{x}_p)}{P} \right)^{\frac{1}{q}} \tag{6.31}$$

Významnosť $E_{\text{vyz}}(h)$ neurónu h môžeme vyjadriť predpisom:

$$\begin{aligned} E_{\text{vyz}}(h) &= \lim_{P \to +\infty} E_q(h) \\ &= \|\vec{w}_h\|_q \left(\int_X \exp\left(-\frac{q\|\vec{x} - \vec{c}_h\|^2}{\sigma_h^2}\right) p(\vec{x}) dx \right)^{\frac{1}{q}} \end{aligned} \tag{6.32}$$

$p(\vec{x})$ je funkcia určujúca hustotu rozdelenia vstupných vzorov z rozsahu množiny X, kde $X \subset R^I$.

Aby sme sa vyhli integrovaniu funkcie $p(\vec{x})$, môžeme ju analyticky vyjadriť pre niektoré často používane funkcie $p(\vec{x})$ ako je napríklad uniformná, normálna, exponenciálna funkcia ([14]).

Uniformné rozdelenie

Vstupné vzory sú vyberané uniformne z rozsahu X, potom funkcia $p(\vec{x})$ je definovaná ako

$$p(\vec{x}) = \left(\frac{1}{S(X)}\right) \tag{6.33}$$

$S(X)$ je veľkosť rozsahu X.

Po dosadení $p(\vec{x})$ do vzťahu (6.32) môžeme tento vzťah aproximovať:

$$E_{\text{vyz}}(h) \approx \|\vec{w}_h\|_q \left(\frac{\pi}{q}\right)^{\frac{I}{2q}} \left(\frac{\sigma_h^I}{S(X)}\right)^{\frac{1}{q}} \tag{6.34}$$

V tomto prípade potrebujeme poznať hodnotu $S(X)$ množiny X. Často býva tato hodnota známa, prípadne ľahko odhadnuteľná. Avšak môžeme jednoducho vstupné vzory jednoducho normalizovať do $[0,1]^I$ a nastaviť $S(X) = 1$.

Normálne rozdelenie

Normálne rozdelenie je známe aj pod názvom Gaussovo rozdelenie. V tomto prípade je funkcia $p(\vec{x})$ definovaná:

$$p(\vec{x}) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(\vec{x} - \vec{c})^2}{2\sigma^2}\right) \tag{6.35}$$

Po dosadení $p(\vec{x})$ do vzťahu (6.32) môžeme tento vzťah odhadnúť:

$$E_{\text{vyz}}(h) \approx \|\vec{w}_h\|_q \left(\frac{\sigma_h}{\sqrt{2q\sigma}}\right)^{\frac{1}{q}} \exp\left(-\frac{(\vec{c} - \vec{c}_h)^2}{2q\sigma^2}\right) \tag{6.36}$$

Exponenciálne rozdelenie

Funkciu $p(\vec{x})$ definujeme nasledovne:

$$p(\vec{x}) = \frac{1}{\vec{c}} \exp\left(\frac{\vec{x}}{\vec{c}}\right) \tag{6.37}$$

Vzťah (6.32) môžeme odhadnúť:

$$E_{\text{vyz}}(h) \approx \|\vec{w}_h\|_q \left(\sqrt{\frac{\pi}{q}}\frac{\sigma_h}{\vec{c}}\right)^{\frac{1}{q}} \exp\left(-\frac{\vec{c}_h}{\vec{c}} - \frac{\sigma_h^2}{4q\vec{c}^2}\right) \tag{6.38}$$

V mnohým aplikáciách sa za funkciu $p(x)$ volí niektorá z práve uvedených funkcií, prípadne ich kombinácia.

Nech $T = \{[\vec{x}_1, \vec{d}_1], \ldots, [\vec{x}_P, \vec{d}_P]\}$ je množina P trénovacích vzorov. Parameter $e_{\min}$ udáva hodnotu očakávanej aproximačnej presnosti siete. Parameter ϵ_p vyjadruje minimálnu vzdialenosť predloženého vzoru $\vec{x}$ od stredu $\vec{c}_{nr}$ najbližšej RBF jednotky, aby sa mohla vytvoriť nová RBF jednotka, ktorej stred bude v predloženom vzore $\vec{x}$. $\vec{e}_p$ vyjadruje odchýlku siete medzi požadovaným výstupom $\vec{d}_p$ a skutočným výstupom $\vec{y}_p$ siete na predložený vstupný vzor $\vec{x}_p$. Parameter κ určuje mieru prekrytia odpovedi RBF jednotiek vo vstupnom priestore.

Na začiatku učiaceho procesu RBF sieť neobsahuje žiadnu skrytú RBF jednotku. Postupne budeme predkladať na vstup sieti trénovacie vzory $\vec{x}$. Posledný trénovací vzor, ktorý bol predložený na vstup RBF sieti označme $\vec{x}_p$.

Pre každý predložený vzor spočítame celkový výstup RBF siete podľa vzťahu:

$$y_j = \sum_{h=1}^{H} w_{hj} \exp\left(-\frac{\|\vec{x}_p - \vec{c}_h\|^2}{2\sigma_h^2}\right), \qquad j = 1, \ldots, O \tag{6.39}$$

Vypočítame hodnoty parametrov ϵ_p a $\vec{e}_p$, ktoré sú potrebné pri rozhodovaní o pridaní novej RBF jednotky podľa vzťahu:

$$\begin{gathered} \epsilon_p = \max\{\epsilon_{\max}\gamma^p, \epsilon_{\min}\},\ 0<\gamma<1 \\ \vec{e}_p = \vec{d}_p - \vec{y}_p \end{gathered} \tag{6.40}$$

Ak je posledný predložený vzor $\vec{x}_p$ dostatočne vzdialený od stredu $\vec{c}_{nr}$ najbližšej RBF jednotky a zároveň významnosť eventuálne pridanej RBF jednotky bude väčšia ako požadovaná minimálna aproximačná presnosť siete $e_{\min}$, tak sa vytvorí nová RBF jednotka. Parametre novo vzniknutej RBF jednotky nastav podľa vzťahov:

$$\vec{w}_{H+1} = \vec{e}_p$$
$$\vec{c}_{H+1} = \vec{x}_p \tag{6.41}$$
$$\sigma_{H+1} = \kappa \left\| \vec{x}_p - \vec{c}_{nr} \right\|$$

Ak neboli splnené podmienky pre pridanie novej RBF jednotky, tak uprav parametre $\vec{w}_{nr}, \vec{c}_{nr}, \sigma_{nr}$ najbližšej RBF jednotky pomocou EKF algoritmu. Následne skontroluj túto upravenú RBF jednotku, či jej významnosť $E_{\text{vyz}}(nr)$ neklesla pod požadovanú hranicu $e_{\min}$. Ak hodnota významnosti $E_{\text{vyz}}(nr)$ sa stala menšou ako $e_{\min}$. Odstráň túto RBF jednotku zo siete.

6.7.1. GGAP-RBF algoritmus

1. Výpočet celkového výstupu RBF siete:
   Vypočítaj celkový výstup RBF siete podľa predpisu:

$$y_j = \sum_{h=1}^{H} w_{hj} \exp\left(- \frac{\left\| \vec{x}_p - \vec{c}_h \right\|^2}{2\sigma_h^2} \right), \qquad j = 1, \ldots, O \tag{6.42}$$

   kde O určuje počet výstupných neurónov a H určuje počet skrytých RBF jednotiek.

2. Výpočet hodnôt rastových kritérií:
   Vypočítaj hodnoty parametrov určujúcich hodnoty kritérií, ktoré sú potrebné pri rozhodovaní o zvýšení počtu RBF jednotiek podľa:

$$\epsilon_p = \max\{\epsilon_{\max}\gamma^p, \epsilon_{\min}\}, \; 0<\gamma<1$$
$$\vec{e}_p = \vec{d}_p - \vec{y}_p \tag{6.43}$$

3. Rozhodnutie o pridaní novej RBF jednotky:
   Vyhodnoť nasledujúce kritéria:

$$\left\|\vec{x}_p - \vec{c}_{nr}\right\| > \epsilon_p \tag{6.44}$$

$$\left\|\vec{e}_p\right\|_q \left(\int_X \exp\left(- \frac{q\left\|\vec{x} - \vec{x}_p\right\|^2}{\kappa^2 \left\|\vec{x}_p - \vec{c}_{nr}\right\|^2} \right) p(x)dx \right)^{\frac{1}{q}} > e_{\min} \tag{6.45}$$

   kde $\vec{x}_p$ je posledný predložený vzor na vstup RBF sieti.

   - Ak boli splnené obe kritéria pre zvýšenie počtu RBF jednotiek, tak vytvor novú RBF jednotku a nastav je parametre podľa:

$$\begin{aligned} \vec{w}_{H+1} &= \vec{e}_p \\ \vec{c}_{H+1} &= \vec{x}_p \\ \sigma_{H+1} &= \kappa\left\|\vec{x}_p - \vec{c}_{nr}\right\| \end{aligned} \tag{6.46}$$

     $\vec{w}_{h+1}$ je vektor vyjadrujúci hodnoty synaptických váh vychádzajúcich z $h+1$ RBF jednotky.

   - Ak nie sú splnené obe kritéria pre zvýšenie počtu RBF jednotiek, tak uprav hodnoty parametrov $\vec{w}_{nr}, \vec{c}_{nr}, \sigma_{nr}$ najbližšej RBF jednotky pomocou EKF algoritmu. Ak významnosť tejto RBF jednotky $E_{\text{vyz}}(nr)$ klesla pod požadovanú hranicu $e_{\min}$, odstráň RBF jednotku s indexom nr. Zníž dimenzionalitu v EKF.

$$E_{\text{vyz}}(nr) < e_{\min}$$

$$E_{\text{vyz}}(nr) = \left\|\vec{w}_{nr}\right\|_q \left(\int_X \exp\left(- \frac{q\left\|\vec{x}_p - \vec{c}_{nr}\right\|^2}{\sigma_{nr}^2} \right) p(x)dx \right)^{\frac{1}{q}} \tag{6.47}$$

4. Ukončenie algoritmu:
   Pokiaľ sú na vstupe ďalšie trénovacie vzory, pokračuj na krok algoritmu číslo 1.

V priebehu GGAP algoritmu, ak nedôjde k pridávaniu RBF jednotky do siete, je potrebné nastaviť parametre najbližšej (v Euklidovej vzdialenosti) RBF jednotky k poslednému predloženému vstupnému vzoru. Keďže sa upravujú hodnoty len jednej RBF jednotky, tak jedine tejto upravenej RBF jednotke sa mohla zmeniť hodnota významnosti. Takže následne stačí skontrolovať hodnotu významnosti práve tejto jednej RBF jednotky a jedine táto RBF jednotka môže byť z dôvodu nízkej významnosti v danom okamihu odstránená. Práve tento fakt, že v každom kroku upravujeme len jednu RBF jednotku, výrazne zrýchľuje tento učiaci algoritmus.

Hodnoty parametrov nastavujeme pomocou EKF (Extended Kalman Filter) algoritmu. EKF algoritmus nastavuje parametre všetkých RBF jednotiek. Keďže v GGAP algoritme potrebujeme nastaviť parametre len jednej RBF jednotky, dochádza k výraznému zjednodušeniu a zrýchleniu výpočtov v EKF algoritme.

6.7.2. Extended Kalman Filter algoritmus

Tento algoritmus aktualizuje vektor $\vec{m}$ obsahujúci hodnoty parametrov $\vec{m} = [\vec{w}_0^T, \vec{w}_1^T, \vec{c}_1^T, \sigma_1, \dots, \vec{w}_H^T, \vec{c}_H^T, \sigma_H]$ jednotlivých RBF jednotiek podľa vzťahu:

$$\vec{m}_i = \vec{m}_{i-1} + K_i e_i \tag{6.48}$$

K_i je Kalmanov úžitkový vektor definovaný predpisom:

$$K_i = P_{i-1} B_i [R_i + B_i^T P_{i-1} B_i]^{-1} \tag{6.49}$$

$B_i = \nabla_{\vec{m}} f(\vec{x}_i)$ je gradientná matica funkcie $f(\vec{x}_i)$ s ohľadom na vektor parametrov $\vec{m}$ definovaná predpisom:

$$B_i = [I, \varphi_1(\vec{x}_i) I, \varphi_1(\vec{x}_i) \frac{2\vec{w}_1}{\sigma_1^2} (\vec{x}_i - \vec{c}_1)^T, \varphi_1(\vec{x}_i) \frac{2\vec{w}_1}{\sigma_1^3} \|\vec{x}_i - \vec{c}_1\|^T, \dots,$$
$$\varphi_H(\vec{x}_i) I, \varphi_H(\vec{x}_i) \frac{2\vec{w}_H}{\sigma_H^2} (\vec{x}_i - \vec{c}_H)^T, \varphi_H(\vec{x}_i) \frac{2\vec{w}_H}{\sigma_H^3} \|\vec{x}_i - \vec{c}_H\|^T]^T \tag{6.50}$$

R_i je rozdiel v miere šumu, P_i je kovariančná matica, ktorá sa upravuje podľa:

$$P_i = [I_{z \times z} - K_i B_i^T] P_{i-1} + q I_{z \times z} \tag{6.51}$$

$I_{z \times z}$ je jednotková matica. z je počet parametrov, ktoré sa aktuálne nastavujú. q je skalár, ktorý určuje prípustné náhodné kroky v smere gradientu vektora. P_i je $z \times z$ pozitívne definitná symetrická matica.

Keď sa zväčší počet RBF jednotiek, zväčší sa dimenzia P_i nasledovne:

$$P_i = \begin{pmatrix} P_{i-1} & 0 \\ 0 & p_0 I_{z_1 \times z_1} \end{pmatrix} \tag{6.52}$$

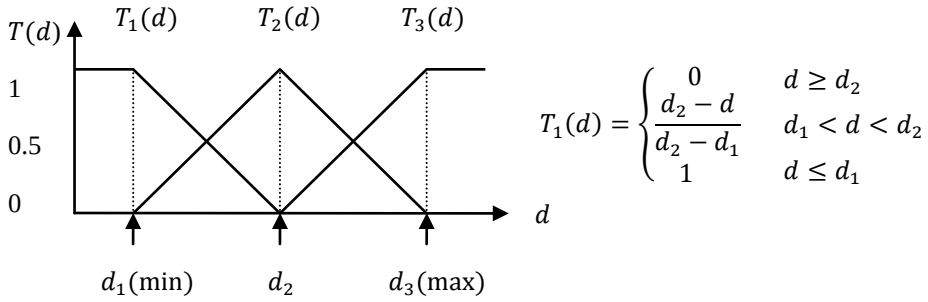
riadky a stĺpce sa inicializujú na hodnotu p_0, ktorá predstavuje odchýlku medzi vstupným vzorom $\vec{x}_i$ a požadovaným výstupom $\vec{y}_i$. z_1 je rozmer jednotkovej matice a vyjadruje počet nových parametrov, ktoré so sebou priniesla nová RBF jednotka.

6.8. Kontextové klastrovanie

Kontextové klastrovanie je modifikáciou FCM klastrovacej metódy (4). Kontextové klastrovanie ([35]) použijeme na rozdelenie vstupných vzorov do klastrov, pričom pri tvorbe týchto klastrov sa berie v úvahu množina preddefinovaných fuzzy množín (nazývané ako kontexty), ktoré vznikli vo výstupnom priestore.

Nech $T = \{[\vec{x}_1, d_1], \dots, [\vec{x}_P, d_P] | \vec{x}_p \in R^n, d_p \in R, p = 1, \dots, P\}$ je množina P trénovacích vzorov.

Najskôr vo výstupnom priestore rozdelíme výstupné hodnoty do klastrov pomocou algoritmu K-means clustering (3) nasledujúcim spôsobom. K-means algoritmom rozdelíme výstupné dáta do $K - 2$ klastrov. Najmenšiu a najväčšiu hodnotu výstupných dát prehlásime za dva centroidy, čiže celkovo takto dostaneme K klastrov reprezentovaných centroidami $\vec{c}_k, k = 1, \dots, K$. Nad týmito centroidami preložíme trojuholníkovú členskú funkciu, tak ako vidieť na obrázku č. 17. Modusy tejto funkcie sú umiestnené v centroidoch $\vec{c}_k$. Členské funkcie označíme $T_1, \dots, T_K$. Hodnotu členstva výstupu d_p v j-tom kontexte fuzzy množiny označíme t_{jp}.



Obrázok č. 17: Trojuholníková členská funkcia s polovičným prekrývaním sa po sebe idúcich fuzzy množín. V tomto prípade je počet klastrov $K = 3$ ([35]).

V ďalšom kroku prevedieme klastrovanie vstupných vzorov pomocou kontextového klastrovania, pričom berieme v úvahu fuzzy množinu kontextov získanú vo výstupnom priestore.

Členskú maticu j-tého kontextu označíme $U(T_j)$ a definujeme predpisom:

$$U(T_j) = \left\{ u_{kp} = [0,1], \sum_{k=1}^{K} u_{kp} = t_{jp}\ \forall p, 0 < \sum_{p=1}^{P} u_{kp} < P\ \forall k \right\} \tag{6.53}$$

K určuje počet klastrov a t_{jp} určuje hodnotu členskej funkcie p-tého vzoru v j-tom kontexte. Chybová funkcia v kontextovom klastrovaní je definovaná predpisom:

$$E_{CFCM} = \sum_{p=1}^{P} \sum_{k=1}^{K} u_{kp}^{m} \left\| \vec{x}_p - \vec{c}_k \right\|^2, \qquad 1 \leq m < \infty, \tag{6.54}$$

m je fuzzyfikačný parameter.

Minimalizáciu chybovej funkcie E_{CFCM} podobne ako v FCM algoritme dosiahneme iteratívnym prepočítavaním hodnôt prvkov členskej matice u_{kp} a pozícií centroidov $\vec{c}_k$. Aktualizáciu hodnôt prvkov členskej matice preveď podľa vzťahu:

$$u_{kp} = \frac{t_{jp}}{\sum_{l=1}^{K} \left(\frac{\left\| \vec{x}_p - \vec{c}_k \right\|^2}{\left\| \vec{x}_p - \vec{c}_l \right\|^2} \right)^{\frac{2}{m-1}}}, \qquad k = 1, \ldots, K, p = 1, \ldots P \tag{6.55}$$

Aktualizáciu pozícií jednotlivých centroidov preveď podľa vzťahu:

$$\vec{c}_k = \frac{\sum_{p=1}^{P} u_{kp}^{m} \vec{x}_p}{\sum_{p=1}^{P} u_{kp}^{m}}, \qquad k = 1, \ldots, K \tag{6.56}$$

6.8.1. Redukcia dimenzionality vstupného priestoru

Cieľom redukcie dimenzionality vstupného priestoru je eliminovať čo možno najväčšie množstvo vstupných premenných, ktoré sa predkladajú na vstupy RBF sieti. Zníženie počtu vstupných premenných vedie k jednoduchším architektúram RBF sieti. Zároveň sa z tohto dôvodu skracuje čas potrebný na návrh architektúry a na proces učenia RBF siete.

Eliminácia vstupných premenných sa však tiež musí riadiť istými pravidlami. Po eliminácií vstupných premenných musí platiť, že pomocou premenných, ktoré zostali,

bude stále možné riešiť pôvodný problém s požadovanou presnosťou. Eliminovať sa budú hlavne tie premenné, ktoré sú bezvýznamné v zmysle, že nemajú zásadný vplyv v procese návrhu architektúry RBF siete ani na konečné výsledky tejto RBF siete.

Uvažujme dva vstupné vzory $\vec{x}_p, \vec{x}_q$ s indexmi p, q. Označme stĺpce členskej matice, ktoré odpovedajú týmto vzorom ako U_p a U_q. Vzájomnú blízkosť (proximity) týchto vstupných vzorov označíme ako $Prox(U_p, U_q)$, ktorú môžeme vyjadriť vzťahom:

$$Prox(U_p, U_q) = B_{p,q}$$
$$B_{p,q} = \sum_{k=1}^{K} \min(u_{kp},\ u_{kq}),\ \ p, q = 1, \dots, P \tag{6.57}$$

Funkcia vyjadrujúca vzájomnú blízkosť dvoch vzorov je symetrická a vracia jednotku pre rovnaké vstupné vzory a taktiež v prípade, že hodnoty stupňov členstva týchto vzorov do jednotlivých klastrov sú zhodné.

Označme maticu blízkosti v kontexte j ako $U_{(j)}$ a platí $Prox(U_{(j)}) = [B_{p,q}]$.

Naším cieľom je znížiť dimenziu vstupného priestoru, uvažujme preto nejakú podmnožinu vstupných parametrov. Pre vstupné vzory s parametrami z tejto podmnožiny spočítame opäť členskú maticu $U'_{(j)}$ predpisom:

$$u'_{kp} = \frac{t_{jp}}{\sum_{l=1}^{K} \left(\frac{\left\| \vec{x'}_p - \vec{c'}_k \right\|^2}{\left\| \vec{x'}_p - \vec{c'}_l \right\|^2} \right)^{\frac{2}{m-1}}},$$
$$k = 1, \dots, K, p = 1, \dots P \tag{6.58}$$

$\vec{x'}_p$ sú vstupné vzory s vybranými vstupnými parametrami. $\vec{c'}_k$ sú centroidy klastrov pre množinu vstupných vzorov s vybranými vstupnými parametrami, ktoré spočítame podľa vzťahu (6.55).

Na určenie podmnožiny vstupných parametrov použijeme genetický algoritmus. Informácie o genetických algoritmoch možno nájsť v literatúre ([32],[18],[20],[44]).

Pomocou matice $U'_{(j)}$ spočítame maticu blízkosti pre zvolené vstupné parametre podľa vzťahu:

$$Prox\left(U'_{(j)}\right) = \left[B'_{p,q}\right]$$

$$B'_{p,q} = \sum_{k=1}^{K} \min\left(u'_{kp}, u'_{kq}\right), \qquad p, q = 1, \dots, P \tag{6.59}$$

Následne určíme vzdialenosť medzi maticami blízkosti podľa vzťahu:

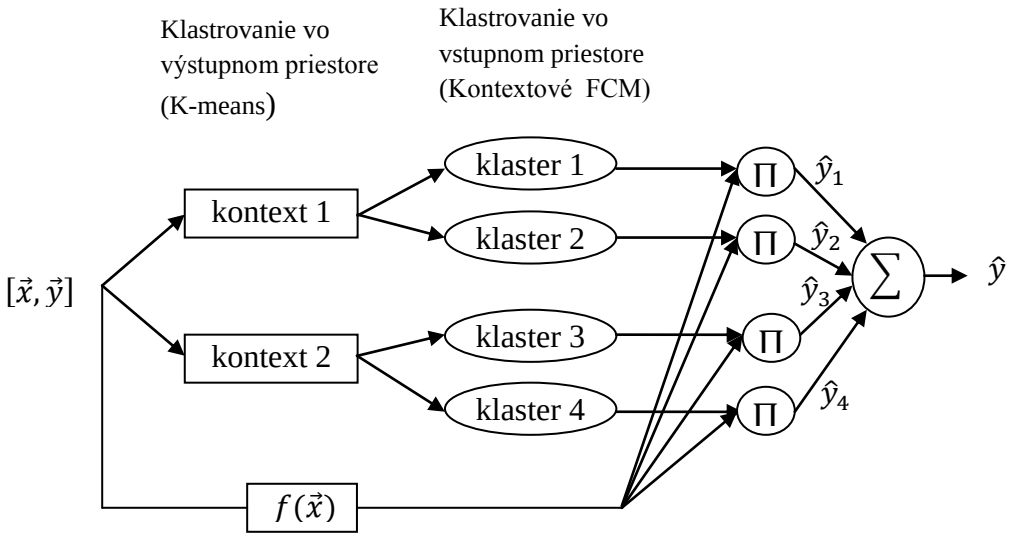
$$D_j = \left\| Prox\left(U_{(j)}\right) - Prox\left(U'_{(j)}\right) \right\| = \left\| B_{p,q} - B'_{p,q} \right\| = \\ = \sum_{p=1}^{P} \sum_{q>p}^{P} \left| B_{p,q} - B'_{p,q} \right| \tag{6.60}$$

Táto vzdialenosť D_j vyjadruje ako veľmi sa zmenilo rozmiestnenie vstupných vzorov s vybranými parametrami do klastrov oproti pôvodným vstupným vzorom. Naším cieľom je, aby rozdelenie vstupných vzorov s vybranými parametrami do klastrov bolo čo najviac podobné rozdeleniu pôvodných vstupných vzorov, čiže chceme, aby táto hodnota D_j bola čo najmenšia.

Vstupné vzory majú dimenziu rovnú n. Chceme vybrať vhodnú podmnožinu I parametrov vstupných vzorov, aby vybrané vzory boli menšej dimenzie $s < n$ ako pôvodné vzory. Výber vhodnej podmnožiny I vstupných parametrov uskutočníme pomocou genetických algoritmov. Minimalizáciu hodnoty D_j môžeme zapísať:

$$I_{opt} = \arg\min_{I} D_j \tag{6.61}$$

Topológia RBF siete



Obrázok č. 18: Príklad RBF siete založenej na koncepte kontextového klastrovania. Obsahuje dva klastre pre každý kontext.

Štruktúru takejto RBF siete môžeme zapísať pomocou množiny pravidiel R^i, $i = 1,2, \dots j \times k$. Definujme i-té pravidlo R^i nasledovne:

R^i: ak vstupný vzor $\vec{x}$ náleží do k-tého klastra a j-tého kontextu, tak výstupnú hodnotu $\hat{y}_i$ určíme podľa vzťahu:

$$\hat{y}_i = m_j + a_i\left(\vec{x}_{(j)} - \vec{c}_k\right) \tag{6.62}$$

$\hat{y}_i$ je výstupná hodnota i-tého lokálneho modelu. m_j je modus kontextu j a $\vec{x}_{(j)}$ je vstupný vzor j-tého kontextu.

Ak uvážime všetky tieto pravidla R^i, môžeme vyjadriť celkový výstup siete nasledujúcim predpisom:

$$\begin{aligned}\hat{y}(\vec{x}) = &\sum_{i=1}^{k} u_i(\vec{x})\left[m_1 + a_i^T\left(\vec{x}_{(1)} - \vec{c}_i\right)\right] \\ &+ \sum_{i=k+1}^{2k} u_i(\vec{x})\left[m_2 + a_i^T\left(\vec{x}_{(2)} - \vec{c}_i\right)\right] + \cdots \\ &+ \sum_{i=(j-1)k+1}^{j\times k} u_i(\vec{x})\left[m_j + a_i^T\left(\vec{x}_{(j)} - \vec{c}_i\right)\right]\end{aligned} \tag{6.63}$$

Chybová funkcia, určujúca výkonnosť siete je definovaná predpisom:

$$\text{RMSE} = \sqrt{\frac{1}{P}\sum_{p=1}^{P}\left(\vec{y}_p - \hat{y}_p\right)^2} \tag{6.64}$$

6.9. Citlivostná analýza

Ďalším využívaným postupom pri návrhu RBF sieti je citlivostná analýza ([41]). Ešte pred samotnou konštrukciou RBF siete prevedieme citlivostnú analýzu a na základe výsledkov tejto analýzy určíme rozmiestnenie stredov jednotlivých RBF jednotiek. Citlivostná analýza nám pomáha určiť citlivosť celkového výstupu siete na zmenu jednotlivých parametrov v sieti. V prípade RBF sieti budeme pod pojmom citlivosť rozumieť matematický odhad druhej mocniny odchýlok siete spôsobených zmenou stredov RBF jednotiek.

Za radiálne bázické funkcie zvolíme Gaussovú funkciu definovanú ako:

$$\varphi(\vec{x}) = \exp\left(-\frac{\|\vec{x}-\vec{c}\|^2}{2\sigma^2}\right) \tag{6.65}$$

Výstup $\vec{y}$ RBF siete spočítame podľa predpisu:

$$y_j = \sum_{h=1}^{H} w_{hj}\,\varphi_h(\vec{x}), \qquad j = 1,..,O \tag{6.66}$$

$\vec{w}_{hj}$ je synaptická váha medzi h-tou RBF jednotkou a j-tým výstupným neurónom.

Nech $\hat{c}_h$ a $\hat{\sigma}_h$ označujú stred a šírku h-tej perturbovanej RBF jednotky. Výstup RBF jednotiek zo skrytej vrstvy označme ako vektor $\vec{v} = (v_1, v_2, \dots, v_H)$. Výslednú odchýlku Δy_j siete spôsobenú touto perturbaciou získame pomocou:

$$\Delta y_j = \hat{w}_j\hat{v} - \vec{w}_j\vec{v} =$$

$$= \sum_{h=1}^{H} \hat{w}_{hj} \exp\left(-\frac{\|\vec{x}-\hat{c}_h\|^2}{2\hat{\sigma}_h^2}\right) - \sum_{h=1}^{H} w_{hj} \exp\left(-\frac{\|\vec{x}-\vec{c}_h\|^2}{2\sigma_h^2}\right), \tag{6.67}$$

$$j = 1, \dots, O$$

$\hat{c}_h = \vec{c}_h + \Delta\vec{c}_h$ sú stredy RBF jednotiek odchýlené od stredov $\vec{c}_h$ perturbovaných RBF jednotiek a $\hat{w}_h = \vec{w}_h + \Delta\vec{w}_h$ sú synaptické váhy. $\vec{w}_h$ môžeme spočítať pomocou pseudo-inverznej metódy (6.4.1). Šírky σ_h RBF jednotiek sa zvyčajne všetky nastavia na nejakú konštantnú hodnotu, ktorá sa odvodiť na základe nejakých predchádzajúcich znalostiach riešeného problému**.**

Perturbaciu h-tej RBF jednotky, čiže odchylku stredu $\Delta\vec{c}_h$ a synaptických váh $\Delta\vec{w}_j$ spojených s j-tým výstupným neurónom môžeme vyjadriť pomocou Gaussovho rozdelenia s odchýlkou $\sigma_{\vec{c}_h}$ a $\sigma_{\vec{w}_j}$:

$$p(\Delta\vec{c}_h) = \frac{1}{\left(\sqrt{2\pi\sigma_{\vec{c}_h}}\right)^I} \exp\left(\frac{\Delta\vec{c}_h^T \Delta\vec{c}_h}{2\sigma_{\vec{c}_h}^2}\right)$$
$$p(\Delta\vec{w}_j) = \frac{1}{\left(\sqrt{2\pi\sigma_{\vec{w}_j}}\right)^H} \exp\left(\frac{\Delta\vec{w}_j^T \Delta\vec{w}_j}{2\sigma_{\vec{w}_j}^2}\right) \tag{6.68}$$

I je dimenzia trénovacích vzorov $\vec{x}$.

Definujme rekurzívne citlivosť ([41]). Nech máme $H-1$ pevne zvolených RBF stredov a chceme pridať ďalší stred. Citlivosť j-tého výstupného neurónu k aktuálne H RBF stredom definujeme ako $\left(\Delta y_j\right)^2$ (štvorec odchýlok siete spôsobených perturbaciou RBF stredov) z ohľadom na všetky $\Delta\vec{c}_h$ a trénovaciu množinu T. Citlivosť S_j^H môžeme definovať predpisom:

$$S_j^H = E\left[\left(\Delta y_j\right)^2\right] = \int p(\Delta\vec{w})\, p(\Delta\vec{c}) \tag{6.69}$$
$$\times \sum_{\vec{x}_p \in T} \sum_{m,n=1}^{H} \widehat{w}_{mj}\widehat{w}_{nj} \exp\left(-\frac{\left\|\vec{x}_p - \vec{c}_m - \Delta\vec{c}_m\right\|^2}{2\sigma_m^2} - \frac{\left\|\vec{x}_p - \vec{c}_n - \Delta\vec{c}_n\right\|^2}{2\sigma_n^2}\right) d\Delta\vec{c}\, d\Delta\vec{w}$$
$$-2\int p(\Delta\vec{w})\, p(\Delta\vec{c})$$
$$\times \sum_{\vec{x}_p \in T} \sum_{m,n=1}^{H} \widehat{w}_{mj} w_{nj} \exp\left(-\frac{\left\|\vec{x}_p - \vec{c}_m - \Delta\vec{c}_m\right\|^2}{2\sigma_m^2} - \frac{\left\|\vec{x}_p - \vec{c}_n - \Delta\vec{c}_n\right\|^2}{2\sigma_n^2}\right) d\Delta\vec{c}\, d\Delta\vec{w}$$
$$+\int p(\Delta\vec{w})\, p(\Delta\vec{c})$$
$$\times \sum_{\vec{x}_p \in T} \sum_{m,n=1}^{H} w_{mj} w_{nj} \exp\left(-\frac{\left\|\vec{x}_p - \vec{c}_m - \Delta\vec{c}_m\right\|^2}{2\sigma_m^2} - \frac{\left\|\vec{x}_p - \vec{c}_n - \Delta\vec{c}_n\right\|^2}{2\sigma_n^2}\right) d\Delta\vec{c}\, d\Delta\vec{w}$$
$$= C_1 - 2C_2 + C_3$$

$C_1,\ C_2,\ C_3$ po integrovaní podľa $\Delta\vec{c}$ a $\Delta\vec{w}$ vyjadríme:

$$C_1 = \sum_{\vec{x}_p \in T} \sum_{m,n=1;m\neq n}^{H} w_{mj} w_{nj} \frac{\left(\sqrt{\sigma_m^2 \sigma_n^2}\right)^I}{\left(\sqrt{\left(\sigma_m^2+\sigma_{\vec{c}_m}^2\right)\left(\sigma_n^2+\sigma_{\vec{c}_n}^2\right)}\right)^I}$$

$$\times \exp\left(-\frac{\left\|\vec{x}_p-\vec{c}_m\right\|^2}{2\left(\sigma_m^2+\sigma_{\vec{c}_m}^2\right)} - \frac{\left\|\vec{x}_p-\vec{c}_n\right\|^2}{2\left(\sigma_n^2+\sigma_{\vec{c}_n}^2\right)}\right) \tag{6.70}$$

$$+ \sum_{\vec{x}_p \in T} \sum_{m=1}^{H} \left(w_{mj}^2+\sigma_{\vec{w}_j}^2\right) \frac{\left(\sqrt{\sigma_m^2}\right)^I}{\left(\sqrt{\left(\sigma_m^2+\sigma_{\vec{c}_m}^2\right)}\right)^I} \times \exp\left(-\frac{\left\|\vec{x}_p-\vec{c}_m\right\|^2}{\sigma_m^2+\sigma_{\vec{c}_m}^2}\right)$$

$$C_2 = \tag{6.71}$$

$$\sum_{\vec{x}_p \in T} \sum_{m,n=1}^{H} w_{mj} w_{nj} \frac{\left(\sqrt{\sigma_m^2}\right)^I}{\left(\sqrt{\left(\sigma_m^2+\sigma_{\vec{c}_m}^2\right)}\right)^I} \times \exp\left(-\frac{\left\|\vec{x}_p-\vec{c}_m\right\|^2}{2\left(\sigma_m^2+\sigma_{\vec{c}_m}^2\right)} - \frac{\left\|\vec{x}_p-\vec{c}_n\right\|^2}{2\sigma_n^2}\right)$$

$$C_3 = \tag{6.72}$$

$$\sum_{\vec{x}_p \in T} \sum_{m,n=1}^{H} w_{mj} w_{nj} \exp\left(-\frac{\left\|\vec{x}_p-\vec{c}_m\right\|^2}{2\sigma_m^2} - \frac{\left\|\vec{x}_p-\vec{c}_n\right\|^2}{2\sigma_n^2}\right)$$

Najviac kritické vektory môžeme určiť pomocou vzťahu (6.69). Stredy RBF jednotiek nemôžu byť určené len na základe citlivosti, pretože niektoré z kritických vektorov môžu byť korelované. K odstráneniu tohto problému použijeme metódu ortogonálnych najmenších štvorcov OLS.

6.9.1. Ortogonálne najmenšie štvorce

Metódu ortogonálnych najmenších štvorcov označujeme ako OLS (Orthogonal Least Squares) ([16][41]). Nech matica $Y = (\vec{y}_1, \vec{y}_2, \dots, \vec{y}_P)^T$ obsahuje výstupy siete pre každý predložený trénovací vzor $\vec{x} \in T$. V našom prípade máme v maticovom zápise rovnicu:

$$Y = VW \tag{6.73}$$

pričom Y, V, W sú matice rozmerov $Y_{P\times O}, V_{P\times P}, W_{O\times P}$.

Každú maticu dokážeme faktorizovať na súčin dvoch matíc, takže predchádzajúci vzťah môžeme prepísať nasledovne:

$$Y = (QA)W \tag{6.74}$$

Matica V je zapísaná v tvare súčinu matíc $Q_{P\times P}$ a $A_{P\times P}$. Matica Q je ortogonálna matica a matica A je horná trojuholníková matica.

Chceme maticu V previesť na ortogonálny tvar, čiže všetky vektory budú navzájom na seba kolmé. V každom kroku sa ortogonalizuje jeden stĺpec matice. Ortogonalizačný proces môžeme popísať nasledujúcimi operáciami.

1. Prvý vektor sa zvolí priamo:

$$\vec{q}_1 = \vec{v}_1 \tag{6.75}$$

2. Pre $p = 2, \dots, P$ opakuj nasledujúce operácie

$$a_{ip} = \frac{\vec{q}_i^T \vec{v}_p}{\|\vec{q}_i\|}, \qquad 1 \le i < p \tag{6.76}$$

$$\vec{q}_p = \vec{v}_p - \sum_{i=1}^{p-1} a_{ip}\vec{q}_i \tag{6.77}$$

6.9.2. Algoritmus citlivostnej analýzy

1. Inicializácia:
   Za hodnoty jednotlivých prvkov v matici V vo vzťahu $Y = VW$ dosaď výsledné hodnoty RBF funkcií pre každý trénovací vzor.

2. Voľba prvého kritického vektoru:
   Spočítaj hodnotu citlivosti pre každý stĺpec matice V pomocou vzťahu (6.69).
   - Stĺpec s najväčšou hodnotou citlivosti zvolíme sa za prvý stĺpec matice $Q^{(1)}$.
   - Spočítaj odchýlku siete $Error^{(1)}$ pre zvolený RBF stred.
   - Nastav $H = 2$.

3. Ortogonalizácia stĺpcov matice V:
   Preveď proces ortogonalizácie metódou OLS (0) so všetkými stĺpcami matice V s využitím stĺpcov matice $Q^{(H-1)}$.

4. Výpočet:
   Pre každý trénovací vzor uvažuj $\vec{c}_p$ ako kandidáta na H-ty stred RBF jednotky, ktorý odpovedá ortogonalnému stĺpcu $\vec{q}_p$, ($H \leq p \leq P$).
   - Spočítaj hodnoty synaptických váh pomocou pseudo-inverznej metódy (6.4.1).
   - Spočítaj hodnoty citlivosti $S^{(H)}(\vec{c}_p)$ predchádzajúcich $H-1$ RBF stredov, pričom hodnoty synaptických váh sú aktualizované podľa vzťahu:

$$w_{pj}^{(H)} = \sum_{r=1}^{P} a_{rp} w_{pj} \tag{6.78}$$

   - Zotrieď stĺpce matice $Q^{(H)}$ podľa:

$$\left\| S^{(H)}(\vec{c}_1) \right\| \geq \left\| S^{(H)}(\vec{c}_2) \right\| \geq \cdots \geq \left\| S^{(H)}(\vec{c}_P) \right\| \tag{6.79}$$

   - Stĺpec s najväčšou hodnotu citlivosti sa zvolí za H-tý stĺpec matice $Q^{(H)}$.
   - Spočítaj odchýlku siete $Error^{(H)}$ pre zvolené RBF stredy.

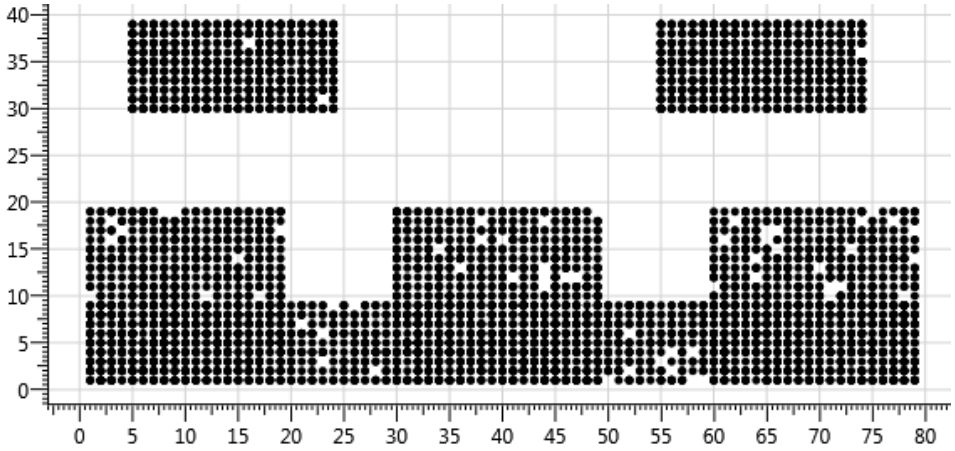
5. Ukončenie algoritmu:
   Ak hodnota rozdielu $Error^{(H)} - Error^{(H-1)}$ je menšia ako nejaká preddefinovaná hodnota, tak ukonči algoritmus. V opačnom prípade zvýš hodnotu $H = H + 1$ a pokračuj na 3. krok algoritmu.

Kritické vektory odpovedajúce prvým H stĺpcom matice $Q^{(H)}$ sa zvolia sa za stredy RBF jednotiek.

7. Experimentovanie s dátami

7.1. Umelo vygenerované dáta

Testovacia množina obsahuje 7 tisíc vstupných vzorov. Rozsah hodnôt, z ktorého boli jednotlivé zložky vstupných vzorov generované, bol prispôsobený tak, aby výsledné rozmiestnenie týchto vzorov vytvorilo určitú geometrickú štruktúru. Vstupné vzory sú tvorené dvomi zložkami, ktoré vyjadrujú x-ovú a y-ovú súradnicu v dvojrozmernom priestore, a to z dôvodu umožnenia jednoduchej vizualizácie dosiahnutých výsledkov. Obrázok č. 19 zachytáva rozmiestnenie vzorov z testovacej množiny. Testovacia množina bola úmyselne vytvorená tak, aby neobsahovala len jasne oddelené zhluky dát, ale aj vzájomne sa prekrývajúce zhluky dát.



Obrázok č. 19: Zobrazenie vstupných vzorov testovacej množiny.

Pred samotným testovaním klastrovacích algoritmov je potrebné vykonať určité predspracovanie vstupných dát. Bežne používanou metódou predspracovania dát je normalizácia vstupných dát. Existuje viacero normalizačných metód, my budeme používať min-max normalizáciu na interval $< A, B >$. Min-max normalizácia lineárne transformuje pôvodné dáta typicky na interval $< 0, 1 >$. Transformáciu prevedieme po zložkách testovacích vzorov podľa nasledujúceho vzťahu:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \tag{7.1}$$

x je pôvodná hodnota, $x_{\max}$ je maximálna hodnota cez všetky vzory a $x_{\min}$ je minimálna hodnota cez všetky vzory. Aby sme mohli vykonať min-max normalizáciu, potrebujeme poznať minimálne a maximálne hodnoty testovacej množiny. Takže v prvom priechode

testovacou množinou určíme extrémne hodnoty a v druhom prechode testovacou množinou vykonáme samotnú min-max normalizáciu.

Proces normalizácie vstupných dát je obzvlášť dôležitý pri klastrovacích úlohách, a to z dôvodu, že tieto klastrovacie metódy počas svojho priebehu počítajú vzdialenosť jednotlivých vstupných vzorov od stredu zhluku. Prevedením normalizácie vstupných dát predídeme vzniku atribútov s veľkými hodnotami.

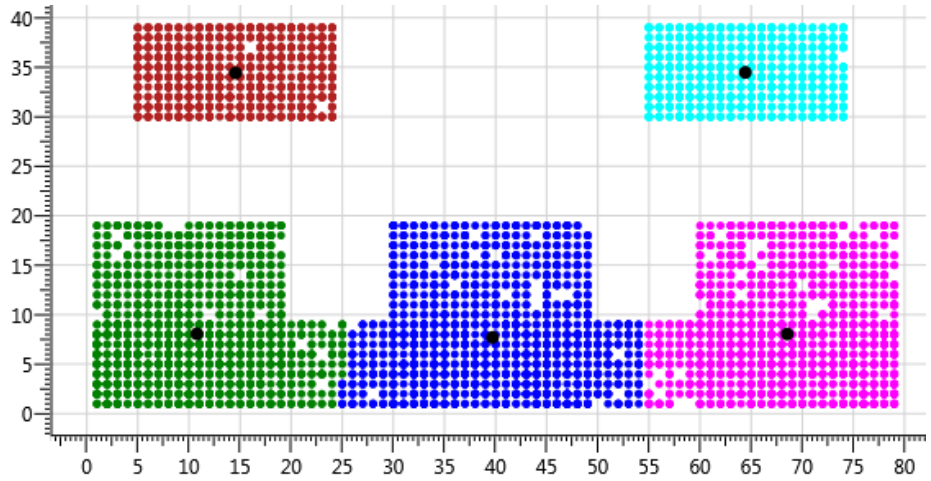
Po prevedení min-max normalizácie vstupných dát testovacej množiny môžeme prejsť k testovaniu niektorých z uvedených klastrovacích algoritmov. V obrázkoch, ktoré budú v nasledujúcom texte uvedené, budeme vstupné vzory znázorňovať farebne a stredy zhlukov budú znázorňovať body čiernej farby.

K-means algoritmus

K-means algoritmus rozdeľuje vstupné vzory do navzájom disjunktných klastrov. Ako je u K-means algoritmu známe, často sa zastaví už v nejakom lokálnom optime a z toho dôvodu nenájde optimálne riešenie. Vďaka pomerne veľkej rýchlosti tohto algoritmu môžeme tento problém čiastočne odstrániť viacnásobným spustením algoritmu na rovnakých dátach, avšak s iným náhodným rozložením centroidov. Na zvolenej testovacej množine teda spustíme tento algoritmus jeden tisíckrát, pričom budeme sledovať k akým výsledkom sa dopracujeme. Najskôr sme pomocou K-means algoritmu hľadali optimálny počet klastrov v rozmedzí od 1 do 7. Tabuľka č. 1 vyjadruje pre každý počet klastrov koľkokrát bol tento počet zvolený za optimálne riešenie. V ďalšom kroku sme zväčšili skúmaný rozsah klastrov na 1 až 10. Tabuľka č. 2 vyjadruje počty pokusov, v ktorých bol príslušný počet klastrov označený za optimálny.

Tabuľka č. 1: Počet prípadov, v ktorých K-means algoritmus určil jednotlivé počty klastrov za optimálne riešenie. Optimálny počet klastrov sa určuje pomocou uhlového kritéria. Počet všetkých pokusov je 1000.

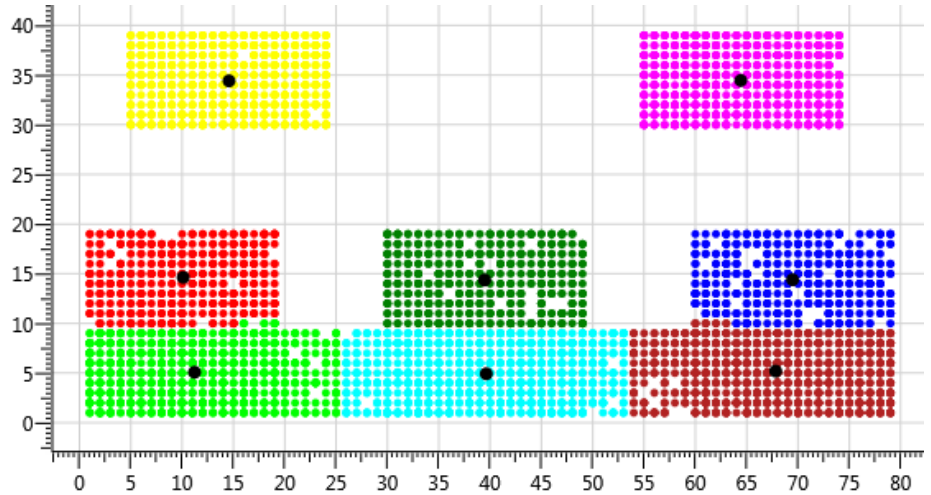
Počet klastrov	1	2	3	4	5	6	7
Počet označení za optimum	0	0	0	86	669	239	6



Obrázok č. 20: Rozdelenie testovacích dát do piatich klastrov pomocou K-means algoritmu.

Tabuľka č. 2: Počet prípadov, v ktorých K-means algoritmus určil jednotlivé počty klastrov za optimálne riešenie. Optimálny počet klastrov určujeme pomocou uhlového kritéria. Počet všetkých pokusov je 1000.

Počet klastrov	1	2	3	4	5	6	7	8	9	10
Počet označení za optimum	0	0	0	35	187	78	154	244	246	56



Obrázok č. 21: Rozdelenie testovacích dát do ôsmych klastrov pomocou K-means algoritmu.

Pri prvom testovaní, pre rozsah klastrov 1 až 7, bol pomerne jednoznačne zvolený optimálny počet klastrov rovný 5, a to v 669 prípadoch z tisíca. Druhý najčastejšie označený optimálny počet klastrov bol rovný 6. Tento počet klastrov však už bol označený v takmer 3 krát menšom počte, konkrétne v 239 prípadoch z tisíca.

Po zväčšení skúmaného rozsahu klastrov na 1 až 10 už také jednoznačné výsledky neboli dosiahnuté. Pre počet klastrov 8 a 9 sme dostali takmer identické hodnoty, a to 244 a 246. Takže za optimálny počet klastrov si prakticky môžeme vybrať medzi 8 a 9, pričom je dobré riadiť sa pravidlom, že ak s menším počtom klastrov získame kvalitatívne porovnateľné rozdelenie vstupných vzorov do klastrov ako pri väčšom počte klastrov, zvolíme menší počet klastrov. Všimnime si, že pomerne veľa krát bol za optimálny počet klastrov zvolený už spomínaný počet klastrov rovný 5, a to 187 krát.

Táto hodnota 187 pre počet klastrov 5 je v konečnom dôsledku pomerne výrazná, keď zoberieme v úvahu, že za optimálny počet bol zvolený takmer dvojnásobný počet klastrov 9, a to v celkom porovnateľnom množstve pokusov 246. V konečnom dôsledku takýmto pomerne výrazným zväčšením počtu klastrov nič nepokazíme, ale často ani nezlepšíme kvalitu výsledného klastrovania. Odporúčame teda začať s testovaním menšieho počtu klastrov a postupne tento počet klastrov do rozumnej miery zväčšovať.

FCM algoritmus

FCM algrotimus narozdiel od K-means algortimu umožňuje, aby vstupný vzor patril do dvoch alebo viacerých klastrov zároveň. Pri FCM algoritme sme spomínali fuzzyfikačný parameter m. Tento parameter určuje ako veľmi sa mení príslušnosť vstupných vzorov ku klastrom vzhľadom k ich vzdialenosti od centroidov jednotlivých zhlukov. Zmienili sme sa o tom, že v literatúre zaoberajúcej sa problematikou FCM klastrovania býva bežne hodnota fuzzyfikačného parametra nastavená na hodnotu 2. Otestujeme preto na našej testovacej množine FCM algoritmus s rôznymi hodnotami fuzzyfikačného parametra, konkrétne s hodnotami $m = \{1.1, 1.5, 2.0, 2.5\}$.

Podmienkou k ukončeniu FCM algoritmu je, že maximálna hodnota zmien pozícií centroidov jednotlivých zhlukov medzi jednotlivými iteráciami algoritmu klesne aspoň na hodnotu ukončovacieho kritéria ε. Bežne sa nastavuje hodnota tohto kritéria na hodnotu 0.05.

Kvalitu rozdelenia vstupných vzorov z testovacej množiny do klastrov budeme hodnotiť podľa pozmenenej verzie koeficientu členskej funkcie, ktorá je zadaná predpisom:

$$F'_K(U) = 1 - \frac{K}{K-1}\left(1 - F_K(U)\right) \tag{7.2}$$

$F_K(U)$ predstavuje koeficient členskej funkcie zadaný predpisom:

$$F_K(U) = \frac{\sum_{i=1}^{P}\sum_{j=1}^{K} u_{ij}^2}{P} \tag{7.3}$$

Tabuľka č. 3: Počet prípadov, v ktorých FCM algoritmus určil jednotlivé počty klastrov za optimálne riešenie. Počet všetkých pokusov je rovný 100.

Fuzzyfikačný parameter m	Ukončovacie kritérium ε	Priemerný čas trvania 1 pokusu (s)	Počet klastrov						
			1	2	3	4	5	6	7
1.1	0.05	76	0	0	0	5	89	6	0
1.1	0.1	65	0	0	0	3	88	8	1
1.5	0.05	78	0	0	0	1	98	1	0
1.5	0.1	63	0	0	0	1	97	2	0
2.0	0.05	79	0	0	0	41	59	0	0
2.0	0.1	66	0	0	0	50	46	2	2
2.5	0.05	93	0	0	0	57	42	1	0
2.5	0.1	69	0	0	0	30	57	12	1

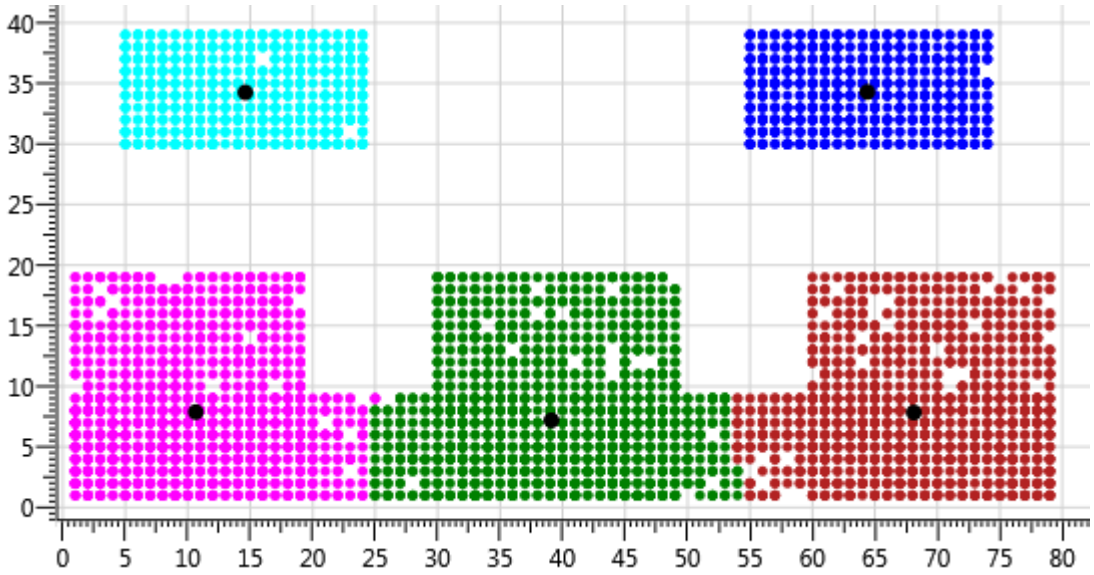
Z výsledkov uvedených v tabuľke vyplýva, že hodnota fuzzyfikačného parametra má nezanedbateľný vplyv na konečné výsledky FCM algoritmu. Z tabuľky vidíme, že pre nižšie hodnoty fuzzyfikačného parametra m, konkrétne 1.1 a 1.5 bol jednoznačne zvolený optimálny počet klastrov rovný 5. Tento počet klastrov bol zvolený za optimálny až v 98 prípadoch zo sto pri hodnote parametra $m = 1.5$.

Pri vyšších hodnotách fuzzyfikačného parametra 2.0 a 2.5 už nie sú výsledné hodnoty také jednoznačné. Za optimálny počet klastrov sa pri týchto hodnotách fuzzyfikačného parametra ponúkajú dve možnosti, a to 4 a 5. Tieto dve možnosti boli označené približne v rovnakom počte. Síce sme v tomto prípade nezískali jednoznačné určenie optimálneho počtu klastrov, ale boli určené dve možnosti, z ktorých jedna bola označená pri nižších hodnotách fuzzyfikačného paratmetra m za optimálny počet klastrov a druhá možnosť preferuje počet klastrov len o jedna menší ako bol zvolený za

optimálny počet pri nižších hodnotách fuzzyfikačného parametra. Čiže vyberáme medzi počtom klastrov, ktorý je blízko k optimalnému počtu klastrov.

Rozdielna hodnota ukončovacieho kritéria $\varepsilon = \{0.05, 0.1\}$ pri jednotlivých hodnotách fuzzyfikačného parametra m nepriniesla žiadnu výraznu zmenu pri určovaní optimálneho počtu klastrov. Avšak zmena hodnoty ε vplýva na výsledný čas trvania FCM algoritmu, čím menšia hodnota ukončovacieho kritéria je volená, tým väčší počet iterácií FCM algoritmu sa musí vykonať k splneniu ukončovacej podmienky.

Najlepšie výsledky na testovacej množine sme dosiahli pri kombinácií parametrov $m = 1.5$ a $\varepsilon = 0.05$, z tohto dôvodu pri ďalších príkladoch klastrovania pomocou FCM algoritmu budeme pracovať práve s týmito hodnotami.



Obrázok č. 22: Rozdelenie testovacích dát do piatich klastrov pomocou FCM algoritmu pri hodnotách paramametrov $m = 1.5$ a $\varepsilon = 0.05$.

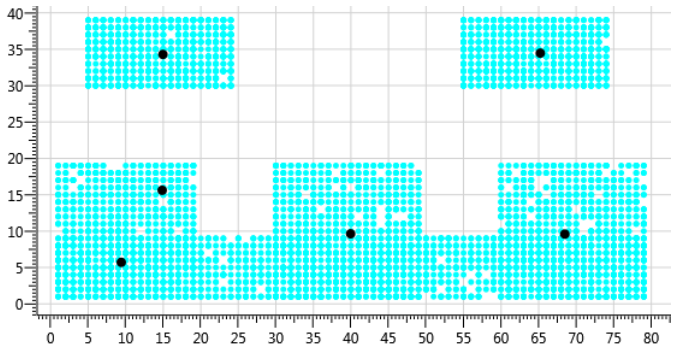
FCM algoritmus sme spustili 100 krát na dátach z testovacej množiny. Dôvodom k nižšiemu počtu pokusov oproti K-means algoritmu, ktorý sme spustili 1000 krát na dátach z testovacej množiny, je väčšia časová náročnosť FCM algoritmu. Jedno spustenie K-means algoritmu, pri ktorom sa určil optimálny počet klastrov z rozsahu 1-7 trval v priemere približne 6 sekúnd, zatiaľ čo pri FCM algoritme jedno spustenie trvalo v priemere približne 70 sekúnd.

Kohonenová mapa

V procese učenia Kohonenovej mapy sa postupne aktualizujú pozície neurónov v mriežke, tak aby na konci učenia boli tieto neuróny čo najlepšie rozmiestnené do oblastí vstupného priestoru, kde sa vyskytujú vstupné vzory. Vstupný priestor je tvorený vstupnými vzormi z testovacej množiny.

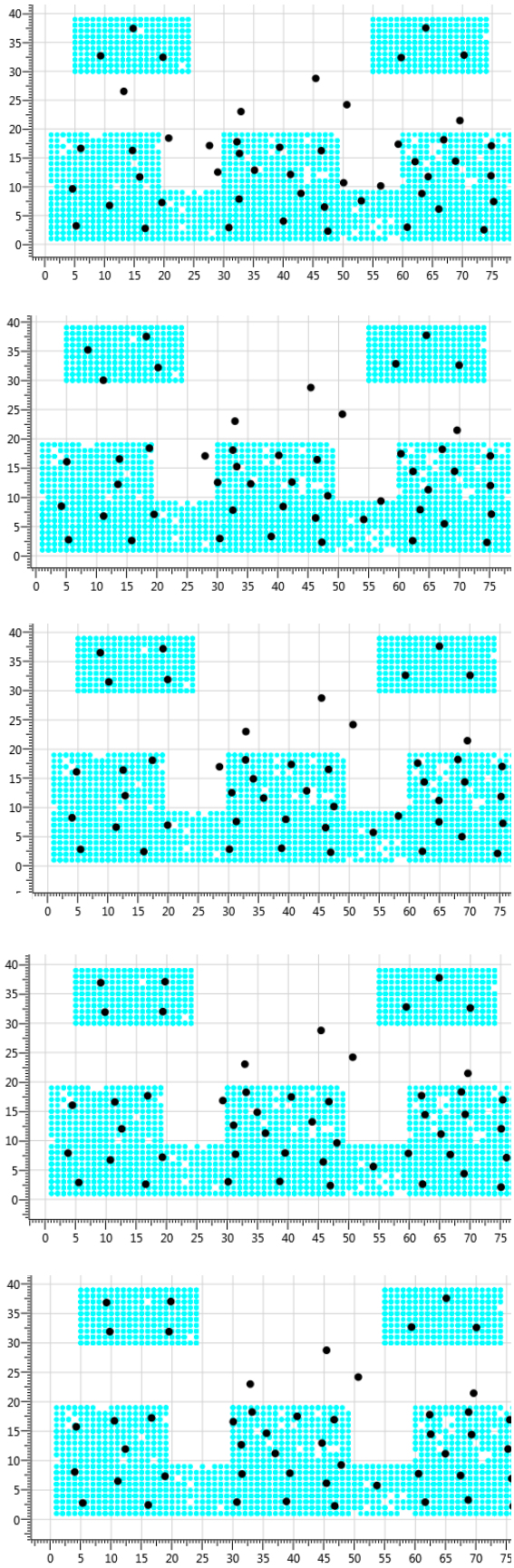
Pri voľbe veľkosti mriežky môžeme postupovať dvomi spôsobmi. V prvom prípade, zvoľme menší rozmer mriežky tak, aby počet neurónov v nej obsiahnutých približne odpovedal očakávanému počtu klastrov testovacej množiny. Pomocou K-means a FCM algoritmov sme zatiaľ zistili, že pre našu testovaciu množinu je optimálny počet klastrov rovný 5. Obrázok č. 23 zobrazuje rozmiestnenie neurónov v mriežke, ktorá má rozmery 2×3. Po skončení učiaceho procesu Kohonenovej mapy môžeme tieto neuróny z mriežky priamo prehlásiť za reprezentantov jednotlivých klastrov.

Na ďalej uvedených obrázkoch sú vstupné vzory znázornené farebne a neuróny v mriežke reprezentujú čierne body. To samé platí aj pre Kohonenové mapy s rastúcou mriežkou a pre model rastúcich neurónových plynov.



Obrázok č. 23: Kohonenová mapa s mriežkou o rozmeroch 2×3.

V praxi sa však takto nízke rozmery mriežky veľmi nepoužívajú. V druhom prípade zvoľme teda väčšie rozmery mriežky, napríklad 7×7. Pri takto zvolených rozmeroch mriežky pracujeme so 49 neurónmi, takže určiť počet a rozmiestnenie klastrov v testovacej množine už nemôže urobiť priamo ako v prvom prípade. Môžeme však postupovať nasledovne. Neuróny v mriežke rozdelíme do klastrov pomocou nejakého klastrovacieho algoritmu. Potom všetky vstupné vzory testovacej množiny také, že po ich predložení na vstup Kohonenovej mape zvíťazili v kompetícii neuróny z jedného klastra neurónov, tvoria jeden klaster v testovacej množine.



Obrázok č. 24: Kohonenová mapa o rozmeroch mriežky 7×7. Jednotlivé obrázky zachytávajú stav učenia Kohonenovej mapy po 20000, 40000, 60000, 80000, 10000 predložených vzoroch z testovacej množiny počas záverečnej dolaďovacej fázy.

Kohonenová mapa s rastúcou mriežkou

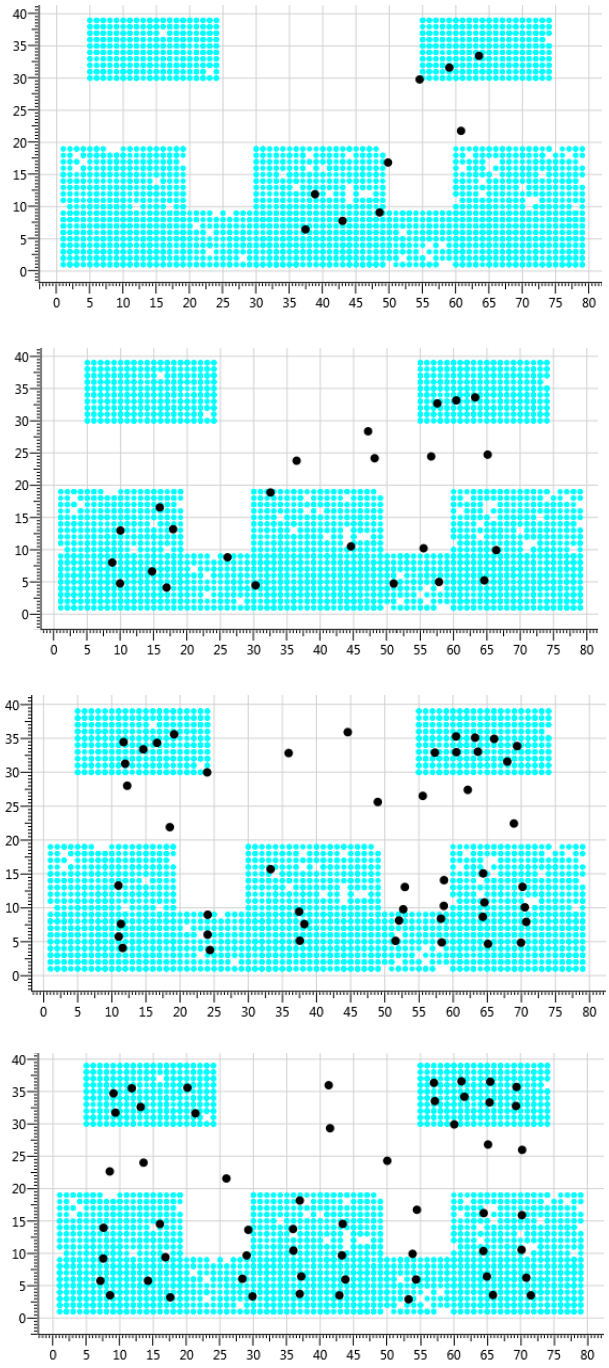
Kohonenové mapy s rastúcou mriežkou sú variantom Kohonenových máp umožňujúcim počas procesu učenia meniť veľkosť topologickej štruktúry (mriežky) výstupných neurónov. Počas učiaceho procesu vždy po vykonaní vopred stanoveného počtu iterácií sa do mriežky pridáva riadok, prípadne stĺpec nových neurónov. Toto zväčšovanie počtu neurónov v mriežke je vykonávané dovtedy, kým počet týchto neurónov nedosiahne ich vopred stanovený počet. Keďže sa v tomto modeli Kohonenových máp pridáva do mriežky v jednom okamihu viacero neurónov súčasne, tak výsledný počet neurónov obsiahnutých v mriežke je často väčší ako vopred stanovený maximálny počet neurónov.

V priebehu učiaceho procesu majú všetky parametre konštantnú hodnotu. Konkrétne sa jedná o parameter učenia η_0 a parameter šírky Gaussovej funkcie σ.

Keďže hodnota parametra učenia η_0 je konštantná počas celého procesu učenia, jej hodnota musí byť primerané malá. Ak by hodnota η_0 bola nastavená na príliš vysokú hodnotu, dochádzalo by k neustálym výrazným zmenám pozícií neurónov v mriežke (reakcia neurónov na predložené vstupné vzory by bola výrazná). Na druhu stranu pri príliš malých hodnotách parametra učenia η_0 by bola aktualizácia pozícií neurónov v mriežke nevýrazná.

Ak si spomenieme na učiaci proces klasických Kohonenových máp, tento proces sme delili na fázu hrubého učenia a fázu dolaďovania. Tieto dve fázy sa líšia napríklad hodnotou parametra učenia, počas hrubého učenia hodnota tohto parametra klesá z hodnoty približne 1 na hodnotu 0.02 a v dolaďovacej fáze pokles hodnoty tohto parametra pokračuje ([22]). To znamená, že hodnoty parametra učenia okolo 0.02 sú hraničné hodnoty tohto parametra, pri ktorých sa zmena pozícií neurónov z výraznej začína meniť na jemnejšiu. A práve takúto hodnotu pre parameter učenia η_0 hľadáme. Zvoľme teda hodnotu parametra učenia napríklad $\eta_0 = 0.05$. Len pre porovnanie v literatúre ([10]) zaoberajúcej sa týmto variantom Kohonenových máp, pracovali s hodnotami parametrov $\eta_0 = 0.1$ a $\sigma = 0.9$. Za hodnotu šírky σ môžeme nastaviť uvedenú hodnotu 0.9.

Pri Kohonenových mapách s rastúcou mriežkou môžeme podobne ako v prípade klasických Kohonenových máp vykonať dolaďovaciu fázu. Dĺžka tejto fázy by mala byť dostatočne dlhá, odporúča sa 10 tisíc až 100 tisíc iterácií. Parameter učenia v tejto fáze môže napríklad lineárne klesať s počtom vykonaných iterácií učiaceho procesu. V prípade, že chceme dodržať pravidlo s konštantnými hodnotami všetkých parametrov počas celého priebehu učenia, dosadíme za parameter učenia nejakú nízku hodnotu, napríklad 0.005.



Obrázok č. 25: Kohonenová mapa s rastúcou mriežkou. Nové neuróny sa vkladajú do mriežky dovtedy, pokiaľ je ich počet menší ako 50. Učiaci proces začínal s mriežkou rozmerov 2×2. Jednotlivé obrázky zachytávajú rozmiestnenie neurónov v mriežke po pridaní nových neurónov, pričom počet týchto neurónov je v danom okamihu postupne 9, 24, 49, 56. Posledný obrázok, kde mriežka obsahuje 56 neurónov, zobrazuje stav Kohonenovej mapy s rastúcou mriežkou už po prebehnutí dolaďovacej fázy, ktorá zahŕňala 10 tisíc adaptačných krokov. Hodnoty parametrov $\eta_0 = 0.05$, $\sigma = 0.9$, $\lambda_r = 30$.

Model rastúcich neurónových plynov

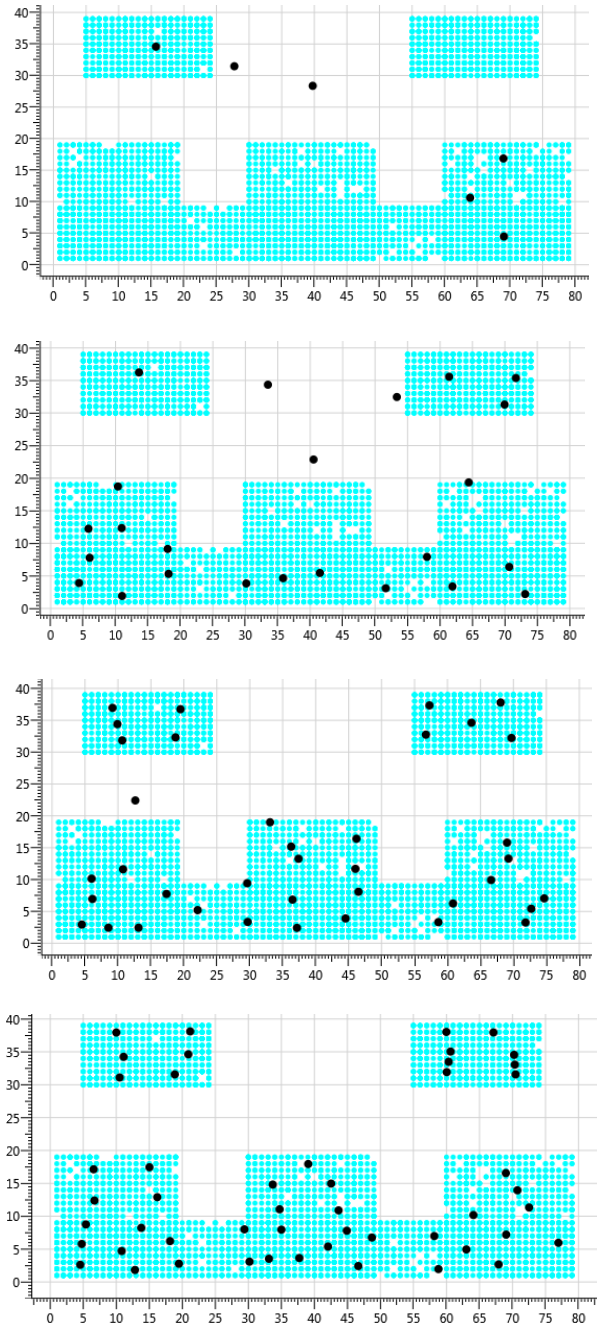
Model rastúcich neurónových plynov je ďalším variantom Kohonenových máp umožňujúcim počas procesu učenia meniť veľkosť topologickej štruktúry (mriežky) výstupných neurónov. Na rozdiel od Kohonenových máp s rastúcou mriežkou sa v tomto modeli pridávajú neuróny do mriežky po jednom. Zväčšovanie počtu neurónov v mriežke sa vykonáva dovtedy, kým počet neurónov nedosiahne vopred stanovený počet.

Aj v tomto modeli Kohonenových máp majú v priebehu učiaceho procesu všetky parametre konštantnú hodnotu. Parameter učenia však nadobúda dve rôzne hodnoty, prvá hodnota ϵ_b sa používa pri aktualizácií váhového vektora víťazného neurónu a druhá hodnota ϵ_n sa používa pri aktualizácií váhových vektorov priamych susedov víťazného neurónu. Pre tieto hodnoty platí vzťah $1 > \epsilon_b \gg \epsilon_n$. Po víťaznom neuróne požadujeme, aby sa priblížil viac k predloženému vzoru ako neuróny z jeho okolia.

Tieto parametre môžeme inicializovať hodnotami, ktoré sme už spomínali pri Kohonenových mapách s rastúcou mriežkou. Za parameter učenia ϵ_n dosadíme nejakú nízku hodnotu, napríklad $\epsilon_n = 0.005$ (hodnota parametra učenia z dolaďovacej fázy Kohonenových máp s rastúcou mriežkou). Za hodnotu parametra učenia pre víťazne neuróny zvolíme hodnotu 0.05.

Pre porovnanie v literatúre ([9]) zaoberajúcej sa týmto variantom Kohonenových máp, pracovali s nasledujúcimi hodnotami parametrov $\epsilon_n = 0.006,\ \epsilon_b = 0.2.$

Za zvyšné hodnoty parametrov dosadíme hodnoty uvedené v ([9]), konkrétne
$vek_{\max} = 50,\ \alpha = 0.5,\ \beta = 0.995,\ \lambda = 100.$



Obrázok č. 26: Model rastúcich neurónových plynov. Nové neuróny sa vkladajú do mriežky dovtedy, pokiaľ je ich počet menší ako 50. Učiaci proces začínal s mriežkou rozmerov 1×2. Jednotlivé obrázky zachytávajú rozmiestnenie neurónov v mriežke po pridaní nových neurónov, pričom počet týchto neurónov je v danom okamihu postupne 6, 24, 38, 50. Hodnoty parametrov $\lambda = 100$, $\epsilon_n = 0.005$, $\epsilon_b = 0.2$, $\alpha = 0.5$, $\beta = 0.995$, $vek_{\max} = 50$.

Po spustení uvedených modelov Kohonenových máp na testovacej množine vstupných vzorov sme získali výsledné rozmiestnenia neurónov v mriežke, ktoré sú zachytené na vyššie uvedených obrázkoch 24-26. Všetky tri modely mali približne rovnaký počet neurónov vo výslednej mriežke, a to 49, 56, 50 v poradí v akom sme tieto modely predstavovali. Neuróny v mriežke sme rozdelili do klastrov pomocou K-means algoritmu. Uvedené výsledky môžeme zhrnúť nasledovne.

Pri klasických Kohonenových mapách po skončení učiaceho procesu len málo neurónov v mriežke nebolo umiestnených do oblasti výskytu vstupných vzorov. V takmer 75% bola testovacia množina rozdelená do 5 klastrov a v 8% do 6 klastrov.

Kohonenové mapy s rastúcou mriežkou majú tú nevýhodu, že z dôvodu udržania tvaru topológie neurónov v mriežke sa v jednom kroku pridáva do mriežky viacero neurónov súčasne. Z tohto dôvodu sa vo výslednej mriežke objavilo viacero neurónov, ktoré nie sú plne využité. Testovacia množina v 69% bola rozdelená do 5 klastrov a v 22% do 6 klastrov.

Model neurónových plynov môže pridávať neuróny do mriežky po jednom neuróne, čím získava veľkú výhodu oproti ostatným uvedeným modelom. Ďalšou veľkou výhodou tohto modelu je, že môže počas procesu učenia niektoré nevyužité neuróny odstrániť z mriežky. Napriek tomu, že na tejto testovacej množine nedošlo k odstraňovaniu neurónov z mriežky, tak tento model Kohonenových máp rozmiestnil neuróny v priestore vstupných vzorov, v zmysle využitia týchto neurónov, najlepším spôsobom. Testovacia množina bola v 84% rozdelená do 5 klastrov.

7.2. Reálne dáta

V tejto časti budeme pracovať s reálnymi dátami z oblasti svetovej ekonomiky, pričom sa zameriame hlavne na vzájomný obchod medzi jednotlivými štátmi sveta. Na základe objemu uskutočnených obchodných transakcií medzi štátmi budeme skúmať prepojenie ekonomík jednotlivých štátov.

Vstupné dáta popisujúce vzájomný obchod jednotlivých štátov sveta sú dostupné z internetovej stránky Medzinárodného menového fondu ([48]). Získané dáta z tohto zdroja sú vo forme dátovej matice. Riadky tejto matice odpovedajú štátom, ktoré sú exportérmi a stĺpce matice odpovedajú štátom, do ktorých sa dováža. Konkrétne prvok matice, ktorý je na i-tom riadku a v j-tom stĺpci vyjadruje objem exportu zo i-tého štátu do štátu j. Objem všetkých obchodov je vyjadrený v miliónoch amerických dolárov.

Získané dáta musíme ešte pred samotným použitím čiastočne upraviť. Štáty, u ktorých nemáme uvedené žiadne informácie o vzájomnom obchode so zvyšnými štátmi, nemusíme vôbec brať v úvahu. Z dátovej matice teda odstránime prázdne riadky a stĺpce. Z pôvodných 198 štátov nám po takomto prečistení zostane 179 štátov sveta.

Ešte pred samotným testovaním dát sa skúsme zamyslieť aké výsledky dostaneme po aplikovaní jednotlivých klastrovacích metód. Chceme rozdeliť štáty sveta do jednotlivých klastrov, pričom poznáme veľkosť objemu vzájomných obchodov týchto štátov. V jednotlivých klastroch by sa teda mali objaviť štáty s podobným objemom zrealizovaných obchodov. Je zrejme, že množstvo týchto zrealizovaných obchodov súvisí s aj veľkosťou jednotlivých štátov, pretože väčšie štáty budú vo väčšom množstve vyvážať aj dovážať.

Na vstupných dátach po prečistení prevedieme proces normalizácie, pričom bola použitá min-max normalizačná metóda. Takto pripravené dáta môžeme následne predložiť na vstup jednotlivým klastrovacím metódam.

K-means a FCM algoritmus

K-means algoritmus po viacnásobnom spustení označil za optimálne rozdelenie vstupných vzorov do 6 klastrov. Odlišné výsledky sme dostali po viacnásobnom prevedení FCM algoritmu. FCM algoritmus preferuje optimálny počet klastrov 2. Na prvý pohľad vyzerá, že rozdelenie do 2 klastrov nám veľa o vstupnej množine nenapovie. FCM algoritmus však nerozdelí štáty zhruba na dve polovice, ale vytvára skupinu obsahujúcu 10-15 štátov s najväčšími objemami prevedených obchodov. Získavame takto predstavu o štátoch, ktoré majú najväčší podiel na celosvetovom objeme obchodov. Uveďme si hlavných predstaviteľov z tejto skupiny ekonomicky silných štátov:

Čína, Francúzsko, Nemecko, India, Taliansko, Japonsko, Rusko, Singapur, Veľká Británia, USA

Zloženie tejto skupiny nám potvrdia aj výsledky ďalších klastrovacích metód. Táto skupina silných štátov môže byť v niektorých prípadoch po podrobnejšom preskúmaní ešte rozdelená do viacerých menších klastrov. Druhú skupinu tvoria zvyšné štáty sveta, o ktorých sme sa pri takto nastavenom klastrovaní skutočne nič nedozvedeli.

Uveďme výsledky K-means algoritmu, ktorý preferoval rozdelenie štátov do 6 klastrov. Prvé dva klastre sú tvorené už spomínanými štátmi. Konkrétne prvý klaster tvorí Čína, Nemecko, USA a druhý klaster tvoria štáty Taliansko, Japonsko, Južná Kórea, Holandsko, Rusko, Veľká Británia.

Tretí a štvrtý klaster majú spoločnú vlastnosť, že zoskupujú štáty z určitej oblasti sveta. Čo je samozrejme logické, že štáty, ktoré sú k sebe geograficky bližšie, uzatvoria vzájomne väčší objem obchodov. Tretí klaster je tvorený štátmi Argentína, Brazília, Čile, Kolumbia a Ekvádor. Štvrtý klaster tvoria štáty India, Malajzia, Saudská Arábia, Singapur, Thajsko, Spojené Arabské Emiráty. Štáty 4-tého klastra sa jednak nachádzajú v rovnakej oblasti sveta, ale všetky tieto štáty môžeme považovať za prístavné štáty. Zrejme ich strategická poloha v lodnej preprave výrazne zvyšuje objem uzatvorených obchodov.

Piaty klaster tvoria štáty Austrália a Brazília. Opäť veľké štáty, ktoré vzájomne obchodujú s veľkým počtom štátov. Šiesty klaster je tvorený zvyšnými štátmi sveta.

Aby sme získali aspoň nejaké informácie o zatiaľ nespomenutých štátoch, zväčšíme rozsah klastrov, z ktorého K-means algoritmus hľadá optimálny počet klastrov. Zvýšme tento rozsah na 15 klastrov. Po viacnásobnom spustení K-means algoritmu bol najčastejšie označený za optimálny počet klastrov 10. Po rozdelení štátov do desiatich klastrov sme opäť získali klastre obsahujúce už spomínané štáty, prípadne len s minimálnou zmenou jednotlivých štátov. Uveďme však niektoré novo vytvorené klastre.

Určite je predmetom záujmu určiť, do skupiny ktorých štátov sa radí Česká a Slovenská Republika. V klastri obsahujúcom Českú Republiku sa veľmi často vyskytujú štáty Rakúsko, Dánsko, Fínsko a Belgicko.

Slovenská Republika sa vyskytuje v jednom klastri najčastejšie so štátmi Maďarsko, Portugalsko, Švédsko, Rumunsko, Ukrajina.

FCM algoritmus po zvýšení požadovaného počtu klastrov na 10 produkuje výsledky veľmi podobné výsledkom K-means algoritmu.

Modely Kohonenových máp

Musíme brať v úvahu, že máme pomerne malý počet vstupných vzorov, a to len 179. Z tohto dôvodu treba voliť menšie rozmery mriežky. Pri ponechaní iniciálnych rozmerov mriežky 7×7 boli štáty rozdelené vo väčšine prípadov do dvoch klastrov. Postupným zmenšovaním rozmerov mriežky sme sa dostali až k mriežke o rozmeroch 3×3. Pri takýchto malých rozmeroch mriežky sme totižto najčastejšie získali rozdelenie štátov do viacerých klastrov. Uveďme si jeden z výsledkov rozdelenia štátov do klastrov pomocou Kohonenovej mapy.

Prvé tri klastre sú väčšinou tvorené už spomínanými najsilnejšími štátmi. Avšak rozdelenie týchto silných štátov je oproti doteraz spomínaným algoritmom viac ustálene.

Prvý klaster veľmi často tvorili dva štáty, a to Japonsko a USA.

Druhý klaster najčastejšie tvorili štáty Čína, Francúzsko a Nemecko.

Tretí klaster tvoria štáty ako Taliansko, Južná Kórea, Holandsko, Rusko, Singapur a Veľká Británia.

Štvrtý klaster tvoria štáty ako napríklad India, Indonézia, Irán, Malajzia, Thajsko a aj Turecko.

Piaty klaster je tvorený štátmi Česká Republika, Rakúsko, Dánsko, Fínsko, ale boli tu zaradené aj štáty Argentína, Čile, Brazília a Kolumbia. Sú tu teda zahrnuté dve skupiny štátov geograficky od seba veľmi vzdialených, ktorých vzájomný obchod určité nie je nijak významný. Vo vnútri oboch skupín sa zrejme nájdu určité spoločné črty.

Šiesty klaster tvoria štáty Slovenská Republika, Poľsko, Maďarsko, Ukrajina a štáty Španielsko, Portugalsko a Nigéria. Môžeme pozorovať podobne ako v predchádzajúcom klastri spojenie dvoch skupín štátov.

Siedmy klaster je tvorený zvyšnými štátmi sveta.

Pri použití Kohonenových máp s rastúcou mriežkou sme objavili oproti klasickým Kohonenovým mapám viacero nových klastrov. Najčastejšie sa samozrejme objavujú klastre, ktoré sme spomenuli už pri Kohonenových mapách. Uveďme si niektoré nové klastre, ktoré doteraz ešte neboli objavené.

Jedným z nových klastrov je skupina štátov Kazachstan, Turkmenistan, Uzbekistan a štáty Lotyšsko a Litva.

Ďalší klaster tvoria štáty Chorvátsko, Grécko a Bulharsko, alebo napríklad štáty Alžírsko, Egypt, Izrael, Líbya a Macedónsko.

Pri použití modelu rastúcich neurónových plynov, boli v každom pokuse štáty rozdelené do dvoch klastrov. Prvý klaster obsahuje vždy niektoré zo silných štátov a druhý klaster tvoria zvyšné štáty sveta. Pomocou tohto modelu sme teda nezískali významnejšie výsledky.

Zhrnutie výsledkov

Po aplikácií uvedených klastrovacích metód môžeme získané poznatky zhrnúť nasledovne. Výsledné rozdelenia štátov do jednotlivých klastrov pomocou uvedených metód sa v podstate líšia len minimálne. Nakoniec rôzne výsledky na rovnakých dátach by predznamenávali nesprávnu funkčnosť spomenutých klastrovacích metód. Celkovo sa javí ako najvýhodnejšie použitie Kohonenových máp s rastúcou mriežkou. Rozdelenie štátov do klastrov pomocou tohto modelu bolo často podobné ako rozdelenie získané pomocou iných uvedených metód, avšak tento model Kohonenových máp odhalil najväčší počet rôznych rozdelení štátov do klastrov.

Čo sa týka rozdelenia štátov do jednotlivých klastrov, tak výsledky môžeme hodnotiť nasledovne. Prvý klaster bol najčastejšie tvorený štátmi USA, Čína, Japonsko a Nemecko. Obchodné vzťahy týchto štyroch štátov majú veľký podiel na celosvetovom objeme obchodov. Tieto štáty majú množstvo obchodných partnerov, sú nie len významný exportéri ale aj veľa dovážajú.

Druhý klaster môžu tvoriť štáty Taliansko, Francúzsko, India, Holandsko, Rusko, Singapur a Veľká Británia. Tieto štáty obchodujú ešte stále s veľkým počtom štátov celého sveta. Oproti štátom z prvého klastra je objem zrealizovaných obchodov znateľne nižší.

Tretí klaster môžu tvoriť štáty Južná Kórea, Malajzia, Saudská Arábia, Singapur, Thajsko. Jedná sa o štáty, ktoré sú významnými vývozcami do celého sveta.

U zvyšných objavených klastroch bolo možné pozorovať prvky regionálneho charakteru. To znamená, že štáty obsiahnuté v jednotlivých klastroch patria do rovnakých geografických oblastí ako napríklad južná Amerika alebo severná Afrika.

8. Program Klastrovacie Techniky

Súčasťou tejto diplomovej práce je aj program Klastrovacie Techniky. Jedná sa o jednoduchý program, pomocou ktorého môžeme testovať niektoré z uvedených klastrovacích metód, a to konkrétne:

- K-means algoritmus
- Fuzzy C-means algoritmus
- Kohonenové mapy
- Kohonenové mapy s rastúcou mriežkou
- Model rastúcich neurónových plynov

Všetky tieto uvedené metódy sú samostatne naprogramované, a teda svojím spôsobom jedinečné.

Program Klastrovacie Techniky bol vytvorený v programovacom jazyku C# a funguje na operačnom systéme Windows. K jeho úspešnému spusteniu je potrebné mať nainštalovaný Microsoft.NET Framework 4.

Diskové požiadavky tohto programu sú v dnešnej dobe zanedbateľné, keďže veľkosť tohto programu nepresahuje ani 1MB. Nie sú stanovené minimálne hardwarové požiadavky pre tento program, ale treba upozorniť, že klastrovacie algoritmy patria medzi výpočtovo náročnejšie algoritmy. Z tohto dôvodu môžu niektoré testy trvať dlhší čas.

Program Klastrovacie Techniky je dostupný na priloženom CD v priečinku Program a je možné spustiť ho pomocou Klastrovacie Techniky.exe.

8.1. Užívateľské rozhranie

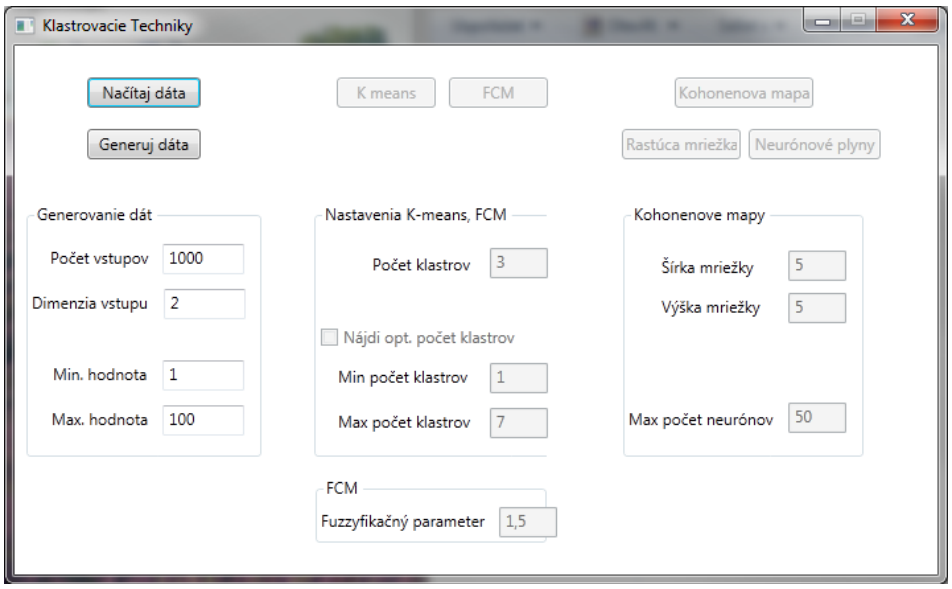
Po spustení programu sa zobrazí hlavné okno programu. Aby sme mohli testovať jednotlivé metódy, je potreba načítať dáta, s ktorými budeme pracovať. Máme dve možnosti.

1. Po kliknutí na tlačítko „Načítaj dáta“ sa zobrazí klasické dialógové okno pre výber súboru, s ktorým chceme pracovať. Vstupné dáta budeme načítavať z klasického textového súboru, v ktorom sa bude používať nasledovné formátovanie. Každý vstupný vzor sa nachádza na samostatnom riadku textového súboru, pričom prvá položka uvedená na riadku bude reprezentovať

názov vstupného vzoru a všetky ďalšie položky z tohto riadku odpovedajú jednotlivým zložkám vstupného vzoru. Prázdne riadky sú ignorované, takže ich výskyt v textovom súbore nespôsobuje žiadny problém. Uveďme si jednoduchý príklad takéhoto textového súboru obsahujúceho 3 vstupné vzory dimenzie napríklad 4.

A 0.1 0.2 0.3 0.4
B 1.1 1.2 1.3 1.4
C 1.2 0.3 0.8 1.5

2. Druhou možnosťou je pracovať s náhodne vygenerovanými dátami. K bližšej špecifikácií náhodných dát slúžia pomocné textové polia uvedené pod tlačítkom „Generuj dáta“, pomocou ktorých môžeme meniť jednotlivé nastavenia. Konkrétne môžeme nastaviť počet vstupných vzorov, ktoré sa budú generovať. Vstupným vzorom môžeme nastaviť dimenziu, teda počet zložiek, z ktorých sa budú skladať. Zároveň môžeme nastaviť interval hodnôt, z ktorého sa budú generovať hodnoty jednotlivých položiek vstupných vzorov. Po kliknutí na tlačítko „Generuj dáta“ sa vygeneruje množina vstupných vzorov, ktoré budú spĺňať špecifikované nastavenia. Pri generovaní vstupných vzorov je názov jednotlivých vstupných vzorov tvorený poradovým číslom.



Obrázok č. 27: Hlavné okno programu Klastrovacie Techniky.

Po načítaní, prípadne vygenerovaní vstupných dát sa sprístupnia tlačítka umožňujúce spustenie jednotlivých klastrovacích metód. Pred samotným spustením niektorej z klastrovacích metód je možne špecifikovať hodnoty niektorých parametrov.

Algoritmy K-means a FCM potrebujú mať určený požadovaný počet klastrov. Tento počet môžeme meniť v textovom políčku „Počet klastrov“. Ak chceme určiť optimálny počet klastrov pre zadané vstupné dáta, zaškrtneme políčko s textom „Nájdi opt. počet klastrov“. Následne sa sprístupnia dve textové políčka „Min. počet klastrov“ a „Max. počet klastrov“ určujúce interval, z ktorého sa hľadá optimálny počet klastrov. Pre FCM algoritmus je možné nastaviť hodnotu fuzzyfikačného parametra v textovom políčku „Fuzzyfikačný parameter“.

V prípade, že chceme pracovať s Kohonenovými mapami, aj tu sú možné určité nastavenia. Pri klasických Kohonenových mapách môžeme nastaviť požadované rozmery mriežky v príslušných textových políčkach „Šírka mriežky“ a „Výška mriežky“.

Pri použití Kohonenovej mapy s rastúcou mriežkou je možné zadať do políčka „Max počet neurónov“ číselnú hodnotu, ktorej význam je nasledovný. Pokiaľ sa v mriežke nachádza menší počet neurónov ako je zadaná hodnota, môžu sa do mriežky pridávať nové neuróny. Keďže v tomto modeli Kohonenových máp sa pridáva viacero neurónov súčasne, výsledný počet neurónov v mriežke býva často väčší ako zadaná hodnota.

Pri použití modelu rastúcich neurónových plynov už má hodnota zadaná v textovom políčku „Max počet neurónov“ odlišný význam. Tento model Kohonenových máp pridáva neuróny do mriežky po jednom, preto hodnota „Max počet neurónov“ v tomto prípade určuje počet neurónov, ktoré budú po skončení procesu učenia v mriežke.

Po zadaní požadovaných nastavení klikneme na tlačítko, ktorého text odpovedá požadovanej klastrovacej metóde, čím spustíme zvolenú klastrovaciu metódu. Po skončení procesu klastrovania sa nám zobrazia očakávané výsledky.

Podľa typu vstupných vzorov môžeme získať 2 druhy zobrazenia výsledkov klastrovania. Uvažujme, že pracujeme so vstupnými vzormi, ktorých dimenzia je dva. V takomto prípade môžeme prvú zložku vstupného vzoru považovať za x-ovú súradnicu a druhú zložku za y-ovú súradnicu. Nie je teda problém zobraziť rozdelenie vstupných dát do klastrov aj graficky. Príkladov takéhoto zobrazenia nájdeme v tejto práci viacero, preto nebudeme pridávať ďalší. Popri grafickom zobrazení sa otvorí nové okno s textovým popisom výsledného rozdelenia vstupných dát do klastrov. Z tohto textového popisu vieme jednoznačne určiť pre každý vstupný vzor, do ktorého klastra bol zaradený.

V prípade vstupných vzorov väčšej dimenzie ako dva sa grafické zobrazenie výsledkov neprevádza. Zobrazí sa teda len textový popis výsledkov klastrovania, z ktorého je však možné určiť všetky potrebné informácie. Uveďme si ukážku takéhoto textového výstupu.

Obrázok č. 28: Textové zobrazenie rozdelenia štátov sveta do klastrov pomocou Kohonenovej mapy. Vstupné dáta popisujú vzájomný obchod štátov sveta (7.2).

8.2. Implementácia programu Klastrovacie Techniky

Grafické rozhranie tohto programu bolo vytvorené pomocou nového grafického framework-u Windows Presentation Foundation. Jedná sa o alternatívny spôsob písania Windows aplikácií k známym Windows Forms. Pri grafickom zobrazovaní výsledkov sa využíva knižnica Microsoft.Research.DynamicDataDisplay. Program ako celok je vytvorený v programovacom jazyku C#.

Samotný program je rozdelený do viacerých tried.

Trieda vstupneHodnoty.cs je abstraktná trieda, ktorá udržiava aktuálne hodnoty zadaných parametrov v hlavnom okne programu.

Štruktúra minMax.cs slúži k uchovávaniu minimálnych a maximálnych hodnôt zložiek vstupných vzorov. Tieto hodnoty sú následne využité pri normalizácií vstupných vzorov.

Trieda K_Means.cs implementuje metódy potrebné k prevedeniu K-means klastrovania. Ide hlavne o metódy, ktoré majú na starosti aktualizáciu pozícií stredov jednotlivých zhlukov a ich vzájomné porovnávanie.

Trieda FCM.cs implementuje metódy potrebné k prevedeniu klastrovania pomocou FCM algoritmu. Najdôležitejšie metódy zabezpečujú aktualizáciu prvkov členskej matice a aktualizáciu pozícií stredov jednotlivých klastrov.

Trieda SOM.cs implementuje základné metódy umožňujúce pracovať s neurónmi v mriežke a stará sa o vznik mriežky samotnej. Táto trieda je dedená všetkými triedami, ktoré implementujú modely Kohonenových máp.

Trieda KohonenovaMapa.cs dedí triedu SOM.cs a implementuje proces učenia Kohonenovej mapy. Obsahuje metódy potrebné k aktualizácií váhových vektorov víťazného neurónu a neurónov z priameho okolia.

Trieda KM_RastucaMriezka.cs dedí triedu SOM.cs a implementuje proces učenia Kohonenovej mapy s rastúcou mriežkou. Nájdeme tu metódy umožňujúce vkladanie nových neurónov do mriežky

Trieda KM_NeuronovyPlyn.cs dedí triedu SOM.cs a implementuje proces učenia rastúcich neurónových plynov. Zahŕňa napríklad metódy na určovanie a správu hodnôt lokálnych chýb jednotlivých neurónov. Ďalej metódy pre správu hrán (zvyšovanie veku hrán, odstraňovanie hran).

Trieda MainWindow.xaml.cs zabezpečuje obsluhu hlavného okna programu. Jedná sa o centrálnu triedu, ktorá inicializuje vznik všetkých uvedených tried. Informácie o iniciálnych hodnotách a nastaveniach jednotlivých položiek hlavného okna je možné nájsť v MainWindow.xaml.

9. Záver

Cieľom tejto diplomovej práce bolo zrekapitulovať metódy vhodné pre klastrovanie vstupných dát a ich vzájomné porovnanie. Na začiatku bol uvedený stručný úvod do problematiky klastrovania. Uviedli sme základný princíp ako určovať podobnosť ľubovoľných objektov, ktorý nám umožní realizovať jednotlivé klastrovacie metódy.

Ako prvým sme sa zaoberali K-means algoritmom. Po predstavení K-means algoritmu sme sa rozoberali problém uviaznutia tohto algoritmu v lokálnom optime. Uviedli sme postup ako tento problém minimalizovať. Ďalej sme sa zaoberali metódami pre odhad optimálneho počtu klastrov. Ukázali sme si viacero typov vstupných dát, s ktorými má K-means algoritmus problémy.

Po prestavení FCM algoritmu sme sa zaoberali vplyvom fuzzyfikačného parametra na výsledné rozdelenie vstupných vzorov do klastrov. Uviedli sme viacero validačných kritérií, ktoré umožňujú ohodnotiť výsledky klastrovania, vďaka čomu dokážeme určiť optimálny počet klastrov.

Ďalej sme zaoberali problematikou Kohonenových máp. Po zoznámení sa s topológiou Kohonenových máp sme rozobrali význam okolia neurónov v topologickej mriežke. Uviedli sme dve rôzne funkcie laterálnej interakcie. Následne sme uviedli dva modely Kohonenových máp, ktoré umožňujú dynamicky meniť počet neurónov v topologickej mriežke. Uviedli sme problém nevyužitých neurónov, ktorý sa týka Kohonenových máp s rastúcou mriežkou. Pri modeli rastúcich neurónových plynov sme spomenuli problém tykajúci sa rôznej dimenzionality v rôznych oblastiach vstupného priestoru.

Predstavili sme si nový model neurónových sieti, a to radiálne bázickú neurónovú sieť. Vysvetlili sme význam skrytých RBF jednotiek. Uviedli sme viacero učiacich algoritmov, vrátane algoritmu GGAP (A Generalized Growing and Pruning), ktorý umožňuje dynamicky meniť počet skrytých RBF jednotiek. Podrobne sme rozobrali algoritmus kontextového klastrovania, ktorý pomáha riešiť problém veľkej dimenzionality vstupných dát. Ďalej sme spomenuli citlivostnú analýzu, ktorá skúma citlivosť celkového výstupu na zmenu jednotlivých parametrov v sieti. RBF siete patria medzi najmladšie neurónové siete a stále sa u nich vyvíjajú nové techniky. Zrejme aj z tohto dôvodu nie je problém nájsť v literatúre nejednoznačnosť v použitom značení.

V ďalšej časti tejto práce boli prevedené experimenty s niektorými z uvedených klastrovacích metód. Pri týchto experimentoch sme uviedli vplyv zmien niektorých parametrov na výsledok klastrovania. Najskôr sme experimentovali s umelo vygenerovanými dátami a následne sme otestovali spomenuté klastrovacie metódy na reálnych dátach s oblastí medzinárodného obchodu.

Záver práce bol venovaný predstaveniu programu Klastrovacie Techniky, ktorý je súčasťou tejto diplomovej práce. Tento program implementuje niektoré zo spomenutých klastrovacích metód.

10. Použitá literatúra

[1] Antonini M., Barlaud M., Mathieu P.: In Signal Processing V. theories and Applications. Proc EUSIPCO-90, Fifth European Signal Processing Conference, ed. by L. Torres, E. Masgrau, M. A. Lagunas (Elsevier, Amsterdam, Netherlands 1990), p. II-1091.

[2] Backer E., Jain A.K.: A clustering performance measure based on fuzzy set decomposition, IEEE Trans. Patten Anal. Mach. Intell. 3 (1) (1981) 66–74.

[3] Bezdek J. C., Ehrlich R., Full W.: FCM: The fuzzy c-means clustering algorithm, in: Computers & Geosciences Vol.10, No. 2-3, pp. 191-203, 1984.

[4] Bezdek J. C.: Cluster validity with fuzzy sets, J. Cybernet. 3 (1974) 58–73.

[5] Bezdek J. C.: Numerical taxonomy with fuzzy sets, J. Math. Biol. 1 (1974) 57–71.

[6] Bezdek J. C.: Pattern Recognition with Fuzzy Objective Function Algorithms, Plenum Press, NewYork, 1981.

[7] Broomhead D. S., Lowe D.: Multivariable functional interpolation and adaptive networks. Cmplex Systems, 2:321-355, 1988.

[8] Cover T. M.: Geometrical and statistical properties of systems of linear inequalities with application in pattern recognition. IEEE Transaction on Electronic Computers, EC-14:326-334, 1965.

[9] Fritzke B.: A Growing Neural Gas Network Learns Topologies, Advances in Neural Information Processing Systems 7, pp. 625-632, MIT PRESS, 1995.

[10] Fritzke B.: Growing Grid - a self-organizing network with constant neighborhood range and adaptation strength, Neural Processing Letters, Vol. 2, pp. 9-13, 1995.

[11] Haykin S.: Neural networks: a comprehensive foundation, Prentice Hall, 1999.

[12] Hong X. and Billings S.: Givens rotation based fast backward elimination algorithm for RBF neural network pruning, *Proc. Inst. Elect. Eng. Control Theory Appl.* 144 (5) (1997), pp. 381–384.

[13] Hong X., Harris C., Brown M. and Chen S.: Backward elimination methods for associative memory network pruning, *Int. J. Hybrid Intell. Syst.* 1 (1) (2004), pp. 90–99.

[14] Huang G.-B., Saratchandran P., Sundararajan N.: A Generalized Growing and Pruning RBF(GGAP-RBF) Neural Network for Function Approximation, in: IEEE Transactions on Neural Networks, Vol. 16, No. 1, (2005) pp. 57-67.

[15] Huang S. H.: Dimensionality reduction in automatic knowledge acquisition: A simple greedy search approach, in: IEEE Trans. Knowl. Data Eng., Vol. 15, No. 6, (2003) pp. 1364-1373.

[16] Chen S., Cowan C. F., Grant P. M.: Orthogonal Least Squares Learning Algorithm for Radial Basis Function Network, IEEE Transactions on Neural Networks, vol. 2, no. 2, March 1991.

[17] Jakubowitz O. G.: In Proc. IJCNN'89, Int. Joint Conf. on Neural Networks (IEEE Service Center, Piscataway, NJ, 1989) p. II-23.
[18] Jiao J., Zhang Y., and Wang Y.: "A heuristic genetic algorithm for product portfolio planning," *Comput. Operat. Res.*, vol. 34, pp. 1777–1799, 2007.
[19] Kallio K., Haltsonen S., Paajanen E., Rosqvist T., Katila T., Karp P., Malmberg P., Piirilä P., Sovijärvi A. R. A.: In Artificial Neural Networks, ed. By T. Kohonen, K. Mäkisara, O. Simula, J. Kangas (North-Holland, Amsterdam, Netherland 1991) p. I-803.
[20] Kim N. J., Kehtarnavaz N., Yeary M. B. and Thornton S.: "DSP-based hierarchical neural network modulation signal classification," *IEEE Trans. Neural Netw.*, vol. 14, no. 5, pp. 1065–1071, Sep. 2003.
[21] Koh J., Suk M., Bhandarkar S. M.: In Proc. ICNN'93, Int. Conf. on Neural Networks (IEEE Service Center, Piscataway, NJ, 1993) p. III-1270.
[22] Kohonen T.: Self-Organizing Maps, 3rd Edition, Springer-Verlag, 2001.
[23] Kohonen T.: Self-Organizing Maps. In Proc. of IEEE volume 78. September 1990.
[24] Liu X., Cheng G., Wu J.: In Proc. ICNN'94, Int. Conf. on Neural Networks (IEEE Service Center, Piscataway, NJ, 1994) p. 649.
[25] Luttrell S. P.: Pattern recognition Letters 10, 1 (1989).
[26] Mao K. Z., Huang G.-B.: Neuron selection for RBF neural network classifier based on data structure preserving criterion, in: IEEE Trans. Neural Netw., Vol. 16, No. 6, (2005) pp. 1531-1540.
[27] Martinetz T., Schulten K.: In Proc. ICNN'93, Int. Conf. on Neural Networks (IEEE Service Center, Piscataway, NJ, 1993) p. II-820.
[28] Moody J. and Darken C.: Learning with localized receptive fields. In D. Touretzky, G. Hinton, and T. Sejnowski, editors, Proceedings of the Connectionist Models Summer School. San Mateo: Morgan Kaufmann, 1989, 1990.
[29] Moody J. and Darken C.: Fast adaptive k-means clustering: some empirical results. In Proc. IJCNN, San Diego '90, volume 2, pages 233-238, 1990.
[30] Moody J., Darken C.: Learning with localized receptive fields. In D. Touretzky, G.Hinton and T. Sejnowski, editors, Proceedings of the Connectionist Models Summer School. San Mateo: Morgan Kaufmann, 1988, 1989.
[31] Morabito M., Macerata A., Taddei A., Marchesi C.: In Proc. Computers in Cardiology (IEEE Comput. Soc. Press, Los Alamitos, CA 1991) p. 181.
[32] Oh S. K., Pedrycz W., and Park H. S.: "Genetically optimized fuzzy polynomial neural networks," *IEEE Trans. Fuzzy Syst.*, vol. 14, no. 1, pp. 125–144, Feb. 2006.
[33] Orr M. J. L. and Hallam J.: Int. J. Neural Syst. 10 5 (2000), pp. 397–415.
[34] Orr M. J. L.: Introduction to radial basis function networks, 1996.

[35] Park H.-S., Pedrycz W., Oh S.-K.: Granular Neural Networks and Their Development Through Context-Based Clustering and Adjustable Dimensionality of Receptive Fields, in: IEEE Transactions on Neural Networks, (2009) 13 p., DOI: 10.1109/TNN.2009.2027319.
[36] Pedrycz W.: Fuzzy clustering with a knowledge-based guidance, in: Pattern Recognition Letters, Vol. 25, (2004) pp. 469-480.
[37] Powell M. J. D.: Radial basis function for multivariable interpolation: A review. In J. C. Mason and M. G. Cox, editors, Algorithms for approximation, pages 143-167. Oxford Science Publications, 1987.
[38] Rojas I., Pomares H., Bernier J. L., Ortega J., Pino B., Pelayo F. J. and Prieto A.: "Time series analysis using normalized PG-RBF network with regression weights," *Neurocomput.*, vol. 42, pp. 267–285, 2002.
[39] Rojas R.: Neural Networks: A Systematic Introduction, Springer-Verlag, 1996.
[40] Salmerón M., Ortega J., Puntonet C. G. and Prieto A.: "Improved RAN sequential prediction using orthogonal techniques," *Neurocomput.*, vol. 41, pp. 153–172, 2001.
[41] Shi D., Yeung D. S., Gao J.: Sensitivity analysis applied to the construction of radial basis function networks, in: Neural Networks, Vol. 18, (2005) pp. 951-957.
[42] Šíma J., Neruda R.: Teoretické otázky neuronových sítí, MATFYZPRESS, Praha, 1996.
[43] Trauwaert E.: On the meaning of Dunn's partition coefficient for fuzzy clusters, Fuzzy Sets and Systems 25 (1988) 217–242.
[44] Verma B. and Zhang P.: "A novel neural-genetic algorithm to find the most significant combination of features in digital mammograms," *Appl. Soft Comput.*, vol. 7, pp. 612–625, 2007.
[45] Wang W., Zhang Y.: On fuzzy cluster validity indices, in Fuzzy Sets and Systems 158 (2007) 2095-2117.
[46] Windham M. P.: Cluster validity for fuzzy clustering algorithms. Fuzzy Sets and Systems 5 (1981) 177–185.
[47] Yeung D. S., Ng W. W. Y., Wang D., Tsang E. C. C., Wang X. Z.: Localized generalization error model and its application to architecture selection for radial basis function neural network, in: IEEE Transactions on Neural Networks, Vol. 18, No. 5, (2007) pp. 1294-1305.

[48] International Monetary Fund: http://www.imf.org/external/index.htm
[49] http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletFCM.html
[50] Wikipedia – Determining the number of cluster in a data set: http://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set#The_Elbow_Method

Dodatok

Obsah priloženého CD

- súbor Diplomová práca.pdf – text diplomovej práce vo formáte pdf

- priečinok Program obsahuje:
  - Microsoft .Net Framework 4
  - vstupné dáta popisujúce medzinárodný obchod
  - obsahuje spustiteľnú verziu programu KlastrovacieTechniky spolu s potrebnými DLL knižnicami

- priečinok Zdrojové súbory, kde je uložený celý projekt