Klasifikace různých zdrojů zvuků pomocí metod umělé inteligence v průmyslu komerční bezpečnosti

Classification of different sound sources with useage of artificial intelligence in commercial security industry

Bc. Michal Maťuš

Diplomová práce
2010

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
akademický rok: 2009/2010

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Michal MAŤUŠ**
Studijní program:	**N 3902 Inženýrská informatika**
Studijní obor:	**Bezpečnostní technologie, systémy a management**
Téma práce:	**Klasifikace různých zdrojů zvuků pomocí metod umělé inteligence v průmyslu komerční bezpečnosti**

Zásady pro vypracování:

1. **Prostudujte doporučenou literaturu a zpracujte literární rešerši na dané téma.**
2. **Zaměřte se na problematiku neuronových sítí.**
3. **Navrhněte a vytvořte databázi zvuků, která bude představovat trénovací množinu.**
4. **Z uvedené databáze extrahujte vhodné charakteristiky (spektrum apod.) vedoucí k úspěšné klasifikaci předkládaných zvuků.**
5. **V programovacím prostředí MS Visual studia vytvořte windows-form aplikaci, obsahující algoritmus realizující vícevrstvou neuronovou síť s učícím algoritmem backpropagation a další algoritmy nezbytné pro vyhodnocení.**
6. **Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování.**

Rozsah práce:

Rozsah příloh:

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

1. **Sibbald, A.: An introduction to sound and hearing [online]. 1999 [cit. 2007-02-10]. Dostupný z WWW: http://www.sensaura.com/whitepapers/index.php.**
2. **Kolmer, F., Kyncl, J.: Prostorová akustika. Praha : SNTL - Nakladatelství technické literatury, 1980. 244 s.**
3. **Frigo, M., Johnson, S. G. FFTW3, [online]. 2006 [cit. 2007-02-15]. Dostupný z WWW: http://www.fftw.org/documentation.**
4. **Zelinka Ivan. Umělá inteligence I. Volume 1. Zlín: Vutium, Brno, 1998. 126 p. ISBN 80-214-1163-5.**
5. **Kvasnička V., Beňušková, Pospíchal J., Farkaš I., Tiňo P. a Kráľ A. (1997). Úvod do teórie neurónových sietí Iris: Bratislava.**
6. **Ben Kröse and Patrick van der Smagt (1996), An introduction to Neural Networks.**
7. **Sinčák , Peter, ANDREJKOVÁ, Gabriela. Neurónové siete Inžiniersky prístup [online]. Dostupný z WWW: http://hq.alert.sk/ámandos/fmfi-uk/Informatika/Neuronove%20Siete/NS1/node1.html.**
8. **Neural network - Wikipedia [online]. Dostupný z WWW: http://en.wikipedia.org/wiki/Neural_network.**
9. **Beňušková L. Umele neurónové siete. In: Návrat P., Bieliková M., Benuskova L, Kapustik I, Unger M. Umela inteligencia.**

Vedoucí diplomové práce: **Ing. Milan Navrátil, Ph.D.**
Ústav elektroniky a měření

Datum zadání diplomové práce: **19. února 2010**

Termín odevzdání diplomové práce: **7. června 2010**

Ve Zlíně dne 19. února 2010

L.S.

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. RNDr. Vojtěch Křesálek, CSc.
ředitel ústavu

ABSTRAKT

Cieľom tejto diplomovej práce je spracovať rešerši o problematike rozpoznávania zvukov pomocou neurónových sietí. Naprogramovať windows aplikáciu pomocou, ktorej bude možné klasifikovať vybraný zvuk do správnej kategórie. Ako jadro klasifikačnej aplikácie bude použitá viacvrstvová neurónová sieť, ktorá bude využívať učiaci algoritmom backpropagation.

Kľúčové slová: neurónová sieť, backpropagation, klasifikácia, rozpoznávanie zvuku

ABSTRACT

The aim of this thesis is to make research on the issue of sound recognition using neural networks. To program a windows-form application by which it will be possible to classify the selected sound to correct category. As the core of this classification application will be used a multilayer neural network which will use the backpropagation learning algorithm.

Keywords: neural network, backpropagation, classification, sound recognition

Poďakovanie:

V prvom rade by som chcel poďakovať svojmu vedúcemu diplomovej práce Ing. Milanovi Navrátilovi, Ph.D za cenné rady a postrehy, ktoré mi veľmi pomohli pri spracovávaní zadanej problematiky. Ďalej sa chcem poďakovať svojej mamine za to, že mi päť rokov sponzorovala moje štúdium a v neposlednej rade tiež svojej priateľke, ktorá pri mne stála a po celú dobu písania tejto práce mi bola psychickou oporou.

Motto:

Na to, aby zlo zvíťazilo, stačí málo: aby dobrí ľudia nerobili nič.

Edmund Burke

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové/bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová/bakalářská práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové/bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně a jeden výtisk bude uložen u vedoucího práce;
- byl/a jsem seznámen/a s tím, že na moji diplomovou/bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou/bakalářskou práci nebo poskytnout licenci k jejímu využití jen s předchozím písemným souhlasem Univerzity Tomáše Bati ve Zlíně, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše);
- beru na vědomí, že pokud bylo k vypracování diplomové/bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové/bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové/bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně …….

podpis diplomanta

OBSAH

ÚVOD

V moderných systémoch sa pre riešenie komplexných problémov často využívajú neurónové siete. Na rozdiel od klasických algoritmov, ktoré sa hodia na riešenie množstva bežných úloh, neurónovú sieť môžeme použiť pri riešení problémov, ktoré nie sú ľahko definovateľné. Štandardné algoritmy nedokážu dobre pracovať s porušenými alebo nekompletnými dátami, ktoré sú v reálnom svete väčšinou jediný dostupný druh dát.

Neurónová sieť je výpočtový model, ktorý je schopný sám sa učiť a nájsť riešenie na zadaný problém. Často je používaný pre riešenie problémov v umelej inteligencii. Neurónové siete sú vytvorené na základe odpovedajúcich biologických štruktúr v ľudskom tele. Sieť sa skladá z umelých neurónov, podobne ako ľudský mozog. Predobrazom umelých neurónov je biologicky neurón, ktorý je ale veľmi zjednodušený.

Hlavným cieľom tejto diplomovej práce je navrhnúť a naprogramovať umelú neurónovú sieť, ktorá by bola schopná naučiť sa klasifikovať zdroje zvukových signálov.

V teoretickej časti sa zameriam na problematiku klasifikácie a strojového vnímania v širšom slova zmysle. Popíšem jednotlivé metódy, ktoré sú často využívané pre klasifikáciu pomocou počítaču. Vypracujem rešerši na zadané téma s tým, že sa zameriam na problematiku neurónových sieti. Vysvetlím princíp funkcie neurónovej siete a popíšem typy najviac používaných neurónových sieti.

V praktickej časti diplomovej práce navrhnem a vytvorím databázu zvukových signálov, ktoré budú predstavovať učiacu množinu vstupných dát neurónovej siete. Z tejto množiny získam charakteristiky, pomocou ktorých bude možné jednotlivé zvuky odlíšiť od ostatných. Vytvorím viacvrstvovú neurónovú sieť využívajúcu algoritmus backpropagation ako spôsob učenia. Túto sieť naprogramujem v programovacom jazyku C#. Pre učenie neurónovej siete a zobrazenie výsledkov využijem ovládacích prvkov klasickej windows-form aplikácie.

Pre tematiku neurónových sietí som sa rozhodol z dôvodu modernosti riešenia problémov týmto spôsobom a širokej škále ich využitia. Neurónové siete majú veľa možností použitia ako v bezpečnostnom priemysle (rozpoznávanie obrazu, biometrické access systémy) tak aj v ekonomike (predpoveď vývoju kurzov na burze, rozpoznávanie písma a podpisov) alebo zdravotníctve (analýza snímkov, prognóza liečenia v onkológii).

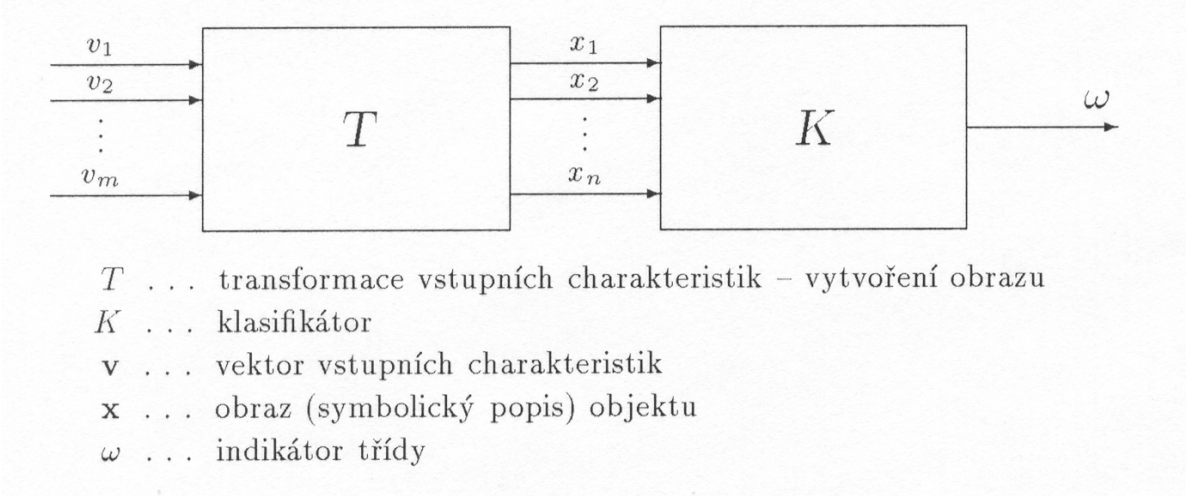
I. TEORETICKÁ ČASŤ

1 PRINCÍPY KLASIFIKÁCIE ZVUKOV

Rozpoznávanie chápeme ako úlohu, pri ktorej objekty triedime podľa ich spoločných vlastností tak, že objekty, ktoré sú si vzájomne podobné zaraďujeme do rovnakej triedy.

Rozlišujeme:

- **klasifikáciu** – zaraďujeme objekty do vopred známeho, pevného počtu tried
- **rozpoznávanie** – počet tried nie je vopred známy a triedy identifikujeme až behom rozpoznávania podľa spoločných vlastností jednotlivých objektov



Obr. 1 – Obecná klasifikačná úloha

Rozhodovacie pravidlo, podľa ktorého klasifikátor priraďuje objekt do klasifikačnej triedy, môžeme obecne definovať ako skalárnu funkciu vektorového argumentu $\omega = d(x)$.

Presnejšie vyjadrenie rozhodovacieho pravidla, ktoré zohľadňuje aj tzv. nastavenie klasifikátor q, je $\omega = d(x, q)$. Nastavení klasifikátoru sa vykonáva tzv. trénovaním alebo učením.

Rozlišujeme:

- **učenie s učiteľom**, keď klasifikátoru predkladáme objekty, u ktorých poznáme ich príslušnosť k triede
- **učenie bez učiteľa**, keď správne zaradenie do klasifikačných tried nepoznáme.

1.1 Rozpoznávanie zvukov človekom

Zvuk je kmitavý pohyb molekúl vzduchu, ktorý je vyvolávaný pružným odporom prostredia. Kmitom hmotného bodu rozumieme pohyb bodu z rovnovážnej polohy do miesta najväčšej výchylky – amplitúdy, odtiaľ do druhej amplitúdy a späť do rovnovážnej polohy.

Perióda tohto kmitavého pohybu sa nazýva frekvencia. Fyzikálnou jednotkou frekvencie je 1 hertz. Keď kmitá hmotný bod s frekvenciou 100 Hz, znamená to, že vykoná 100 kmitov za sekundu.

Akustická intenzita I je fyzikálna veličina, vyjadrujúca množstvo akustickej energie (energie kmitajúcich častíc prostredia), ktoré prejde jednotkovou plochou za jednotku času. Akustický tlak odpovedá sile pôsobiacej na element plochy v prostredí vlnivého deju. Akustická intenzita je úmerná štvorci akustického tlaku.

Rozsah intervalu minimálnej (I_0) a maximálnej (I_1) akustickej intenzity, v ktorom môžeme vnímať referenčný tón 1000 Hz je daný pomerom štvorcu akustického tlaku nazývaného prah citlivosti. Pre stanovenie hladiny intenzity zvuku volíme preto logaritmus pomeru intenzít $\boldsymbol{log(I / I_0)}$. Táto hodnota sa nazýva hladina akustickej intenzity a jednotka odvodená z tejto hodnoty sa nazýva 1 bel (1 B).

Orientační hodnoty:

- šepot: 10 - 20 dB
- tlmená reč: 35 - 45 dB
- reč strednej hlasitosti: 50 - 55 dB
- symfonický orchester: 70 - 90 dB
- rocková hudba: 110 - 130 dB

Človek vníma zvuk pomocou vedenia zvukových vĺn cez sluchové orgány a pomocou sluchového nervu až do mozgu. Sluchom je človek schopný rozoznať zvuky a tóny, ich intenzitu, výšku, smer aj odkiaľ prichádzajú. Človek počuje a rozlišuje tóny od kmitočtu 16 Hz až 16000 – 20000 Hz. Maximálna citlivosť sluchu je pre tóny okolo 1000 až 3000 Hz.

Sluchový orgán sa skladá z vonkajšieho, stredného a vnútorného ucha. Vonkajšie ucho sa skladá z ušnice, zvukového kanáliku a bubienka. Stredné ucho sa skladá z bubienkovej dutiny, sluchových kostičiek a eustachovej trubice a vnútorné ucho sa skladá z oválneho okienka, slimáka a cortiho orgánu (nervových buniek, ktoré tvoria sluchové receptory).

Kmity zvukových vĺn prechádzajú zvukovým kanálikom a rozkmitajú bubienok. Sluchovými kostičkami sa prenesú na oválne okienko vo vnútornom uchu. Tieto kostičky zosilnia vibrácie až 30 krát. Kmity oválneho okienka sa prenášajú tekutinou horného priestoru slimáka a pôsobia na vláskové bunky (sluchové receptory), v ktorých dochádza k prevodu mechanickej energie zvukových vĺn na elektrický signál.

Pomocou tohto ústrojenstva dokáže človek rozpoznať výšku tónu aj hlasitosť zvuku. Krátke vlny (vysokého kmitočtu) dosiahnu najvyššieho rozkmitu blízko oválneho okienka. Čím dlhšia je vlna (čím nižšieho kmitočtu), tým bližšie k vrcholu slimáka je jej maximum. A tak výšku vnímaného tónu určuje to, ako ďaleko od oválneho okienka sú vláskové bunky najviac podráždené. Čím bližšie k nemu, tým vyšší tón je vnímaný, čím ďalej, tým je tón hlbší.

Hlasitosť zvuku je vnímaná tak, že slabý zvuk vyvoláva vzruch len v malom počte nervových vlákien. Pri zvyšujúcej sa hlasitosti zvuku vznikajú vzruchy aj v príslušných susediacich vláknach. Vlákna sluchového nervu prechádzajú mozgovou spodinou pod kôrovými centrami cez niekoľko synapsií. Tie predstavujú akési relé, v ktorom sa mení kód elektrických informácií (impulzov). Do mozgu tak prichádzajú len zhluky elektrických impulzov. Oblasti mozgovej kôry k tomu určené musia rozoznať tieto informácie a vytvoriť podľa nich zvukový obraz.

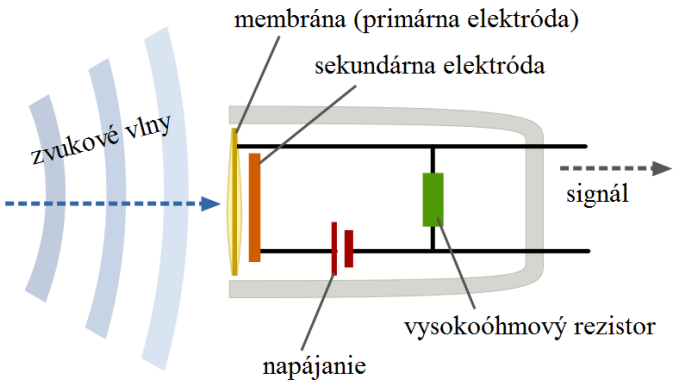
1.2 Rozpoznávanie zvukov počítačom

Tak ako pri vnímaní zvukových vĺn človekom je potrebné najskôr mechanické vlnenie vzduchu premeniť na elektrické impulzy, ktoré dokáže počítač spracovať. Pri riešení tohto problému môžeme počítač chápať ako ľudský mozog, ktorý dokáže len spracovávať a vyhodnotiť prijaté elektrické impulzy. K tomu potrebujeme ďalšie zariadenia, ktoré dokážu premeniť akustický (zvukový) signál na signál elektrický. Toto zariadenie sa volá mikrofón. Takto získaný signál je v analógovej podobe a je ho treba previesť na diskrétny (digitálny) signál zrozumiteľný číslicovým systémom.

Membrána **dynamického mikrofónu** sa pohybuje v závislosti na akustickom tlaku. Toto zároveň spôsobuje aj pohyb cievky, ktorá je spojená s membránou. V cievke pohybujúcej sa v magnetickom poli vzniká elektrický signál. Výstupné napätie sa mení v závislosti na pohybe membrány a cievky. U tohto mikrofónu pohybuje membrána cievkou v magnetickom poli permanentného magnetu, čím sa vytvára striedavé napätie respektíve prúd (ide o opačný proces ako sa odohráva v reproduktoroch).

Uhlíkový mikrofón je založený na stláčaní uhlíkových granulí. Ak sú tieto stlačené ich odpor klesá a výstupné napätie rastie.

Kondenzátorový mikrofón obsahuje dve tenké kovové doštičky, ktoré navzájom tvoria kondenzátor. Jedna z týchto doštičiek je pevne uchytená a druhá pohyblivá doštička má funkciu akustickej membrány. Vzájomné približovanie a odďaľovanie týchto doštičiek spôsobuje zmenu kapacity a tým aj zmenu výstupného napätia.



Obr. 2 – Kondenzátorový mikrofón

Piezoelektrický mikrofón je založený na piezoelektrickom jave. Kryštálová substancia pri svojej deformácii generuje polarizované elektrické napätie.

Mikrofónom zaznamenaný signál je potrebné previesť do digitálnej formy za pomoci analógovo digitálneho prevodníku. A/D prevodník je elektronická súčiastka určená pre prevod spojitého (analógového) signálu na signál diskrétny (digitálny). Dôvodom tohto prevodu je umožnenie spracovania analógového signálu na číslicových systémoch. V digitálnej podobe sa dajú signály oveľa kvalitnejšie zaznamenávať aj prenášať.

Prevod spojitého signálu na diskrétny sa skladá z dvoch fáz. Najskôr sa vykoná vzorkovanie signálu, a potom nasleduje kvantovanie signálu.

Vzorkovanie sa vykonáva tým spôsobom, že rozdelíme vodorovnú os signálu na rovnomerné úseky a z každého úseku odoberieme jednu vzorku. Je zrejmé, že tak z pôvodného signálu stratíme mnoho detailov, pretože namiesto spojitej čiary, ktorú je možné do nekonečna zväčšovať dostávame množinu diskrétnych bodov s intervalom odpovedajúcim použitej vzorkovacej frekvencii.

Pri vzorkovaní môže vzniknúť tzv. vzorkovacia chyba. Ak sa v pôvodnom spojitom signáli vyskytujú frekvencie vyššie ako je polovica vzorkovacej frekvencie (nazývaná tiež Nyquistová frekvencia), dôjde, ako hovorí Shannonov teorém, k úplnému a nenávratnému skresleniu signálu vďaka javu nazývanému aliasing. Aliasingu sa dá zabrániť len pomocou takzvaného antialiasing filtru, čo je dolná prepusť zaradená pred prevodníkom. Tá nedovolí frekvenciám vyšším ako je Nyquistova frekvencia vstúpiť do prevodníku.

Vzhľadom k tomu, že počítače a ďalšie zariadenia ďalej spracovávajúce digitálny signál dokážu vyjadriť čísla len s obmedzenou presnosťou, je treba navzorkované hodnoty upraviť i na zvislej ose. Pretože sa hodnota vzorku dá vyjadriť len po určitých kvantách, nazývame túto fázu A/D prevodu kvantovanie. Aby bolo možné určiť, ktoré hodnoty má po kvantovaní nadobudnúť určitá vzorka, je treba rozdeliť priestor okolo jednotlivých hodnôt na tolerančné. Ktorémukoľvek vzorku, ktorý padne do daného tolerančného pásu, je pri kvantovaní priradená daná hodnota .

Pretože sa digitálny signál spravidla spracováva na zariadeniach pracujúcich v dvojkovej číselnej sústave, bývajú počty kvantizačných úrovní A/D prevodníkov rovné *n*-tej mocnine čísla 2, pričom nakvantovaný signál je možné vyjadriť v *n* bitoch.

1.3 Metódy automatického rozpoznávania zvuku

Po prevedení akustického vlnenia na spojitý elektricky signál a transformovaní tohto signálu na digitálny je potrebné spracovať tieto dáta počítačom a roztriediť jednotlivé zvuky do správnych kategórií. Riešenie tohto problému je oveľa zložitejšie ako sa môže zo začiatku zdať. Je to preto, že variabilita zvukov v prírode je omnoho zložitejšia pre získanie odlišného obrazu zvuku, pretože sú zvuky v prírode rôznorodé. Najviac používané metódy rozpoznávania zvukov spadajú do týchto hlavných kategórii.

Vzorovo-založené metódy. V týchto metódach je neznáma vzorka zvuku porovnávaná voči množine predznačených vzorov, za účelom nájsť najlepšiu zhodu. Má to však nevýhodu, že predznačené vzory sú fixné, takže rôzne vzory rovnakého typu zvuku môžu byť modelované iba použitím mnohých vzorov pre daný typ, čo sa napokon stáva nepraktické a náročné na čas a výpočtový výkon. Problém je v porovnaní obrazu vzorového a neznámeho zvuku, pretože zvuky môžu byt odlišné v čase a rýchlosti. K porovnávaniu sa používa dynamické programovanie. Funkcia dynamického skreslenia času DTW (dynamic time warping), ktorá nelineárne upraví časovú osu tak, aby vzorky zvukov boli čo najviac podobne.

Znalostne-založené metódy. V týchto metódach sú znalosti experta o zmenách v jednotlivých zvukoch zakódované do systému. Je treba formulovať znalosti o vytváraní zvuku (báza znalostí) a údajov o konkrétnom zvuku (báza dát). Používajú sa princípy expertných systémov. Akustické znalosti vyjadrujú vzťah medzi jednotlivými druhmi zvukov. To je výhoda od explicitného modelovania zmien vo zvukoch, ale takéto znalosti je zložité dosiahnuť a úspešne použiť. Takže táto metóda bola posúdená ako nepraktická, ale aj napriek tomu boli hľadané automatické učiace procedúry.

Štatisticky-založené metódy. Zmeny vo zvukoch sú modelované štatisticky (napr. Skryté Markové modely - HMM), s použitím automatických učiacich procedúr. Táto metóda reprezentuje súčasne najmodernejší stav. Hlavná nevýhoda štatistických modelov je, že musia vytvoriť modelovacie predpoklady, ktoré sú môžu byť nepresné. Využívajú sa metódy vektorovej kvantizácie a diskrétnej Fourierovej transformácie.

Pre automatické rozpoznávanie zvukov v mojom programe som sa rozhodol použiť metódu využívajúcu diskrétnu Fourierovu transformáciu a následné zaradenie zvuku do kategórie pomocou neurónovej siete s dopredným šírením.

1.3.1 Dynamické skreslenie času (DTW)

Metód a DTW sa používa pre porovnávanie dvoch úsekov zvuku, vyjadrených postupnosťou akustických vektorov, vzniknutých rozdelením slov do mikrosegmentov a ich klasifikáciou súborom krátkodobých charakteristík. Akékoľvek dáta, ktoré sa môžu meniť do lineárnej reprezentácie môžu byt analyzované s DTW. Vo všeobecnosti ide o metódu, ktorá umožňuje počítaču nájsť optimálnu zhodu medzi dvoma sekvenciami (napr. časová rada) s určitými obmedzeniami, napr. sekvencie sú obaľované nelineárne, aby sa navzájom zhodovali.

Pre množinu rozpoznávaných slov vytvoríme súbor referenčných postupností akustických vektorov (obvykle pre každý zvuk niekoľko postupností odpovedajúcich niekoľkým rôznorodým zneniam toho zvuku). Metódou DTW porovnávame túto postupnosť s referenčnými, a za rozpoznané slovo vezmeme to, ktoré odpovedá najväčšej zhode.

1.3.2 Skryté Markové modely (HMM)

Z predstavy o vytváraní zvuku vychádza i konštrukcia klasifikátoru, založená na modelovaní akustického signálu pomocou Markového modelu. Pri tomto procese sú generované dve vzájomne viazané postupnosti náhodných premenných, a to podporný Markový reťazec, ktorý je postupnosťou konečného počtu stavov, a reťazec konečného počtu spektrálnych vzorov. Pre jednotlivé spektrálne vzory sú vytvorené náhodné funkcie, ktoré pravdepodobnostne ohodnocujú vzťah týchto vzorov ku všetkým stavom.

Predpokladá sa, že v diskrétnych časových okamihoch je proces v jedinom stave, a tak ho možno pozorovať prostredníctvom náhodnej funkcie korešpondujúcej s bežným stavom. Podporný Markov reťazec potom mení stavy podľa svojej matice pravdepodobností prechodov. Viditeľný je len výstup náhodných funkcií a nie je možné priamo pozorovať stavy podporného Markovho reťazca. Odtiaľ je odvodený termín skrytý Markov model (Hidden Markov Model - HMM).

Ľavo-pravý Markov model sa využíva pri modelovaní procesov, ktorých vývoj je spojený s postupujúcim časom. Základnou vlastnosťou týchto modelov je, že proces začína príchodom prvého spektrálneho vzoru z počiatočného stavu modelu a z narastajúcim časom dochádza k prechodu zo stavu s nižším indexom do stavu s vyšším indexom, alebo

zotrvá v pôvodnom stave. Prechod modelom je teda zľava doprava. Proces končí príchodom posledného spektrálneho vzoru, pričom model sa v tom okamihu nachádza v konečnom stave.

1.3.3 Rýchla Fourierova transformácia (FFT)

Z teórie analógových systémov vieme, že na analýzu periodických signálov sa zväčša používa Fourierova transformácia (Fourierove rady), ktorá vedie k diskrétnym (čiarovým) spektrám. Analýza neperiodických signálov pomocou Fourierovej transformácie vedie k spojitým spektrám. Avšak aj napriek tomu, že je Fourierova transformácia určená pre analýzu periodických funkcií, často sa používa na analýzu neperiodických signálov (napr. biomedicínskych, zvukových, seizmických, atď.), pričom neperiodický signál prechádza procesom periodifikácie (za periódu signálu sa považuje celý navzorkovaný signál). Fourierova transformácia nám teda umožňuje identifikovať spektrálne zložky signálu – spektrum frekvencií, z ktorých je signál zložený.

Diskrétna Fourierova transformácia vychádza z Fourierovej transformácie, ktorá je určená nasledujúcim vzťahom

$$H(f_n) = \int_{-\infty}^{\infty} h(t) e^{2\pi i f_n t} dt \tag{1}$$

kde $f_n = \frac{n}{N\Delta}, n = 0,..,N-1,$ $h(t)$ je analyzovaná funkcia. Aproximáciou integrálu definovaného vzťahom (1) diskrétnou sumou dostávame vzťah pre výpočet diskrétnej Fourierovej transformácie

$$H_n \equiv \sum_{k=0}^{N-1} h_k e^{\frac{2\pi i k n}{N}} = \sum_{k=0}^{N-1} W^{nk} h_k \tag{2}$$

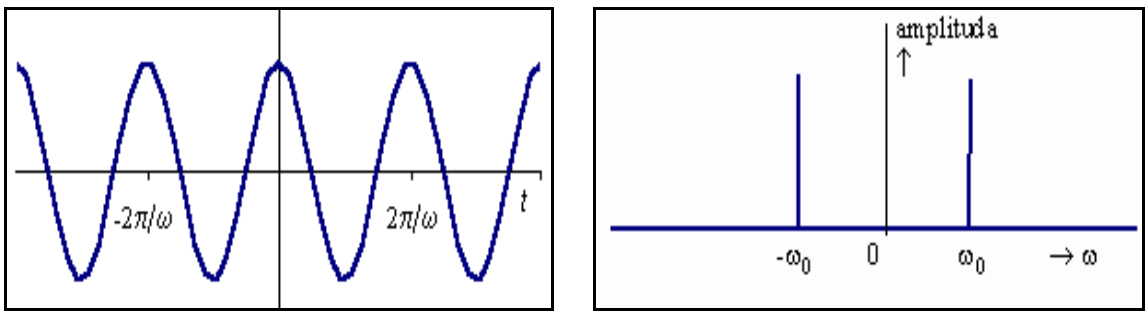
kde H_n je *n*-tá komponenta Fourierovej transformácie, *H* je periodická s periódou N a symetrická, *N* určuje počet diskrétnych vzoriek signálu, je h_k *k-tá* vzorka vzorkovaného signálu, $W^{nk} = e^{\frac{2\pi i}{N}}$, $n = 0,..,N-1$.

Predchádzajúci algoritmus (DFT) pre výpočet diskrétnej Fourierovej transformácie je pomerne jednoduchý, čo má za následok kvadratickú časovú zložitosť $O(n^2)$. Z tohto dôvodu bola Fourierova transformácia veľmi málo používaná v praktických aplikáciách.

V roku 1942 Danielson a Lanczos vypracovali verziu tohto algoritmu označovanú pod názvom **rýchla Fourierova transformácia**. Časová zložitosť tohto algoritmu je tento krát logaritmická *O(N log₂ N)*. Danielson a Lanczos ukázali, že diskrétna Fourierova transformácia dĺžky N môže byť počítaná ako suma dvoch Fourierových transformácií dĺžky *N/2*. Jedna môže byť počítaná s párnymi bodmi (bodmi nachádzajúcimi sa na párnom mieste) a druhá nepárnymi bodmi (bodmi nachádzajúcimi sa na nepárnom mieste) pôvodnej Fourierovej transformácie, čo je vyjadrené vzťahom (3).

$$
\begin{aligned}
F_k &= \sum_{j=0}^{N-1} e^{\frac{2\pi ijk}{N}} f_j = \\
&= \sum_{j=0}^{\frac{N}{2}-1} e^{\frac{2\pi ik(2j)}{N}} f_{2j} + \sum_{j=0}^{\frac{N}{2}-1} e^{\frac{2\pi ik(2j+1)}{N}} f_{(2j+1)} = \\
&= \sum_{j=0}^{\frac{N}{2}-1} e^{\frac{2\pi ikj}{\frac{N}{2}}} f_{2j} + W^k \sum_{j=0}^{\frac{N}{2}-1} e^{\frac{2\pi ikj}{\frac{N}{2}}} f_{(2j+1)} = F_k^e + W^k F_k^o
\end{aligned}
\tag{3}
$$

Zo vzťahu (3) je vidieť, že výpočet diskrétnej Fourierovej transformácie F_k (s periódou N) sa redukuje na výpočet zložiek F_k^e a F_k^o (s periódami $N/2$). Tento istý spôsob môžeme použiť pri výpočte F_k^e, tak že budeme túto zložku počítať z $N/4$ párnych bodov a $N/4$ nepárnych bodov z bodov použitých pri výpočte F_k^e. Teda F_k^e sa bude skladať zo zložiek F_k^{ee} a F_k^{eo}, atď. až kým bude potrebné počítať 1-bodovú transformáciu. Pričom Fourierova transformácia dĺžky 1 (s periódou dĺžky 1) je tá istá hodnota. Podobne sa postupuje pri výpočte zložky F_k^o. Výhoda tohto algoritmu je teda v jeho rekurzivite. Avšak obmedzením je počet bodov vstupujúcich do transformácie, ktorý musí byť $N = 2^r$, $r > 0$.



Obr. 3 – Graf a spektrum kosínusovej vlny

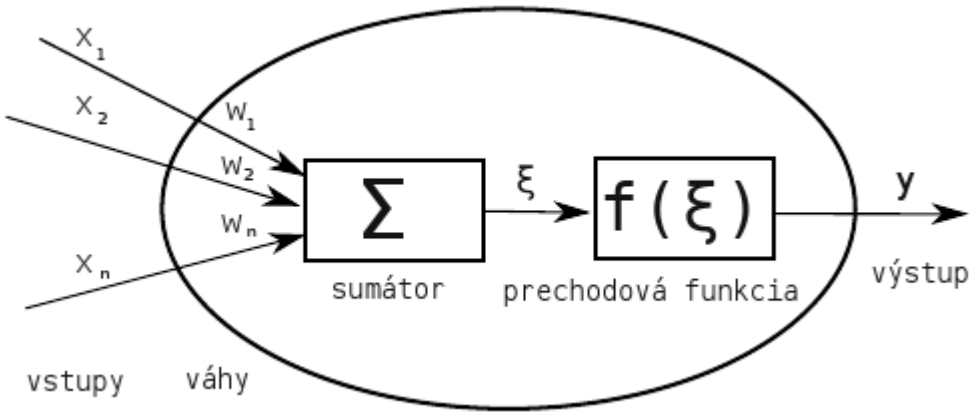
2 NEURÓNOVÉ SIETE

Počítač sa veľmi dobre hodí pre riešenie množstva bežných úloh. Je celkom rýchly a vykonáva presne to, na čo je naprogramovaný. Nanešťastie vám nemôže pomôcť, keď vy sami celkom presne nerozumiete problému, ktorý chcete riešiť. A čo je ešte horšie, štandardné algoritmy nepracujú dobre s porušenými alebo nekompletnými dátami. V reálnom svete je práve toto často jediný dostupný druh dát. Riešením je použitie umelých neurónových sietí, výpočtového systému, ktorý je schopný sám sa učiť.

Neurónová sieť je jedným z výpočtových modelov používaných v umelej inteligencii. Jej vzorom je chovanie odpovedajúcich biologických štruktúr. Umelá neurónová sieť je štruktúra určená pre distribuované paralelné spracovanie dát.

Skladá sa z umelých neurónov, ktorých predobrazom je biologický neurón. Neuróny sú vzájomne prepojené a navzájom si predávajú signály a transformujú ich pomocou určitých prenosových funkcií. Majú ľubovoľný počet vstupov, ale len jeden výstup.

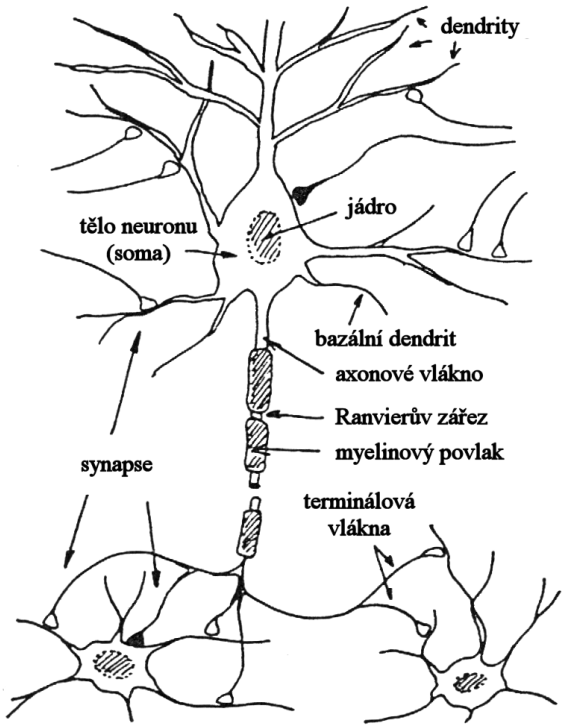
Umelý neurón je v podstate jednoduchá jednotka, ktorá vynásobí všetky vstupy ich váhami (menia sa behom učenia) a takto získané hodnoty sčíta. Výslednú hodnotu dosadí do prenosovej funkcie (funkcia, ktorá určí odozvu na vstupný podnet) a výstup tejto funkcie je výstupom z neurónu, ktorý slúži ako vstup do ďalších neurónov.



Obr. 4 – Model umelého neurónu

Biologický neurón je omnoho zložitejší ako umelý neurón. Typický neurón má telo rozdelené na vstupnú časť tvorenú dendritmi, často aj telom bunky, vodivú časť tvorenú axónom (neuritom) a výstupnú časť, ktorú tvoria nervové zakončenia na konci rozvetveného axónu.

Signál z presynaptických neurónov je prijímaný na dendritoch, zakončených veľkým počtom krátkych výbežkov - tŕňov. Tŕňov môže byť až o niekoľko násobne viac, ako dendritov. Behom celého života neurónu sa môžu tieto tŕne vytvárať a je možné, že ich počet a veľkosť súvisí s priebehom učenia. Vzruch sa od dendritov šíri po tele neurónu smerom k axónu. Axón (okrem koncovej časti) je obalený myelínovou pošvou, ktorá urýchľuje šírenie signálu. Na konci sa axón opäť rozvetvuje a jednotlivé vlákna prechádzajú do tzv. butonov. Medzi butonom a postsynaptickým dendritom, resp. tŕňom je synaptická štrbina (cca 10 - 40 nm). Elektrický signál, ktorý sa šíri telom neurónu, na synapsii spúšťa sekréciu neurotransmiteru. Neurotransmiter vyvoláva na postsynaptickej membráne dendritu excitačnú resp. inhibičnú reakciu. Ak sú prichádzajúce signály na postsynaptickej membráne dostatočne silné, dochádza k depolarizácii a postsynaptický neurón vysiela signál.



Obr. 5 – Model biologického neurónu

V súčasnosti patria neurónové siete medzi významnú časť počítačovo orientovanej umelej inteligencie, kde zaujali postavenie univerzálneho matematicko-informatického prístupu ku štúdiu a modelovaniu procesov učenia. Okrem umelej inteligencie majú neurónové siete nezastupiteľné uplatnenie tiež v kognitívnej vede, lingvistike, neurovede, riadení procesov, prírodných a spoločenských vedách, kde sa pomocou nich modelujú nie len procesy učenia a adaptácie, ale aj široké spektrum rôznych problémov klasifikácie objektov a taktiež problémov riadenia zložitých priemyselných systémov.

Neurónové siete je možné použiť na riešenie veľkého množstva úloh z oblasti klasifikácie, predikcie, optimalizácie. Mimo iné sa používajú aj pre rozpoznávanie a kompresiu obrazov alebo zvukov, predpovedanie vývoja časových udalostí (napr. burzových indexov), niekedy dokonca k filtrovaniu spamu. V lekárstve slúži k prehlbovaniu znalostí o fungovaní nervových sústav živých organizmov.

2.1 História neurónových sietí

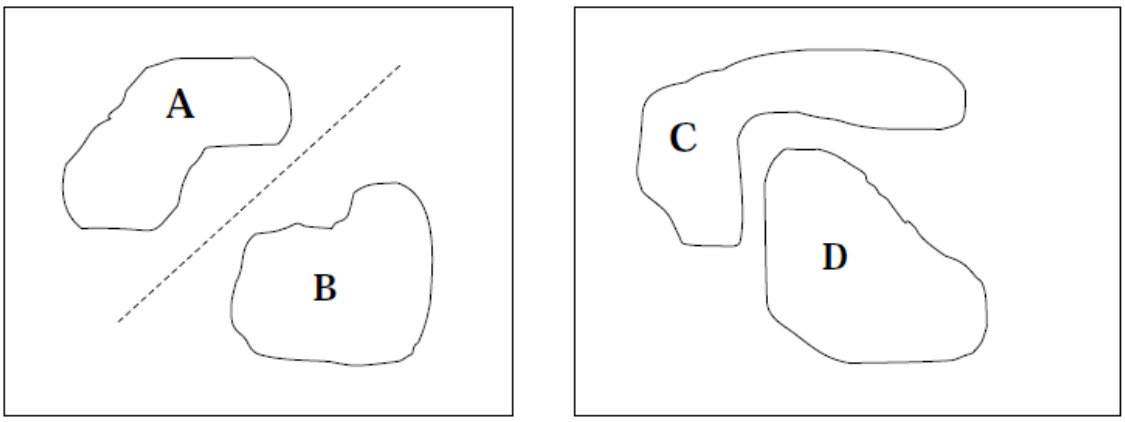
Za počiatok vzniku odboru neurónových sieti je považovaná práca Warrena McCullocha a Waltera Pittse z roku 1943, ktorí vytvorili veľmi jednoduchý matematický model neurónu. Číselné hodnoty parametrov v tomto modelu boli prevažne bipolárne, tj. z množiny {– 1, 0, +1}. Ukázali, že najjednoduchšie typy neurónových sietí môžu v princípe realizovať ľubovoľnú aritmetickú alebo logickú funkciu.

V roku 1949 bola vydaná kniha The Organization of Behavior od Donalda Hebba. Poskytla návod ako používať učiace pravidlo pre synapsie neurónov (interface). Ovplyvnil veľa ďalších vedcov napriek tomu 40.-50. roky nepriniesli zásadný pokrok v oblasti neurovýpočtov. V roku 1951 bol postavený prvý neuropočítač menom Snark a u jeho zrodu stál Marvin Minsky. Bol veľmi úspešný z technického hľadiska, automaticky adaptoval váhy, ale nikdy sa nestal užitočným.

Frank Rosenblatt zovšeobecnil model neurónu v roku 1957 na tzv. perceptrón, ktorý počítal s reálnymi číslami. Jedná sa o pevnú architektúru jedno vrstvovej siete s n vstupnými a m výstupnými neurónmi. Zároveň navrhol učiaci algoritmus, ktorý v konečnom čase našiel odpovedajúci váhový vektor parametrov nezávisle na počiatočnej konfigurácii. V tomto prípade sa reálne stavy neurónov vo vstupnej vrstve nastavili na vstup siete a výstupní neuróny počítali svoj binárny stav, ktorý určil výstup siete rovnakým

spôsobom ako obecný neurón. Na základe tohto výskumu spolu s Charlesem Wightmanem zostrojil behom rokov 1957 a 1958 druhý neuropočítač. Volal sa Mark I Perceptron a bol navrhnutý pre rozpoznávanie znakov. Znak sa premietal na svetelnú tabuľu. Tá bola monitorovaná maticou 20x20 fotosnímačov. Intenzita celkového počtu 400 snímačov bola vstupom do neurónovej siete perceptrónov. Počítač mal odpovedať či sa jedná napríklad o znak "A" alebo "B" atď. Z ďalších parametrov tohto počítače bolo 512 adaptovateľných váhových parametrov, ktoré boli realizované poľom 8x8x8 potenciometrov. Hodnota odporu u každého potenciometru odpovedala príslušnej váhe neurónu. Zaujímavosťou bolo nastavovanie váhy pomocou elektrických motorčekov. Učiace pravidlo riadil analógový obvod. Vďaka úspechu sa oblasť neuropočítačov dostala do každodennej pozornosti.

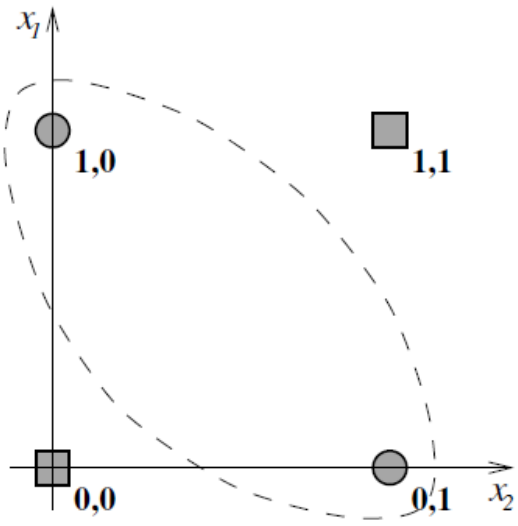
ADALINE sa objavuje ako ďalší typ neurónového prvku, ktorý bol veľmi podobný perceptrónu. Aktívna dynamika sa u tohto modelu líšila tým, že výstupy siete boli obecne reálne a jednotlivé. Adaline realizovali lineárnu funkciu. Bernard Widrow a jeho študenti boli autori ADAptívneho LIneárneho NEurónu. Činnosť Adaline má jednoduchú geometrickú interpretáciu. Jednotlivé vstupné podnety $x_1,x_2,...,x_m$ môžu predstavovať hodnoty vstupných atribútov nejakého objektu (napr. teplota, výška, váha ...). Každý objekt je potom možné reprezentovať ako bod $x = x_1,x_2,...,x_m$ v m-rozmernom priestore. Bod x leží v jednej z dvoch častí priestoru oddelených od seba rozdeľujúcou nad rovinou (pre $m = 2$ sa pohybujeme v rovine a rozdeľujúca nad rovina je priamka). Body ležiace v jednej časti priestoru môžeme považovať za obrazy objektov patriacich do rovnakej triedy. Adaline môžeme teda považovať za lineárny klasifikátor objektov do dvoch tried.



Obr. 6 – Lineárne separabilné a lineárne neseparabilné problémy

Prelomom 50.-60. rokov zaznamenal úspech Karl Steinbuch, ktorý vyvinul model binárnej asociatívnej pamäte. Na rozdiel od pamäti klasických počítačov, kde kľúč k vyhľadaniu položky v pamäti je adresa, u asociatívnej pamäti dochádza k vybaveniu určitej udalosti či informácie na základe jej čiastočnej znalosti. Rozvoj zaznamenal dva typy asociatívnych pamätí a to auto asociatívne a hetero asociatívne. U prvého druhu šlo o spresnenie vstupnej informácie a u hetero asociatívnej dochádza k vybaveniu určitej združenej informácie na základe vstupnej asociácie. Rozdiel bol, že miesto afinnych kombinácií počítal len lineárnu kombináciu vstupov. Jedná sa o označenie jedného zo vstupu do neurónu, ktorý má nastavenú váhu na záporný vnútorní potenciál.

Napriek množstvu nesporných úspechov sú siete a neuropočítače používané len z experimentálneho hľadiska. Nie príliš veľká publicita tiež nepriala rozvoju odboru. Tieto skúsenosti nepriamo ovplyvnili veľa ďalších vedcov a inžinierov. Najlepší z najlepších odchádzajú do iných oblastí výskumu. Marvin Minsky a Seymour Papert využili svoj vplyv a diskreditovali celý odbor. Kniha Perceptrons od vyššie uvedených autorov opäť poukázala na nemožnosť perceptrónu spočítať funkciu XOR. To čiastočne pozastavilo prísun financií do tejto oblasti. Jednotlivé projekty boli prevažne realizované zo súkromných nadácií.



Obr. 7 – Grafické znázornenie funkcie XOR

Až v 80. rokoch sa objavilo pár zaujímavých talentov ako Shun-Ichi Amari, James Anderson, Kunihiko Fukushima, Stephen Grossberg, Teuvo Kohonen a David Willshaw. Predložili vlastné grantové projekty na neuropočítače a ich využitie. Vyniká grantová

agentúra DARPA (Defense Advanced Research Projects Agency). To, čo bolo zdanlivo mŕtve, sa opäť dostáva do popredia. Dôkazom toho je John Hopfield, ktorý vo svojich prácach datovaných roku 1982 a 1984 ukázal súvislosť niektorých modelov s fyzikálnymi modelmi magnetických materiálov. Neskôr boli podľa neho nazvané tzv. Hopfieldove siete, ktoré fungujú na princípe auto asociatívnej pamäti. Sieť má pevnú topológiu s *n* neurónmi, ktoré sú zapojené cyklicky. Všetky neuróny sú zároveň výstupné.

V roku 1986 bádatelia združený do PDP skupiny (Parallel Distributed Processing Group) zo zástupcami Davidom Rumelhartom, Geoffreyom Hintonom a Ronaldom Williamsom publikovali učiaci algoritmus spätného šírenia chyby tzv. backpropagation pre viacvrstvovú neurónovú sieť. Podarilo sa im vyriešiť problém, ktorý Minsky označil za nevyriešiteľný a tým pozastavil vývoj neurónových sietí. Je zároveň jedným z najpoužívanejších algoritmov a súčasťou 80% všetkých neurosystémov. Model bol zovšeobecnením siete perceptrónov pre acyklickú architektúru so skrytými vrstvami. Sieť sa učila pomocou algoritmu spätného šírenia chyby. Nejednalo sa však o nový objav, ale o druh siete vymyslenej a publikovanej v období ticha Arthurom Brysonom a Yu-Chi Ho, Paulom Werbosom a Davidom Parkerom. Backpropagation bol využitý a aplikovaný v systéme NETtalk vyvinutým Terrencom Sejnowskim a Charlesom Rosenbergom. Systém úspešne konvertoval anglicky napísaný text na hovorenú reč. Bol priamym konkurentom systému DECtalk, ktorý obsahoval stovky pravidiel vytváraných lingvistami po celé desaťročia. Už z tohto malého príkladu je vidieť nesporná sila neuropočítaču.

V roku 1987 sa v San Diegu konala prvá väčšia konferencia s výhradným zameraním na neurónové siete. IEEE International Conference on Neural Networks uvítala cez 1700 účastníkov s jasným výsledkom založenie medzinárodnej spoločnosti pre výskum neurónových sietí INNS (International Neural Network Society). Vzniklo množstvo svetoznámych časopisov ako sú napr. Neural Networks, Neural Computation, IEEE Transactions on Neural Networks.

V dnešnej dobe sa predovšetkým výpočty simulujú na klasických PC staniciach. Niektoré nádejné neurónové siete sú zavádzané do praxe, či už v podobe automatického riadenia alebo náhrada u meracích prístrojov. Hopfieldova sieť sa dá jednoducho použiť na rozpoznávanie obrazu.

2.2 Rozdelenie neurónových sietí

Všetky typy neurónových sietí sú zložené z rovnakých stavebných jednotiek – neurónov. Môžu obsahovať rôzne prenosové funkcie, spojenie medzi sebou a učiaci algoritmus. To určuje o aký typ siete sa jedná.

Umelé neurónové siete sa delia podľa niekoľkých hlavných kritérií.

Podľa metódy šírenia informácie:

- siete s dopredným šírením (feed forward)
- rekurentné neurónové siete

Podľa počtu vrstiev:

- jedno vrstvová sieť (Kohonenova sieť, Hopfieldova sieť),
- viac vrstvová sieť (ART sieť, perceptrón, klasická viac vrstvová sieť s algoritmom backpropagation)

Podľa štýlu učenia:

- deterministický (algoritmus backpropagation)
- stochastickým (náhodné nastavovanie váh)

Podľa spôsobu učenia:

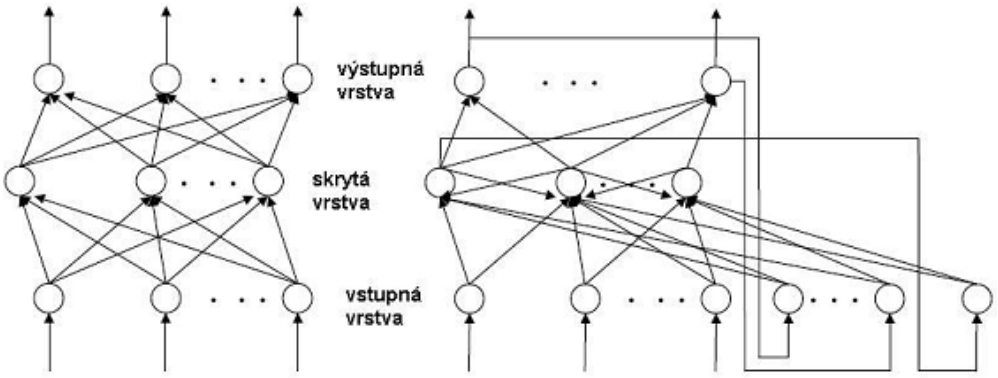
- s učiteľom (algoritmus backpropagation)
- bez učiteľa (Hopfieldova sieť)

2.2.1 Podľa metódy šírenia informácie

Dopredné neurónové siete sa vyznačujú tým, že v nich existujú iba dopredné spojenia medzi neurónmi. Každý neurón jednej vrstvy vysiela signály na každý neurón nasledujúcej vrstvy. Spojenia do predchádzajúcej vrstvy ani v rámci jednej vrstvy neexistujú.

Rekurentné neurónové siete majú komplikovanejšie rozdelenie vrstiev - niektoré vrstvy a neuróny sú zároveň vstupné a zároveň výstupné. Signál môže putovať cez sieť v oboch smeroch. Sú význačné svojou dynamickosťou a ich stav sa mení pokiaľ sa

synaptické váhy neustália na hodnote s čo najmenšou chybou siete. V takomto ustálení zotrvávajú až do zmeny vstupu, kedy sa znova hľadá rovnovážny stav.

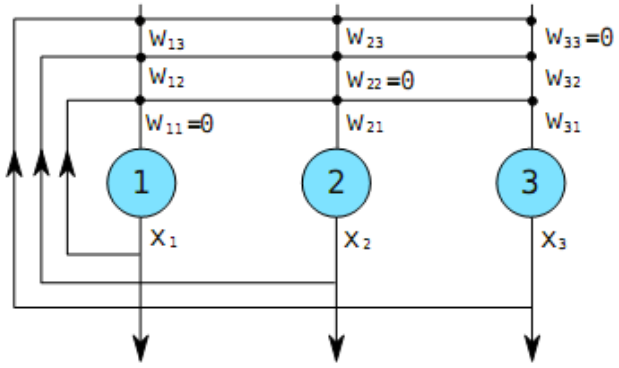


Obr. 8 – Dopredná a rekurentná neurónová sieť

2.2.2 Podľa počtu vrstiev

Rozlišujeme z koľkých vrstiev sa daná neurónová sieť skladá. Existujú siete z jednou, dvomi a viac vrstvami. Siete z jednou alebo dvoma vrstvami bývajú väčšinou špeciálne siete ako napríklad Hopfieldova alebo Kohonenova sieť, ktoré majú svoj špeciálny učiaci algoritmus a topológiu. Pre topológiu sietí väčšinou platí pravidlo, že každý neurón je spojený s každým ďalším neurónom vo vyššej vrstve. Zvláštnosťou je Hopfieldova sieť, v ktorej je spojený každý neurón so všetkými ostatnými. Každý spoj je ohodnotený váhami, ktoré vyjadrujú aký význam má tento spoj pre daný neurón.

Pre viacvrstvové siete sa obvykle používa klasický učiaci algoritmus backpropagation. Viacvrstvovú neurónovú sieť si môžeme predstaviť ako sieť zloženú z niekoľkých vrstiev, pričom každá vrstva sa skladá z ľubovoľného počtu neurónov. Problém vhodného počtu vrstiev a počtu neurónov v jednotlivých vrstvách bol vyriešený až v druhej polovici dvadsiateho storočia.



Obr. 9 – Hopfieldova sieť s tromi neurónmi

2.2.3 Podľa štýlu učenia

Štýl učenia siete v podstate znamená to ako sa získavajú správne váhy jednotlivých neurónov. V prípade, že sa váhy zisťujú pomocou nejakého výpočtového algoritmu ako je napríklad algoritmus backpropagation, jedná sa o deterministický štýl učenia.

Ak sú však váhy získavané pomocou generátoru náhodných čísiel hovoríme o stochastickom štýle učenia. Tento spôsob získavania váh siete sa väčšinou využíva len pri inicializácii siete.

2.2.4 Podľa spôsobu učenia

Učiace algoritmy je možné podľa typu učiaceho procesu rozdeliť na učenie s učiteľom a učenie bez učiteľa.

Učenie s učiteľom (kontrolované učenie, supervised learning) je metóda učenia neurónovej siete, pri ktorej je sieti poskytovaná informácia ako má na prichádzajúce vstupy reagovať. Sieť sa snaží prispôsobiť svoje váhy tak aby sa momentálna výstupná informácia čo najviac podobala požadovanému originálu. Príklady, ktoré tvoria učiace dáta sú reprezentované ako vstupy a k ním prislúchajúce výstupy. Vstupom môže byt *n*-rozmerný vektor a výstupom iba jedna hodnota. Úlohou učiaceho algoritmu je minimalizovať učiacu chybu, ktorá je v tomto prípade definovaná zvyčajne ako euklidovská vzdialenosť medzi vektorom skutočných výstupov neurónovej siete a vektorom požadovaných výstupov.

Učenie zvyčajne končí ak je dosiahnutá nejaká minimálna učiaca chyba. Výsledkom učenia sú hodnoty jednotlivých neurónových váh. Tento spôsob môžeme prirovnať k učeniu dieťaťa čítať písmená tak, že mu ukážeme obrázok písmena (vstup) a vyslovíme A (požadovaný výstup). Dieťa sa nás snaží napodobniť a tým sa učí rozpoznávať písmená. Učenie s učiteľom teda neznamená onen dospelý, ale spôsob predkladania vstupov a vyžadovanie napodobnenia požadovaných výstupov.

Učenie bez učiteľa (nekontrolované učenie, unsupervised learning) je metóda učenia neurónovej siete spôsobom, keď sieti nedefinujeme ako má reagovať na daný vstup, čiže neposkytneme informáciu o požadovanom výstupe. Ako má sieť na prichádzajúce vstupy reagovať je ponechané na charaktere učiaceho algoritmu. Učenie zvyčajne končí, keď neurónová sieť dosiahne stabilný stav.

2.3 Učenie neurónovej siete

Jeden zo základných predpokladov pre správnu funkciu neurónovej siete je jej naučenie (adaptácia) na daný problém. Väčšinou sa jedná o klasifikáciu prvkov do rôznych tried. Triedou rozumieme množinu, ktorá zahrnuje *x* prvkov so spoločnými vlastnosťami.

Činnosť neurónovej siete teda môžeme rozdeliť na dve fázy:

- aktivačná (vybavovacia) fáza
- adaptačná (učiaca) fáza

Novo vytvorenú, ale tiež akúkoľvek nenaučenú neurónovú sieť môžeme považovať za skupinu neurónov, ktoré nevedia vyriešiť žiadny problém. Aby mohla byť sieť používaná pre riešenie problému, kvôli ktorému vznikla je potrebné ju najskôr naučiť podobne ako ktoréhokoľvek živého jedinca. Z toho dôvodu boli vyvinuté algoritmy, pomocou, ktorých sa príslušná sieť dokáže naučiť na danú množinu informácií. Algoritmus učenia k svojej činnosti potrebuje tzv. trénovaciu množinu.

Trénovacia množina je skupina vektorov obsahujúcich informácie o danom probléme pre učenie. Pokiaľ učíme sieť štýlom učenia s učiteľom tak trénovacia množina obsahuje dvojice vektorov vstup a výstup. Ak učíme sieť bez učiteľa obsahuje trénovacia množina len vektory vstupných informácií. Ak používame len aktivačnú fázu učenia siete hovoríme o vybavovaní. Túto fázu používame vtedy, keď je sieť už naučená. Striedanie aktivačnej a adaptačnej fáze je vlastné učenie siete.

2.3.1 Aktivačná fáza

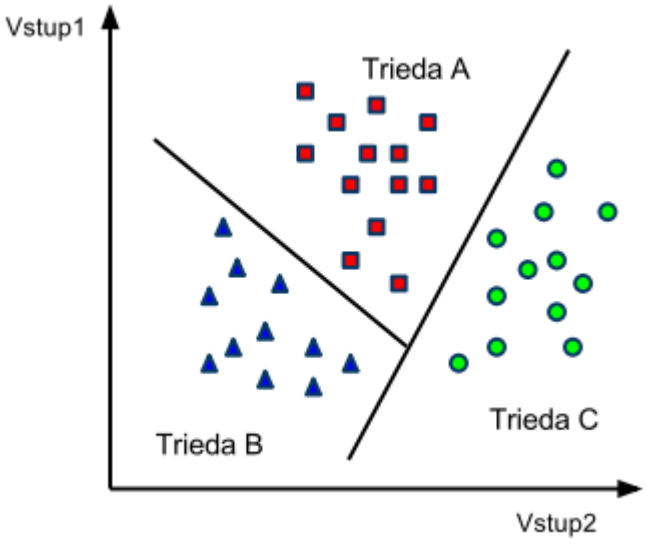
Je to proces, pri ktorom sa predložený vektor informácií na vstup siete, prepočíta cez všetky spoje siete spolu s ich ohodnotením váhami až na výstup siete, kde sa objaví odozva siete na tento vektor vo forme výstupného vektoru. Pri učení sa výstupný vektor porovná s originálnym vektorom a rozdiel medzi nimi (lokálna odchýlka, chyba) sa uloží do pamäťovej premennej.

2.3.2 Adaptačná fáza

Je proces, pri ktorom je minimalizovaná lokálna odchýlka siete tak, že sa prepočítavajú váhy jednotlivých neurónových spojov smerom od výstupu až k vstupu siete za účelom čo najväčšej podobnosti výstupného vektoru s vektorom originálnym.

Po dokončení adaptačnej fáze sa opäť opakuje aktivačná fáza. Ďalší získaný rozdiel lokálnej odchýlky sa pripočíta k predošlému a tento proces sa znovu opakuje rovnakým postupom až sa prejde celá trénovacia množina vektorov. Jeden trénovací cyklus celej trénovacej množiny sa nazýva **epocha** učenia siete. Suma všetkých lokálnych chýb za jednu epochu sa nazýva **globálna chyba** (odchýlka). Ak je globálna chyba menšia ako chyba pri predchádzajúcej epoche tak sa sieť učí. Proces učenia sa skončí vtedy, keď je hodnota globálnej chyby menšia ako nami požadovaná chyba.

Z tohto môžeme usúdiť, že učenie siete nie je nič iné len prepočítavanie daných informácií zo vstupu siete na výstup a naopak. Pri učení sa vo vstupnom priestore vytvárajú zhluky bodov, ktoré predstavujú jednotlivé členy triedy, pričom každý zhluk predstavuje jednu triedu. V praxi to funguje tak, že mame zadaných niekoľko tried a z každej triedy vyberieme množinu zástupcov reprezentovaných informáciami vo forme vektorov. V tomto prípade sa učiaci algoritmus snaží nájsť takú kombináciu váh jednotlivých neurónových spojení, že ak položíme pri vybavovacej fáze na vstup siete vektor popisujúci určitú triedu na výstupe bude vektor, ktorý danú triedu definuje.



Obr. 10 – Nelineárna hranica medzi triedami

To či sa sieť naučí reagovať správnymi odozvami na dané podnety, závisí na množstvu vektorov v trénovacej množine a ich veľkosti, na topológii siete, odlišnosti vlastností jednotlivých tried, príprave trénovacej množiny a iných okolnostiach. Schopnosť siete priraďovať vstupné členy jednotlivým triedam je daná tým, že sieť v podstate počíta vzdialenosť daného členu od členov už priradených do triedy a na základe toho usudzuje do akej triedy vstupný vektor patrí.

2.4 Matematický model umelého neurónu

Matematická reprezentácia neurónu je oproti biologickej štruktúre podstatne zjednodušená a pozostáva z týchto základných častí:

- *n* reálnych vstupov $x_1, ..., x_n$
- synaptické váhy $w_1, ..., w_n$
- aktivačná funkcia neurónu *f*, ktorej výsledkom je výstup z neurónu
- vnútorný potenciál neurónu ξ
- prah neurónu θ

Do *i-tého* neurónu vstupuje *n-výstupných* aktivít predchádzajúcich neurónov a každý spoj je ohodnotený váhovým koeficientom w_{ij}, ktorý určuje váhu informácie z *j-tého* predsynaptického neurónu. Sumačná funkcia jednotlivých výstupov $x_1, x_2, ..., x_n$ z predsynaptických neurónov ohodnotených váhovými koeficientmi $w_{i1}, w_{i2}, ..., w_{in}$ s pripočítaním hodnoty prahového koeficientu sa nazýva neurónový potenciál ξ. Prahový koeficient θ_i predstavuje vstup z okolitého sveta. Určuje hodnotu vstupného váženého súčtu, pri ktorom je neurón najviac citlivý na zmenu tejto sumy. Formálne sa tiež niekedy označuje ako nultý vstup x_0 s hodnotou 1 a s váhou $w_0 = 0$.

Aktivita (výstup) neurónu je určená funkciou neurónového potenciálu, ktorú nazývame aktivačnou funkciou *f*. Formulujeme teda vzťahy medzi vstupom a výstupom *i-tého* neurónu:

$$x_i = f(\xi_i) \qquad \xi_i = \sum_{j=1}^{n} x_j w_{ij} + \theta$$

2.4.1 Aktivačná funkcia neurónu

Aktivačná funkcia (prenosová funkcia) je funkcia, ktorá transformuje vstupný signál na signál výstupný v intervaloch 0 až 1 alebo -1 až +1, Pre správny chod neurónov a neurónových sietí je dôležité akú aktivačnú funkciu zvolíme. Existujú rôzne funkcie, pri ktorých obecne platí, že ich hodnota má byť v intervale -1 až +1 pri spojitých funkciách (sigmoida, hyperbolický tangens) alebo nespojitých v intervale 0 až 1 (binárna funkcia).

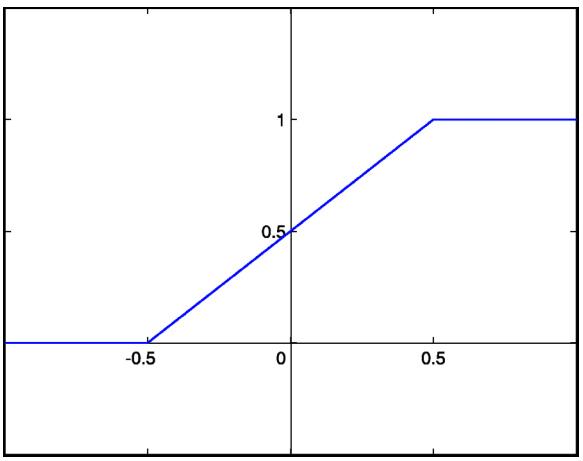
Voľba aktivačnej funkcie závisí na probléme, ktorý chceme riešiť neurónovou sieťou. Ak chceme napríklad rozlíšiť či je výrobok dobrý alebo zlý stačí binárna funkcia. Ak by sme chceli použiť spojitú funkciu musíme rozhodnúť, ktorá jej hodnota znamená dobrý, a ktorá zlý výrobok. Medzi najpoužívanejšie funkcia partia:

- lineárna funkcia (funkcia perceptrón)
- skoková funkcia (binárna funkcia)
- logistická funkcia (sigmoida)
- hyperbolický tangens

2.4.1.1 Lineárna funkcia

Je funkcia, ktorá bola použitá v prvej neurónovej sieti, a ktorá vďaka svojej linearite dokázala riešiť len lineárne separabilné problémy. To viedlo k úpadku záujmu o neurónové siete na skoro 20 rokov. Jej zápis je:

$$f(\xi) = \xi$$

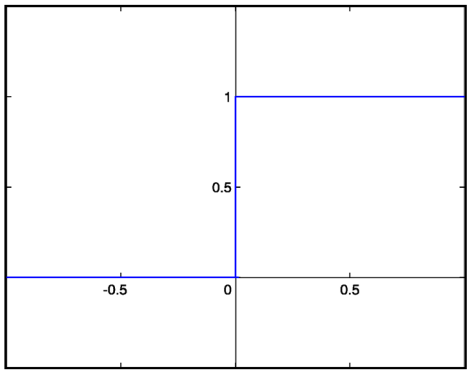


Obr. 11 – Lineárna funkcia

2.4.1.2 Skoková funkcia

Je to dvoj hodnotová funkcia, ktorá môže nadobudnúť len hodnotu 0 alebo 1. Takéto neuróny nazývame McCulloch-Pittsove neuróny. Jej zápis je:

$$f(\xi)=\begin{cases}1 & pre \quad \xi > 0\\ 0 & pre \quad \xi \le 0\end{cases}$$

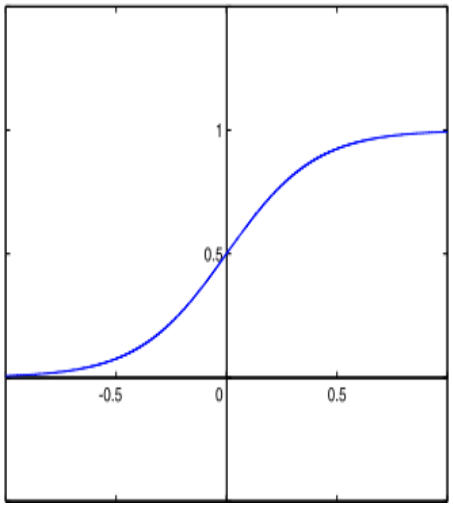


Obr. 12 – Skoková funkcia

2.4.1.3 Logistická funkcia (sigmoida)

Jedna z najpoužívanejších prenosových funkcií, ktorá bola odvodená od prenosovej funkcie biologického neurónu. Jej hodnoty sa blížia 0 v mínus nekonečne a 1 v plus nekonečne. Pre 0 je hodnota 0,5. Výhodou sigmoidálnej prenosovej funkcie oproti skokovej je existencia spojitej prvej derivácie v každom bode.

$$f(\xi)=\frac{1}{1+e^{-\xi}}$$

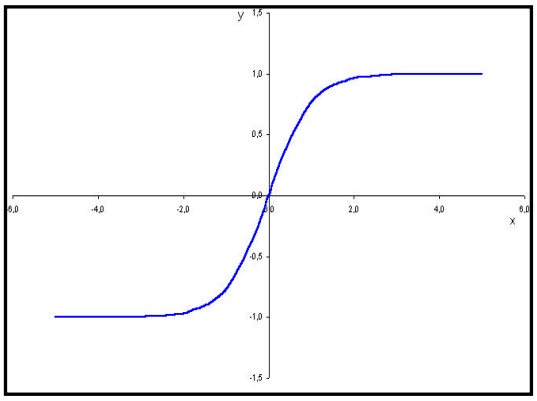


Obr. 13 – Sigmoidálna funkcia

2.4.1.4 Funkcia hyperbolický tangens

Táto funkcia je obdoba logistickej funkcie s tým rozdielom, že môže nadobudnúť hodnôt -1 až +1 čo znamená, že poskytuje mimo iné väčší lineárny úsek okolo počiatku, čo ma tiež svoj význam.

$$\tanh(\xi)=\frac{e^{\xi}-e^{-\xi}}{e^{\xi}+e^{-\xi}}$$

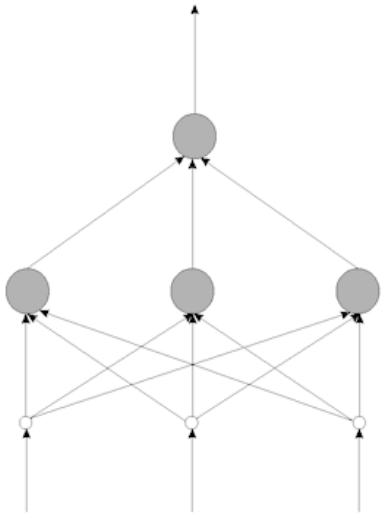


Obr. 14 – Funkcia hyperbolický tangens

2.5 Typy neurónových sietí

2.5.1 Perceptrón

Perceptrón sa skladá z jediného výkonného prvku, ktorý má nastaviteľné váhové koeficienty a nastaviteľný prah. Algoritmus vhodný k nastaveniu parametrov perceptrónu publikoval Rosenblatt v roku 1958. Preceptrón s jedným výkonným prvkom umožňuje riešiť len lineárne separabilné problémy a klasifikovať len do dvoch tried. Ak zväčšíme počet výkonných prvkov pracujúcich v perceptrónu a zväčšíme aj počet jeho vrstiev, je nim možno klasifikovať do viac tried.



Obr. 15 – Trojvrstvová perceptrónová sieť

Úlohou vstupnej vrstvy perceptrónu je rozdeliť vstupní vektor, väčšinou dvojrozmerný do jednorozmerného. Druhú vrstvu tvoria detektory príznakov. Posledná, tretia vrstva obsahuje rozpoznávače vzorov (pattern recognizers alebo perceptrons). Zmenou oproti dvom predchádzajúcim vrstvám je to, že jej váhy nie sú nastavené pevne, ale menia sa pri procese trénovania (učenia).

Prenosová funkcia je skoková. Keďže je funkcia v podstate transformáciou vstupného vektoru na výstup, je definovaná nasledovne. Nech je dané $\{x_1, \ldots, x_n\}$ čo je skutočná reálna množina premenných z R_n. Nech F je množinu funkcií definovaných na členoch množiny R_n. Pokiaľ existuje množina takých váh, že platí rovnice (4), potom je táto funkcia skoková.

$$\psi(x) = F_n\left(\sum_{i=1}^{M} w_i \phi_i(x) + w_{M+1}\right) = F_n\left(\sum_{i=1}^{M} w_i y_i + w_{M+1}\right) \tag{4}$$

Táto funkcia, ako môžeme vidieť z rovnice (4), je realizovaná neurónom od McCullocha-Pittsa s *M* vstupmi a vhodne zvolenými váhami. Neuróny vo vrstve, ktorá je schopná sa učiť, majú spravidla ešte jeden výstup navyše, jeho hodnota je konštantne nastavená na 1. Je to z dôvodu nastavovania prahu neurónu rovnakým spôsobom ako sa nastavujú váhy siete. Vlastný vstupný vektor perceptrónu má tvar (5).

$$y_i = \left(\phi_1\left(x(i)\right), \phi_2\left(x(i)\right), \ldots, \phi_M\left(x(i)\right), 1\right)^T \tag{5}$$

Keď sa príslušné prahy položia rovné 0, vykonáme tak úpravu perceptrónu, ktorá sa používa pri demonštrovaní učenia perceptrónu. Keďže výstupný neurón môže nadobúdať len dve hodnoty, môžeme vstupné vektory priradiť len do dvoch tried.

Keďže potrebujeme kvôli učeniu nastavovať nové váhy a prahy neurónov, existujú pre perceptrón nasledujúce metódy učenia:

- **Metóda koeficientov** - koeficient môže byť fixní či modifikovateľný absolútnou alebo zlomkovou korekciou
- **Gradientná metóda**

2.5.1.1 Učiaci algoritmus perceptrónu

- Váhy sú nastavené náhodne.
- Ak je výstup správny, váhy sa nemenia.
- Ak má byť výstup rovný 1, ale je 0, zvýšia sa váhy na aktívnych vstupoch.
- Ak má byť výstup rovný 0, ale je 1, znížia sa váhy na aktívnych vstupoch.

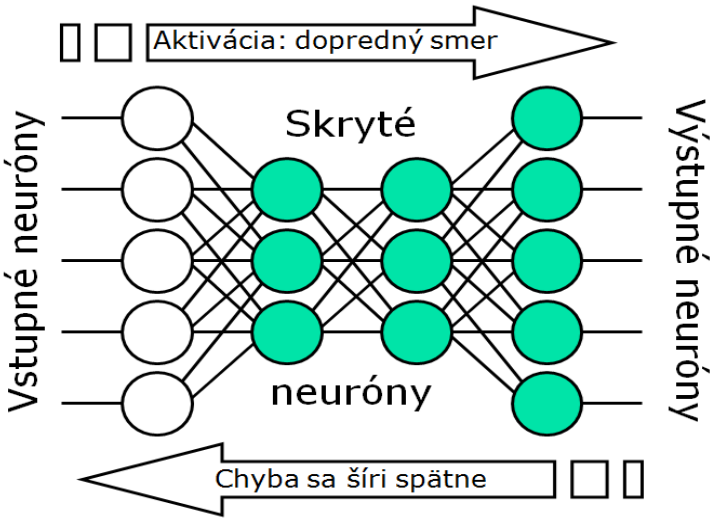
Vstupy sú aktívne vtedy, ak ich hodnota nad prahom nie je nulová. Veľkosť, s akou sa menia váhy závisí na konkrétne zvolenej variante:

- Pri zvyšovaní aj znižovaní sa aplikujú pevné prírastky.
- Prírastky sa menia v závislosti na veľkosti chyby. Takto zrýchlená konvergencia však môže mať za následok nestabilitu učenia.
- Premenné a konštanty sa kombinujú v závislosti na veľkosti chyby.

2.5.2 Viacvrstvová sieť

Viacvrstvová neurónová sieť je zložená z *N* vrstiev, ktoré sú medzi sebou prepojené smerom zo vstupu na výstup. Každý neurón je prepojený zo všetkými neurónmi v predchádzajúcej vrstve a tieto spojenia sú ohodnotené synaptickými váhami. Jedná sa o najrozšírenejší a najpoužívanejší sieť, ako pre klasifikáciu tak pre regresiu (a teda aj predpovedanie časových radov). Táto sieť sa učí pomocou metódy učenia s učiteľom, kde sa najčastejšie používa tzv. algoritmus backpropagation čo znamená spätné šírenie chyby. Ako aktivačná funkcia viacvrstvovej siete sa používa sigmoidálna funkcia.

Okrem základného algoritmu backpropagation existuje množstvo sofistikovaných metód učenia, napr. metóda združených gradientov, Levenbergova-Marquardtova metóda atd. K nevýhodám siete patrí ťažké riešenie problému lokálneho minima a pomerne dlhá doba učenia. Toto sa však dá zlepšiť pomocou momentu, šumu alebo pridaním ďalších neurónov do skrytej vrstvy.



Obr. 16 – Viacvrstvová neurónová sieť

2.5.2.1 Učiaci algoritmus backpropagation

Tento algoritmus opravuje (prenastavuje) synaptické váhy jednotlivých neurónových spojov spätným chodom tak, aby ich veľkosti boli z hľadiska riešeného problému optimálne. Hľadá sa globálne minimum chybovej funkcie. Nastavenie váh prebieha v opačnom smere, než akým sa šíria vstupné informácie. U tohto algoritmu sú tiež dve fázy aktivačná a adaptačná. Aktivačná fáza je používaná pri učení a vybavovaní siete. Je to teda aktivita, pri ktorej sa vstupná informácia dostane na výstup a je modifikovaná

množinou váh a prenosnými funkciami jednotlivých neurónov. Pri adaptačnej fáze je výstupný vektor porovnaný s požadovaným originálom a rozdiel medzi nimi je použitý pre výpočet nových váh. Vypočítajú sa takým spôsobom, že sa najskôr upravia váhy pri výstupných neurónoch a tak sa postupuje späť až k neurónom vo vstupnej vrstve. Potom je adaptačná fáza ukončená a znovu sa opakuje aktivačná fáza.

Chybu vypočítame z nasledujúcich tvrdení. Vychádzajme z toho, že máme trojvrstvovú neurónovú sieť a pripravenú trénovaciu množinu skladajúcu sa z dvojice vektorov $T_s = \{x[k], d[k]\}_{k=1}^{N}$, kde *N* je počet vektorov, *x[k]* je *k-tý* vektor a *d[k]* je výstupný vektor. Odchýlka je daná vzťahom (6).

$$E(k) = \frac{\sum_{j=1}^{m} \left[y_j(k) - d_j(k) \right]^2}{2} \tag{6}$$

Kde $y_j(k)$ je odozva na vstupný vektor a $d_j(k)$ je požadovaný výstupný vektor. Chyba *E(k)* je chyba za jeden vektor cez všetky jeho prvky. Globálna chyba (chyba za epochu) sa vypočíta zo vzorca (7). Kde *N* je počet dvojíc vektorov v trénovacej množine.

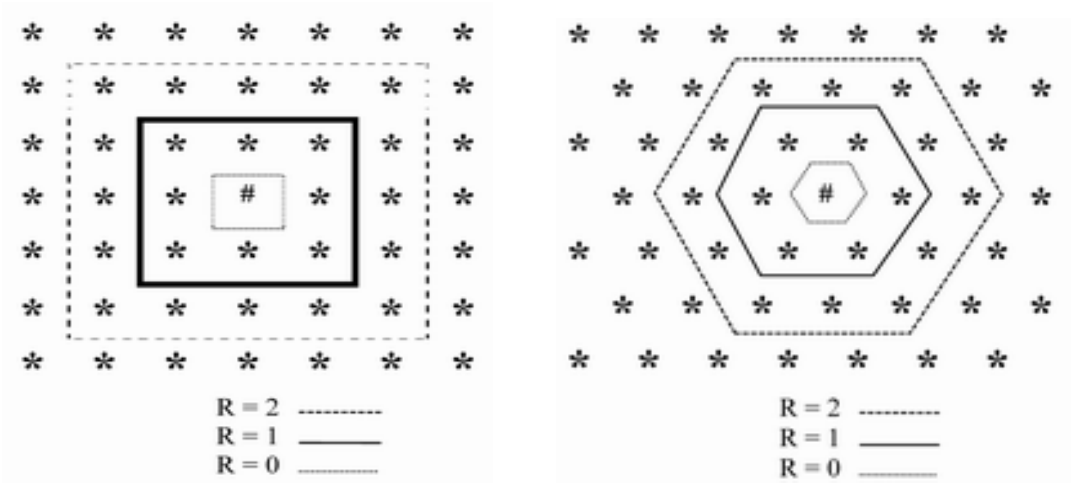
$$E_t = \sum_{k=1}^{N} E_k \tag{7}$$

2.5.3 Kohonenové samoorganizujúce sa mapy (SOM)

Samoorganizujúce sa neurónové siete s učením bez učiteľa sú stále viacej využívané pre rozlíšenie, rozpoznávanie a triedenie neznámych číslicových signálov a dát. Hlavným predstaviteľom sú Kohonenové samoorganizujúce sa mapy (SOM – self organizing maps). Tie sami rozpoznávajú zhodné prvky alebo naopak rozdiely medzi signálmi takže je s nimi možné spracovať úplne neznáme signály a dáta. To je výhoda, ktorá z Kohonenových máp, za cca 20 rokov od ich vzniku, urobila veľmi často využívanú a veľmi obľúbenú neurónovú sieť.

Základný princíp Kohonenovej mapy nie je príliš zložitý, avšak jeho pochopenie vyžaduje trochu predstavivosti. Základ tvorí usporiadaná štruktúra neurónov, kde ku každému prislúcha unikátny vektor koeficientov označované ako váhy *w*. Najčastejšie má

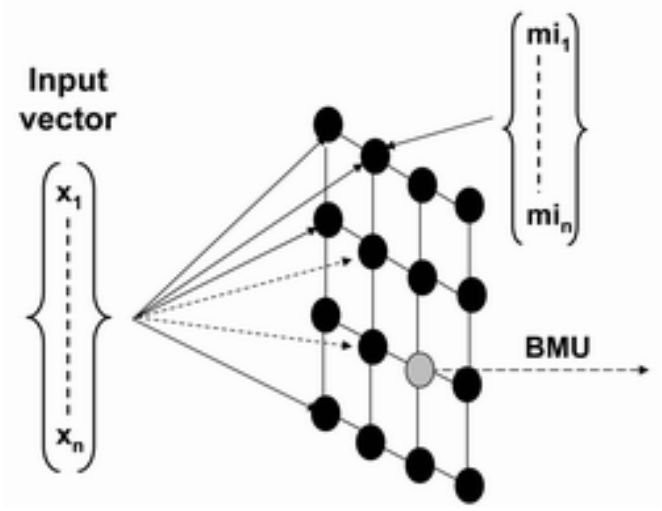
štruktúra formu dvojrozmernej $k = 2$ štvorcovej alebo obdĺžnikovej matice, hexagonálneho útvaru alebo nikdy aj jednorozmerného vektoru $k = 1$.



Obr. 17 – Možné štruktúry usporiadania neurónov Kohonenových máp

Rozmer štruktúry k nemá nič spoločného s počtom váh každého neurónu, teda dimenziou neurónu n. Obvykle platí, že $k < n$, rovnako ako počet neurónov $m < n$. Naopak tvar štruktúry usporiadania neurónov má vplyv na učenie mapy a počet váh je vždy zhodný s počtom parametrov vstupných vzoriek, hodnôt alebo koeficientov vstupujúceho spracovávaného signálu.

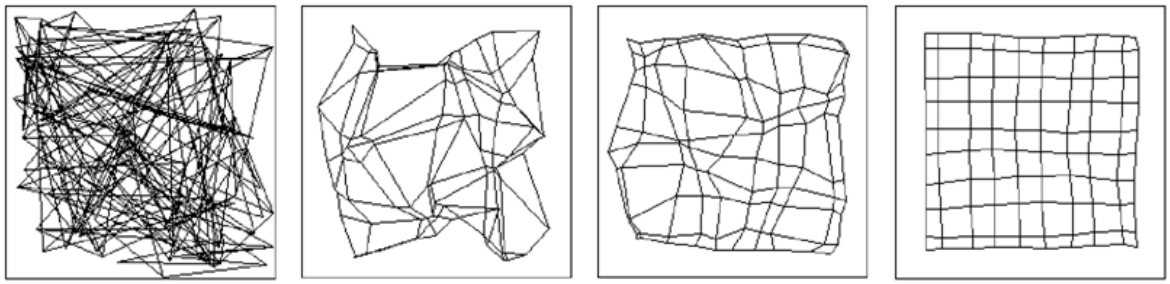
Tvar usporiadania neurónov má vplyv na voľbu tzv. okolia neurónu R, ktoré vymedzuje jeho susedov (najbližšie postavené neuróny). V maticovom usporiadaní neurónov (štvorcové alebo hexagonálne) je veľkosť okolia rovná počtu radov neurónov od centrálneho neurónu. Váhy každého neurónu naopak definujú polohu neurónu v priestore.



Obr. 18 – Štruktúra Kohonenovej mapy s víťazným neurónom BMU

Kohonenové mapy sa učia spôsobom, že matici neurónov sa postupné predkladajú vektory vstupného signálu tak, že sa zvlášť porovnáva rozdiel príslušných hodnôt vektoru váh každého neurónu s hodnotami vektoru vstupného signálu. K vyjadreniu rozdielu sa môže využiť rôznych algoritmov, ale najčastejšie sa používa výpočet euklidovskej vzdialenosti. Výsledkom je teda počet hodnôt, rovný počtu neurónov v štruktúre (napr. 100 hodnôt v matici 10 x 10 neurónov). Potom sa vyberie jediný neurón s najmenším rozdielom hodnôt a označí sa jako tzv. víťazný neurón (winner).

Pri predkladaní prvého vstupného vektoru sa jeho hodnoty porovnávajú s náhodne vygenerovanými hodnotami váh jednotlivých neurónov. Váhy víťazného neurónu sa potom upravujú tak, aby sa čo najviac priblížili hodnotám práve predloženého vstupného vektoru. Pri opätovnom opakovaní dávky učiacich vektorov alebo postupným predkladaním ďalších nových dávok sa učiaci koeficient obvykle znižuje. Spolu s víťazným neurónom sa menia aj tie susedne neuróny, ktoré sú v definovanom okolí *R*. Ich váhy sa upravujú rovnakým spôsobom ako u víťaza. Pri opätovnom opakovaní dávky vstupných vektorov sa môže vykonávať aj znižovanie hodnoty okolia *R* až na *R = 0*, tzn. adaptuje sa len víťaz.



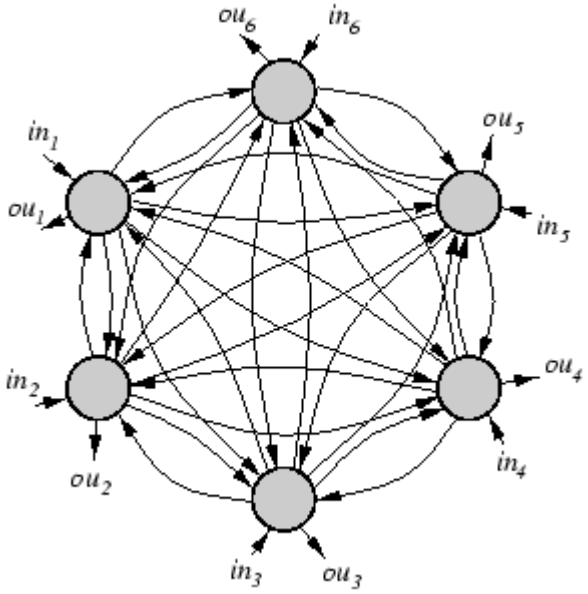
Obr. 19 – Proces učenia Kohonenovej siete

Vo výsledku by sa malo dosiahnuť stavu, keď v maticovej štruktúre neurónov vznikne niekoľko významných centier, tzv. zhluky, medzi ktorými sa výrazne líšia hodnoty váh neurónov. Počet zhlukov by mal byť zhodný s počtom odlišných vlastností alebo parametrov, ktoré Kohonenova mapa našla v predložených dávkach učiacich vektorov. To tiež znamená, že funkčnosť mapy a neurónových sietí obecne, výrazne závisí na zložení signálov a informácií v učiacich dávkach.

2.5.4 Hopfieldová sieť

Hopfieldova sieť so svojím učiacim algoritmom patrí do triedy plne rekurentných sietí s binárnym vstupom, trénovaných bez učiteľa. Po prvýkrát ju popísal John .J. Hopfield v roku 1982. Pracuje ako asociatívna pamäť, ktorej úlohou je uchovávať určité vzory a rekonštruovať nekompletný alebo poškodený vstup tak, aby skonvergoval k jednému z týchto vzorov.

Každý neurón v sieti je prepojený so všetkými ostatnými neurónmi, pričom prepojenia sú symetrické, to znamená, že $w_{ij} = w_{ji}$ pre $i, j = 1, ..., n$. Žiaden neurón však nemá prepojenie samého so sebou, teda $w_{ii} = 0$ pre $i = 1, ..., n$.



Obr. 20 – Hopfieldova sieť so 6 neurónmi

Neuróny sa môžu nachádzať v dvoch stavoch. S aktivačnou hodnotou +1 a s aktivačnou hodnotou -1. Stav *i-tého* neurónu označíme s_i. Teda platí $s_i = \pm 1$. Aktivačná hodnota každého neurónu je daná vzťahom:

$$s_i = g(h_i) = g\left(\sum_{j=1}^{n} w_{ij} x_j - \theta_i\right)$$

Kde h_i je celkový vstup do *i-tého* neurónu a $x_1, ..., x_n$ sú vstupné hodnoty neurónov, ktoré spolu tvoria vstupný vektor $x = (x_1, ..., x_n)$, pričom platí, že $x_i = \pm 1$. Vektor aktivačných hodnôt (stavov) všetkých neurónov $s = (s_1, ..., s_n)$ tvorí stav siete.

Práca siete spočíva v tom, že stavy neurónov v čase *t* sú opäť použité na vstupe siete v čase *t +1*, takže $x_i(t+1) = s_i(t)$ pre $i=1, ..., n$ a teda sieť v každom časovom kroku upravuje stavy neurónov podľa pravidla:

$$s_i(t+1) = g\left(\sum_{j=0}^{n} w_{ij} x_j(t+1)\right) = g\left(\sum_{j=0}^{n} w_{ij} s_j(t)\right)$$

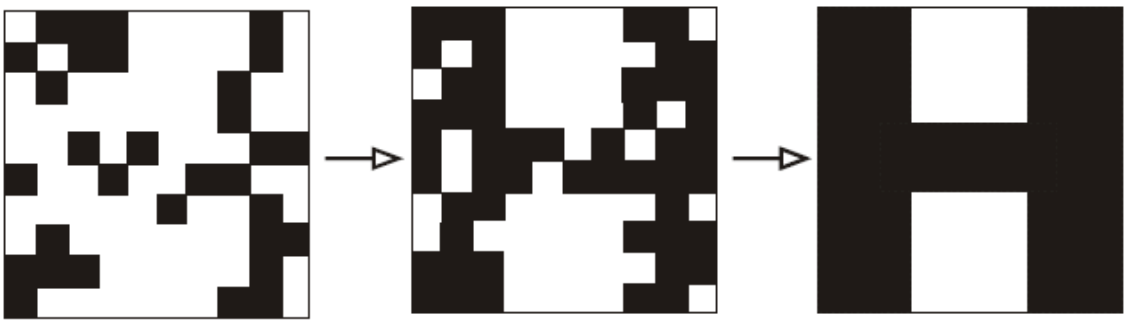
Tento postup sa opakuje až kým sa stav siete nezmení v dvoch po sebe nasledujúcich krokoch, teda nastane rovnosť. Tento stav siete $s = (s_1, ..., s_n)$ nazývame stabilným stavom.

Aktivačné hodnoty neurónov po konvergencii tvoria výstup siete. Všetky vektory $y = (y_1, ..., y_n)$, ku ktorým skonverguje sieť, ak na jej vstup podávame rôzne vektory $x = (x_1 ,..., x_n)$, nazývame exemplárnymi vzormi siete. Pre každý exemplárny vzor platí:

$$s_i = g\left(\sum_{j=0}^{n} w_{ij} s_j\right)$$

V každom časovom okamihu môžeme definovať celkovú energiu siete pomocou energetickej funkcie pričom úlohou siete je dosiahnuť stav s čo najmenšou energiou. Energetická funkcia *E* je funkcia *n* premenných $s_1, ..., s_n$ a dosahuje na svojom definičnom obore lokálne maximá a minimá. Bolo dokázané, že funkcia *E* dosahuje svoje lokálne minimá práve pre vektory *x*, ktoré sú exemplárnymi vzormi siete.

Hopfieldove siete nachádzajú veľké uplatnenie najmä pri riešení problémov rozpoznávania vzorov s binárnou reprezentáciou. Takými sú napríklad čierno-biele obrazy reprezentované pomocou matice pixelov, pričom každý bod obrazu má hodnotu -1 (biely) alebo 1 (čierny).



Obr. 21 – Činnost Hopfieldovej siete pri rekonštrukcii poškodeného písmena H

II. PRAKTICKÁ ČASŤ

3 DATABÁZA ZVUKOV

Pre správne naučenie neurónovej siete je veľmi dôležité navrhnúť a vytvoriť trénovaciu množinu odpovedajúcu problému, ktorý chceme pomocou tejto siete riešiť. V tejto práci sa budem zaoberať klasifikáciou zdrojov zvuku. Rozhodol som sa, že pomocou siete, ktorú som vytvoril budem rozpoznávať výstrely jednotlivých zbraní. Ako neurónovú sieť použijem viacvrstvovú perceptrónovú sieť s dopredným šírením využívajúcu algoritmus backpropagation ako spôsob učenia.

Pre zvuky výstrelov zbraní ako trénovaciu množinu som sa rozhodol z dôvodu spätného využitia v priemysle komerčnej bezpečnosti alebo pre armádne potreby. Tento typ neurónovej siete sa veľmi často používa pre rozpoznávanie reči a hovoreného slova. Ja som sa z dôvodov modernosti problému rozhodol pre rozpoznávanie typov zbraní. Výstrely zo zbraní budem zaraďovať do troch hlavných kategórií:

- malé zbrane (revolvery, pištole)
- stredne zbrane (automatické zbrane, guľomety)
- výbušniny (granáty, trhaviny)

3.1 Vytvorenie trénovacej množiny

Trénovacia množina je skupina zvukov, ktoré je možné zaradiť do rovnakej kategórie na základe spoločných vlastností. Aby neurónová sieť dokázala zvuky spoľahlivo rozpoznať je dôležité aby sa charakteristiky zvukov z jednej kategórie odlišovali od zvukov z inej kategórie. Keďže som sa rozhodol použiť neurónovú sieť, ktorá sa učí s učiteľom je nutné aby trénovacia množina obsahovala tak ako vektory vstupnej charakteristiky tak aj vektory určujúce výstup zo siete, teda vektor určujúci kategóriu zvuku. Pri vytváraní správnej trénovacej množiny musíme dbať na to, že pre každú kategóriu musí byť dostatočný počet vzoriek vstupných vektorov aby sa sieť naučila správne zaradiť aj zvuky na, ktoré nebola naučená a sú podobné zvukom z vybranej kategórie. Väčšinou to býva 100 až 150 vzoriek pre každú kategóriu.

Keďže nemám zbrojný preukaz a bohužiaľ ani nemám prístup k veľkému množstvu rôznorodých zbraní a výbušnín, rozhodol som sa, že si svoju trénovaciu množinu zvukov stiahnem v digitálnej forme z internetu. Po dlhom hľadaní na internete som zistil

skutočnosť, že získať kvalitné ukážky výstrelov zo zbraní nie je také jednoduché ako sa spočiatku zdalo. Väčšina internetových stránok ponúkala tieto zvuky za peniaze, ale podarilo sa mi získať pätnásť pre každú kategóriu zvukov zadarmo. Tento počet zvukov nie je pre správne naučenie neurónovej siete dostačujúci.

V reálnom prípade by neboli zvuky výstrelov rovnaké aj keby vychádzali z rovnakého typu zbrane. Rozhodujúcim faktorom je v tomto prípade vzdialenosť, z ktorej bol výstrel uskutočnený. Z toho dôvodu bolo uskutočnene meranie, pri ktorom som, za pomoci môjho vedúceho prace, získal zvuky výstrelov z rôznych vzdialeností. Pomocou tohto merania bol získaný dostatočný počet zvukov pre vytvorenie dosť veľkej trénovacej množiny pre správne naučenie siete.

3.1.1 Získanie zvukov z rôznych vzdialeností

Meranie prebiehalo na letisku v Štípě neďaleko od Zlína. Za pomoci výkonných reproduktorov sme prehrávali jednotlivé zvuky, ktoré boli následne zaznamenávané vysoko citlivými mikrofónmi umiestnenými v rôznych vzdialenostiach.

Obr. 22 – Príprava na prehrávanie zvukov

Zvuky boli prehrávané na prenosnom počítači, ktorý bol pomocou zosilňovaču pripojený na reproduktory. V multimediálnom prehrávači som vytvoril zoznam zo všetkých

zvukových súborov, ktoré som získal z internetu a tie sa následne prehrali všetky za sebou v danom poradí. K zaznamenaniu prehrávaných zvukov sme využili dva vysoko citlivé mikrofóny, ktoré boli umiestnené v rôznych vzdialenostiach od reproduktorov. Počiatočné rozmiestnenie mikrofónov bolo nasledujúce:

- Prvý mikrofón bol vo vzdialenosti 100 metrov
- druhý vo vzdialenosti 50 metrov.

Do digitálnej podoby sa tak na jedno prehratie celého zoznamu súborov zaznamenali zvuky z dvoch vzdialeností. Vzdialenosť mikrofónov od reproduktorov sme zisťovali pomocou GPS prijímaču. Po každom prehraní celého zoznamu sme posunuli obidva mikrofóny o 10 metrov bližšie k reproduktorom. Celé meranie sme opakovali až kým nebol prvý mikrofón 60 metrov a druhý 10 metrov od reproduktorov. Takto sme získali rôzne hladiny toho istého zvuku v rozmedzí 10 až 100 metrov.

Obr. 23 – Počiatočné rozmiestnenie mikrofónov

Cely prehraný zoznam zvukov sa zaznamenal do jedného dlhého zvukového súboru, ktorý obsahoval jednotlivé zvuky výstrelov zbraní. Pri každom meraní sme takto získali 2 zvukové súbory. Dohromady sme teda zaznamenali 10 súborov, ktoré reprezentovali zvuky z rôznych vzdialeností. Tieto veľké súbory samozrejme nebolo možné použiť ako trénovaciu množinu pre neurónovú sieť a preto ich bolo treba znovu rozdeliť na jednotlivé výstrely a roztriediť do správnych kategórií.

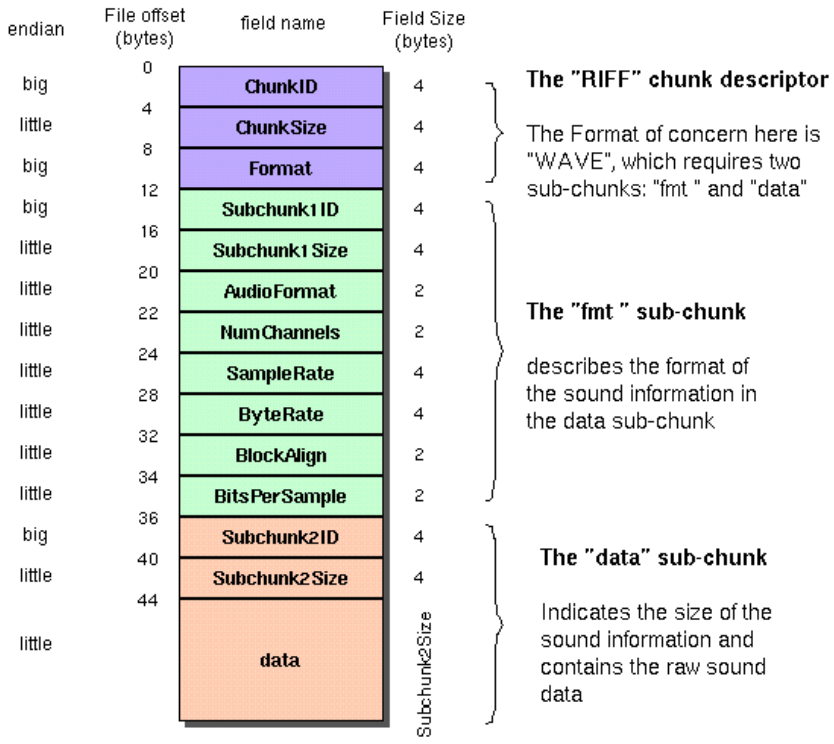
Zvuky som rozdelil pomocou špeciálneho programu pre spracovanie zvukových súborov a do kategórií som ich rozdelil podľa zoznamu z multimediálneho prehrávaču, ktorý som si pred meraním vytvoril. Takto som získal množinu približne 150 zvukov pre každú kategóriu. Takúto veľkú trénovaciu množinu už je možné použiť pre úspešné naučenie neurónovej siete na klasifikovanie výstrelov zo zbraní do správnych kategórií.

3.2 Extrahovanie zvukových charakteristík

Ako ďalší krok pri spracovávaní trénovacej množiny pre vstup do neurónovej siete musíme z vybraného zvuku extrahovať správne charakteristiky. Pomocou týchto charakteristík budeme schopný zvuk jednoznačne odlíšiť od ostatných a zaradiť ho tak do správnej kategórie. Získané zvukové súbory boli uložené do štandardného multimediálneho formátu WAV (WAVE), z ktorého som následne vo svojom programe vytvoril jednorozmerný vektor dát, ktoré už je možné priviesť na vstupné neuróny siete.

3.2.1 Wave zvukový formát

Zvukový formát WAVE vznikol zo základu špecifikácie Microsoft RIFF formátu pre ukladanie multimediálnych súborov.



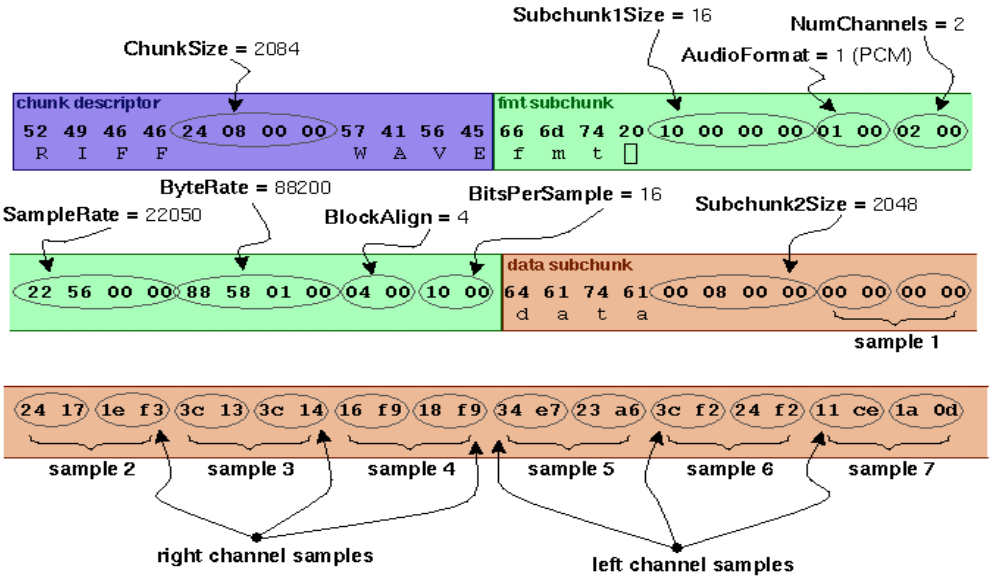
Obr. 24 – Grafická reprezentácia RIFF štandardu

Multimediálne aplikácie vyžadujú zaznamenávanie širokej škály dát, vrátane bitových máp, audio dát a video dát. RIFF poskytuje spôsob, ako uložiť všetky tieto typy dát. Súbory špecifikácie RIFF začínajú hlavičkou, ktorá určuje typ zvuku, metódu kompresie, vzorkovaciu frekvenciu a iné informácie potrebné pre rozpoznanie o aký súbor sa jedná. Po hlavičke nasleduje sekvencia dávok informácií reprezentujúce samotný multimediálny záznam.

Druh dát, ktorý RIFF súbor obsahuje, môžeme rozpoznať podľa prípony súboru. Príklady údajov, ktoré môžu byť uložené v súboroch RIFF sú:

- Audio vizuálne dáta (Audio/visual data .AVI)
- Zvukové dáta (Waveform data .WAV)
- MIDI informácie (Musical Instrument Digital Interface .RMI)
- Paleta farieb (Color palette .PAL)
- Multimediálny filmový záznam (Multimedia movie .RMN)
- Animovaný kurzor (Animated cursor .ANI)
- Zväzok iných RIFF súborov (bundle of other RIFF files .BND)

Súbor WAV je často RIFF súbor s jedným WAVE záznamom, ktorý sa skladá z dvoch častí dát tzv. chunks a to "fmt chunk" s uvedenými informáciami o formáte, a "data chunk" obsahujúci samotné zvukové dáta.

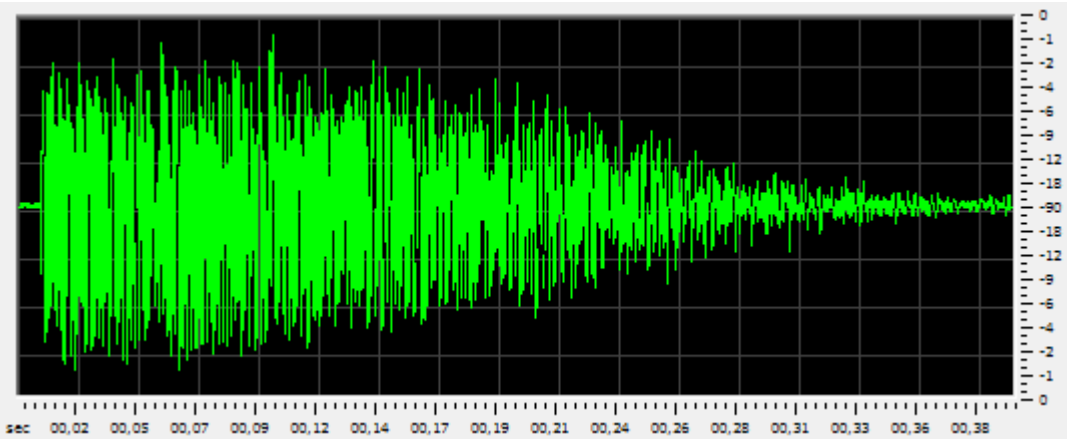


Obr. 25 – Význam jednotlivých bytov pre konkrétny WAVE súbor

3.2.2 Spektrálna výkonová hustota

V predchádzajúcej kapitole sme sa zaoberali získaním zvukových dát z WAVE multimediálneho súboru. V programovacom jazyku C# sa tieto dáta získavajú otvorením zvukového súboru pomocou objektu *FileStream* a nasledovne spracovaním pomocou objektu *BinaryReader*, ktorý rozdelí jednotlivé zvukové informácie do vektoru premenných typu *Int32[]*. V mojom programe sa o celý tento proces stará trieda *WavFileReader,* do ktorej som implementoval všetky potrebné procedúry pre úspešné spracovanie WAV súboru.

Takto vzniknutý vektor dát môže mať rôznu veľkosť v závislosti na dĺžke zvukového záznamu a vzorkovacej frekvencii súboru, v ktorom bol tento zvuk uložený. Môžeme tak povedať, že tento vektor je súbor výkonových hodnôt signálu v závislosti na časovej ose. Grafickú reprezentáciu môžeme vidieť na nasledujúcom obrázku:



Obr. 26 – Grafická reprezentácia zvukového súboru

O zobrazenie výkonovej závislosti zvukového signálu na čase sa stará *Windows Form* komponenta *WavSpectrum,* ktorú som sám naprogramoval za pomoci grafickej knižnice *System.Drawing* a objektu *Graphics.* Na ose x tohto grafického zobrazenia je znázornený čas a na ose y je výkon signálu v decibeloch. Výkon v decibeloch som vypočítal pomocou vzorcu:

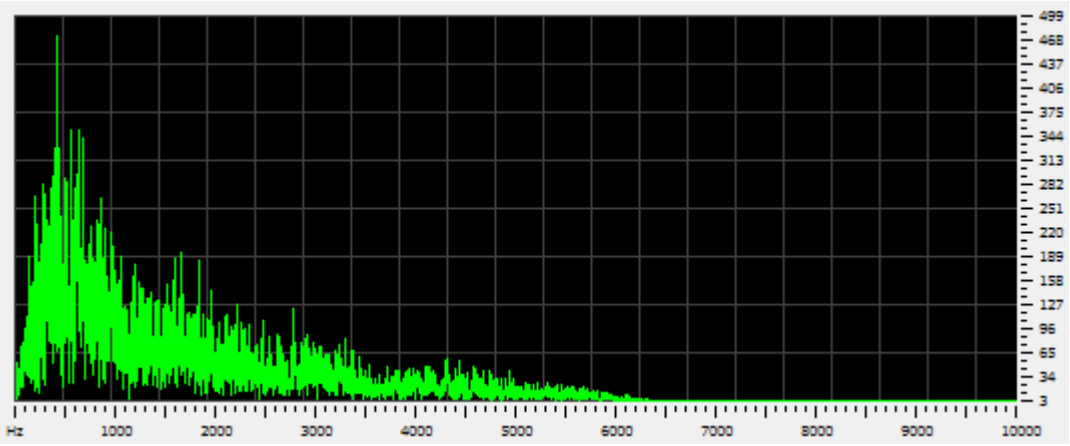
$$dB = 20\log\left(\frac{p}{p_0}\right)$$

Kde p je aktuálny výkon signálu a p_0 je maximálny možný výkon signálu, ktorý je pri WAV súbore 2^{15} čiže 32768 .

Keďže majú jednotlivé vektory rôzny počet členov a neurónová sieť má pevne daný počet vstupných neurónov je nutná úprava aby boli všetky vektory rovnako veľké. Tento problém by sa dal vyriešiť tak, že by sa vypočítal priemer výkonu signálu z rôzne dlhých časových úsekov tak aby mali všetky vektory rovnaký počet členov. Tento spôsob však nie je veľmi presný, pretože sa môže stať, že budeme mať vo výslednom vektore veľa nulových hodnôt tam kde zvuk začína alebo končí. Niektoré zvuky zbraní majú dokonca nulové hodnoty amplitúdy aj strede zvukového záznamu. Napríklad guľometné dávky.

Preto sa často pri rozpoznávaní zvukov neurónovou sieťou robí takzvaná analýza spektrálnej výkonovej hustoty čo je vlastne rozloženie výkonu v závislosti na frekvencii signálu. K tomuto účelu sa využíva diskrétna Fourierova transformácia DFT, o ktorej už väčšina bola napísaná v teoretickej časti tejto práce. Jedná sa o vyjadrenie časovo závislého signálu pomocou harmonických signálov, tj. funkcií sínus a kosínus. Slúži pre prevod signálov z časové oblasti do oblasti frekvenčnej. Ako už bolo v predchádzajúcej kapitole spomenuté, algoritmus DFT je pomerne jednoduchý a preto je časovo náročný na výpočet $O(n^2)$.

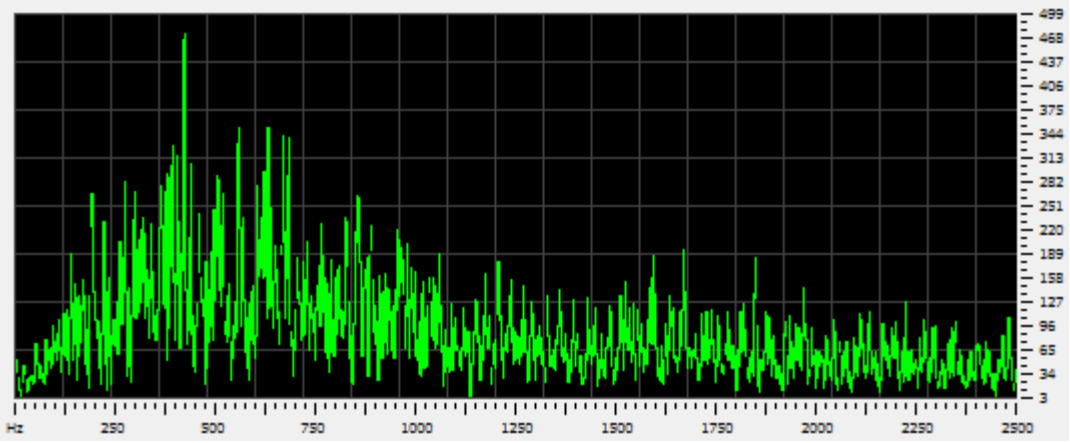
Z tohto dôvodu bola vypracovaná nová verzia tohto algoritmu označovaná rýchla Fourierova transformácia FFT (Fast Fourier Transform). Časová zložitosť tohto algoritmu je tento krát logaritmická $O(N \log_2 N)$. Pre uskutočnenie tejto transformácie na počítači je veľmi často využívaná knižnica FFTW (Fastest Fourier Transform in the West), ktorá je určená pre programovací jazyk C++. Kvôli tomu som sa aj ja rozhodol, že túto knižnicu použijem, keďže je voľne dostupná na internete pod licenciou open source. Najnovšia verzia tejto knižnice využíva podporu viacerých vlákien pre multijadrové procesory a je taká rýchla, že dôkaze vypočítať spektrálnu výkonovú hustotu v reálnom čaše.



Obr. 27 – Spektrálna výkonová hustota zvukového súboru

Na obrázku 27 je zobrazená grafická reprezentácia spektrálnej výkonovej hustoty, ktorá je zobrazovaná pomocou komponenty *FFTSpectrum*. Jedná sa o podobnú komponentu ako *WavSpectrum*, ktorá bola upravená tak aby zobrazovala závislosť výkonu na frekvencii. Vektor zvukových informácií po Fourierovej transformácii obsahuje hodnoty výkonu pre jednotlivé frekvencie v závislosti na vzorkovacej frekvencii, ktorou bol digitálny signál zaznamenaný. Ak je napríklad vzorkovacia frekvencia 8000 Hz môže zvukový záznam obsahovať len frekvencie od 0 do 4000 Hz.

Zvukove súbory, ktoré som použil pre trénovaciu množinu som všetky upravil tak aby mali vzorkovaciu frekvenciu 22050 Hz takže obsahujú frekvencie od 0 do 11025 Hz. To je dobre vidieť na grafickom zobrazení spektrálnej výkonovej hustoty na obrázku 27. Z obrázku je tak isto zrejmé, že najviac frekvencií sa nachádza v rozmedzí 0 až 5000 Hz. Toto som pozoroval u približne 80% všetkých zvukových záznamov v trénovacej množine. Preto nie je potrebné aby sa na vstup neurónovej siete predkladali hodnoty frekvencií väčšie ako 5000 Hz.



Obr. 28 – Frekvenčné spektrum obsahujúce frekvencie od 0 do 2500 Hz

V tomto prípade som zašiel ešte ďalej a rozhodol som sa, že výstupný vektor z Fourierovej transformácie zmenším na 5000 hodnôt čo predstavuje frekvencie od 0 do 2500 Hz, pretože v tomto rozmedzí sa nachádzajú najväčšie rozdiely a podobnosti jednotlivých výstrelov zo zbraní.

Môj program pr klasifikáciu zvukov ponúka možnosť nastaviť si počet vstupných neurónov od 1 do 250. Preto je vektor upravený Fourierovou transformáciou rozdelený na určitý počet úsekov, z ktorých je vypočítaný aritmetický priemer. Tento novo vzniknutý vektor, obsahujúci maximálne 250 hodnôt, už je použiteľný ako vstup do neurónovej siete.

4 APLIKÁCIA PRE KLASIFIKÁCIU ZDROJOV ZVUKOV

V tejto kapitole si podrobne predstavíme aplikáciu slúžiacu pre klasifikáciu zdrojov zvuku, ktorú som naprogramoval v rámci zadania tejto diplomovej práce. Ako programovací jazyk som sa rozhodol využiť jazyk C# (vyslovované anglicky ako C Sharp). Jedná sa o vysokoúrovňový objektovo orientovaný programovací jazyk vyvinutý firmou Microsoft zároveň s platformou .NET Framework. Tento jazyk bol neskôr schválený štandardizačnými komisiami ECMA a ISO. Microsoft založil C# na jazykoch C++ a Java a je teda nepriamym potomkom jazyka C, z ktorého čerpá syntax.

Programovací jazyk C# využíva pre beh aplikácie tzv. event-driven prístup (volanie udalostí). Udalosť je v podstate spôsob, ako objekty programu oznamujú, keď sa stane nejaká zaujímavá vec na stane užívateľa. Najznámejšie použitie pre udalosti je v grafickom užívateľskom rozhraní (GUI) pri objektoch, ktoré reprezentujú ovládacie prvky. Udalosti sú vyvolané, keď užívateľ urobí nejakú akciu s takýmto ovládacím prvkom, napríklad klikne na tlačidlo. Udalosť je výsledok tejto akcie. Existujú dva dôležité pojmy vo vzťahu k udalosti a to zdroj udalosti a prijímač udalosti. Objekt, ktorý vyvoláva udalosť sa nazýva zdroj udalosti a objekt, ktorý reaguje na udalosť sa nazýva prijímač udalosti.

Aplikácia pre rozpoznávanie zvukov sa skladá z dvoch hlavných ovládacích formulárov. Sú to formuláre *MainForm* a *NetworkSettings*. Formulár v Microsoft Visual Studiu je vlastne okno, ktoré sa otvorí po spustení aplikácie a obsahuje jednotlivé ovládacie prvky ako sú napríklad tlačidlá, menu, zoznamy alebo grafické prvky. Prvý zo spomenutých formulárov *MainForm* slúži pre ovládanie programu samotného, pre zobrazenie zvukových charakteristík v grafickej podobe a obsahuje knižnicu zvukov určených pre klasifikáciu. Formulár *NetworkSettings*, ako už názov napovedá, slúži pre nastavenie a učenie neurónovej siete, ktorá je jadrom programu.

Okrem dvoch hlavných formulárov aplikácia obsahuje viacero pomocných tried slúžiacich napríklad pre výpočet spektrálnej výkonovej hustoty alebo pre načítanie zvukového súboru do premennej. A v neposlednej rade jej súčasťou je neurónová sieť, ktorá sa skladá z viacerých spolupracujúcich tried. O všetkých častiach programu si povieme viac v nasledujúcich podkapitolách.

4.1 Hlavné prvky aplikácie

Ako už bolo spomenuté hlavnými prvkami aplikácie sú dva ovládacie formuláre, ktoré sprostredkúvajú komunikáciu medzi užívateľom a programom. Jedná sa o takzvané grafické užívateľské rozhranie (graphical user interface GUI), pomocou ktorého je možné aplikáciu jednoducho ovládať. V zdrojovom kóde aplikácie sa jedná o dva formuláre označené ako trieda *MainForm* a trieda *NetworkSettings*. Z programátorského hľadiska sú obidve tieto triedy potomkami triedy *Form*, ktorá obsahuje metódy pre pracú s oknami Windows.

Trieda *MainForm* slúži pre samotné ovládanie aplikácie pomocou, ktorého sú zobrazované charakteristiky zvukov, otvárané nové zvuky z knižnice a je na nej zobrazený výstup z neurónovej siete, zrozumiteľný pre človeka, vo forme názvu kategórie, do ktorej zvuk patrí. V ďalších kapitolách budeme o tejto triede hovoriť ako o formulári ovládania aplikácie.

Trieda *NetworkSettings* slúži pre nastavenie neurónovej siete. Pomocou nej je možné nastaviť počet vstupov, výstupov a skrytých vrstiev siete ako aj iné parametre siete ako je napríklad rýchlosť učenia a trénovacia množina. V ďalších kapitolách budeme o tejto triede hovoriť ako o formulári nastavenia neurónovej siete.

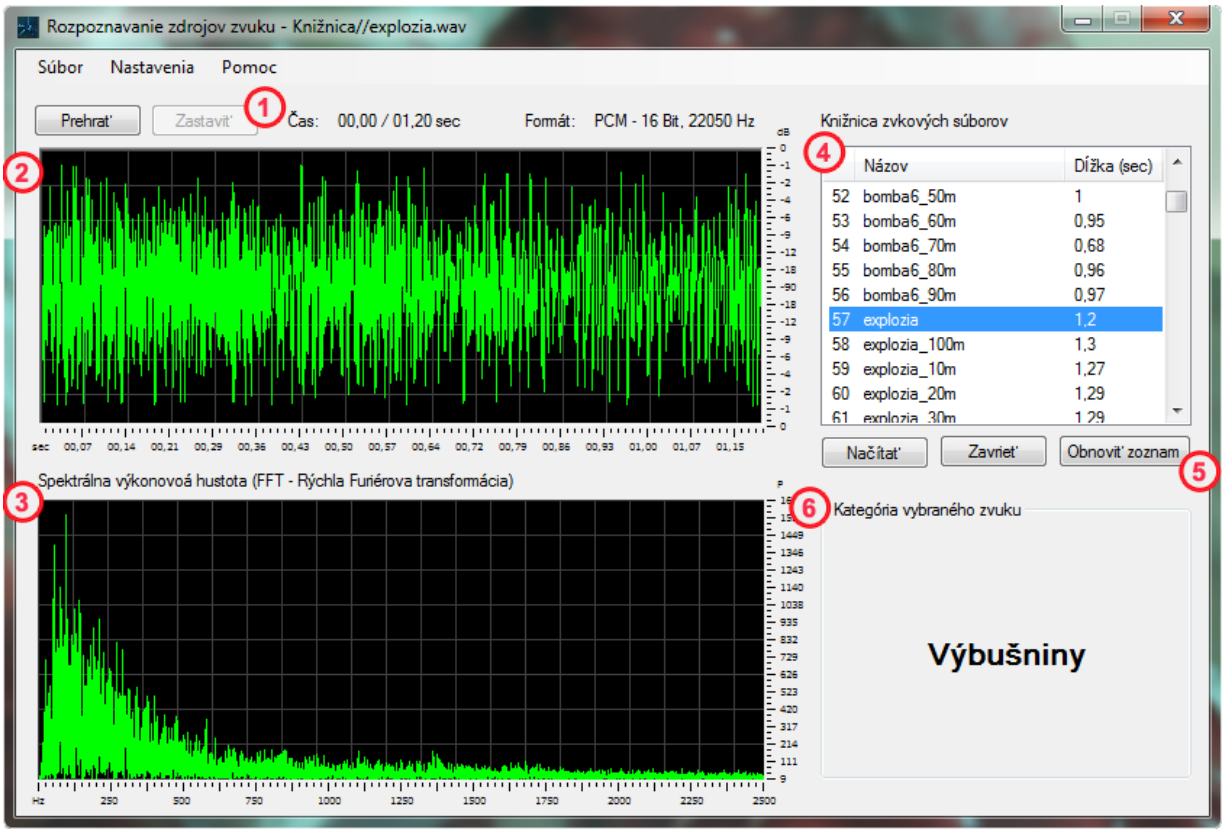
4.1.1 Formulár ovládania aplikácie

Práve tento formulár sa zobrazí hneď pri spustení aplikácie. Jedná sa o hlavný formulár, z ktorého je možné spustiť všetky ďalšie prvky aplikácie. Po zatvorení tohto formuláru sa ukončí aj celá aplikácia. Pred prvým zobrazením formuláru je vyvolaná udalosť *MainForm_Load()*. V rámci tejto udalosti sú vykonané procedúry slúžiace pre prvotné nastavenie aplikácie. Procedúry sú spúšťané v nasledujúcom poradí:

- získanie nastavení aplikácie prečítaním XML súboru
- inicializácia neurónovej siete
- načítanie zvukových súborov z pevného disku

Ovládacie a zobrazovacie prvky vo formulári ovládania aplikácie sú rozdelené do niekoľkých skupín. Tieto skupiny sú zobrazene na obrázku 29 a sú označené číslami v nasledujúcom poradí:

1. Ovládanie prehrávaču zvuku a informácie o súbore
2. Grafické zobrazenie zvukového signálu v závislosti na čaše
3. Grafické zobrazenie spektrálnej výkonovej hustoty
4. Knižnica zvukových súborov
5. Ovládacie prvky pre knižnicu zvukových súborov
6. Zobrazenie výstupu z neurónovej siete

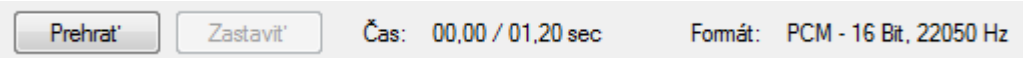


Obr. 29 – Formulár ovládania aplikácie

4.1.1.1 Ovládanie prehrávaču zvuku a informácie o súbore

Tieto ovládacie tlačidlá slúžia pre prehrávanie prave otvoreného zvukového súboru. Tlačidlo *Zastaviť* je neaktívne a aktivuje sa až pri prehrávaní zvuku. Čas prehrávania je zobrazovaný v sekundách a je možné vidieť aktuálny čas a celkovú dĺžku zvukového

súboru. V tejto skupine prvkov sú tiež zobrazené základne informácie o súbore ako je formát otvoreného WAV súboru a vzorkovacia frekvencia signálu. Táto skupina ovládacích a informačných prvkov je podrobnejšie zobrazená na obrázku 30.



Obr. 30 – Ovládanie prehrávaču zvuku a informácie o súbore

4.1.1.2 Grafické zobrazenie zvukového signálu v závislosti na čaše

Pomocou tohto zobrazovacieho prvku je užívateľovi zobrazovaný časový priebeh zvukového signálu. Tento priebeh je graficky zobrazený pomocou komponenty *WavSpectrum*. Po otvorení zvukového súboru je pomocou triedy *WavFileReader* tento súbor rozdelený na hlavičku a dátový záznam. Dáta sú prečítané a uložené do premennej typu *Int32[]*. Tento vektor dát je následne predaný komponente *WavSpectrum* tak, že je nastavená premenná *WavData*. Toto nastavenie premennej spustí funkciu *Invalidate()* čo má za následok, že je vyvolaný *OnPaint()* event a celá komponenta je prekreslená podľa zadaných vstupných dát.

4.1.1.3 Grafické zobrazenie spektrálnej výkonovej hustoty

Zobrazovací prvok slúžiaci pre grafické zobrazenie závislosti výkonu zvukového signálu na frekvencii. Táto komponenta má názov *FFTSpectrum* a pracuje na rovnakom princípe a komponenta *WavSpectrum*. Jediný rozdiel je vo vstupných dátach, ktoré sú pred predaním tejto komponente upravené Fourierovu transformáciou z časovej závislosti na frekvenčnú závislosť. Na ose *x* je zobrazená frekvencia signálu v Hz a na ose *y* je zobrazený výkon pre danú frekvenciu.

4.1.1.4 Knižnica zvukových súborov

V knižnici zvukových súborov sú všetky WAV súbory, ktoré sa nachádzajú na pevnom disku v adresári *sound,* ktorý je umiestnený v pracovnom adresári aplikácie. Tento zoznam je vytvorený pomocou komponenty *ListView*. Knižnica je rozdelená do troch stĺpcov, kde v prvom je poradové číslo súboru, v druhom názov súboru a v treťom celková dĺžka zvuku v sekundách. Zoznam zvukov sa načíta pri spúšťaní hlavného formuláru a pri veľkom počte súborov v adresári je toto načítanie časovo náročné.

4.1.1.5 Ovládacie prvky pre knižnicu zvukových súborov

Táto skupina ovládacích prvkov slúži pre ovládanie knižnice zvukových súborov. Tlačidlo *Načítať* otvorí vybraný súbor z knižnice. Je potrebné aby bol súbor najprv označený kliknutím na jeho názov v knižnici až potom je možné tento súbor načítať. Alternatíva pre toto tlačidlo je možnosť načítať súbor dvojitým kliknutím na jeho názov priamo v knižnici zvukov.

Tlačidlom *Zavrieť* je možné zatvoriť pravé načítaný súbor. Toto tlačidlo je pri spustení formuláru neaktívne a aktivuje sa až po načítaní zvuku do pamäte.

Tlačidlo *Obnoviť zoznam* slúži pre vynútenie znovu načítania celej knižnice z pevného disku. Táto možnosť slúži pre prípad ak by sa uskutočnili nejaké zmeny v adresári *sound* mimo aplikáciu (pridanie alebo vymazanie zvukových súborov). Tieto zmeny po kliknutí na toto tlačidlo zobrazia v knižnici súborov bez nutnosti reštartovať celu aplikáciu.

4.1.1.6 Zobrazenie výstupu z neurónovej siete

V tejto časti formuláru je zobrazená kategória, do ktorej je zvuk zaradený. Je to vlastne výsledok dotazu na neurónovú sieť. Pri načítaní zvuku sa okrem ostatných funkcií vykoná aj dotaz na neurónovú sieť pomocou funkcie *Test()*. Táto funkcia vykoná aktivačnú fázu neurónovej siete a skontroluje výstupné neuróny. Vráti poradie neurónu, ktorý má najväčšiu hodnotu a to je neurón, ktorý identifikuje kategóriu zvuku.

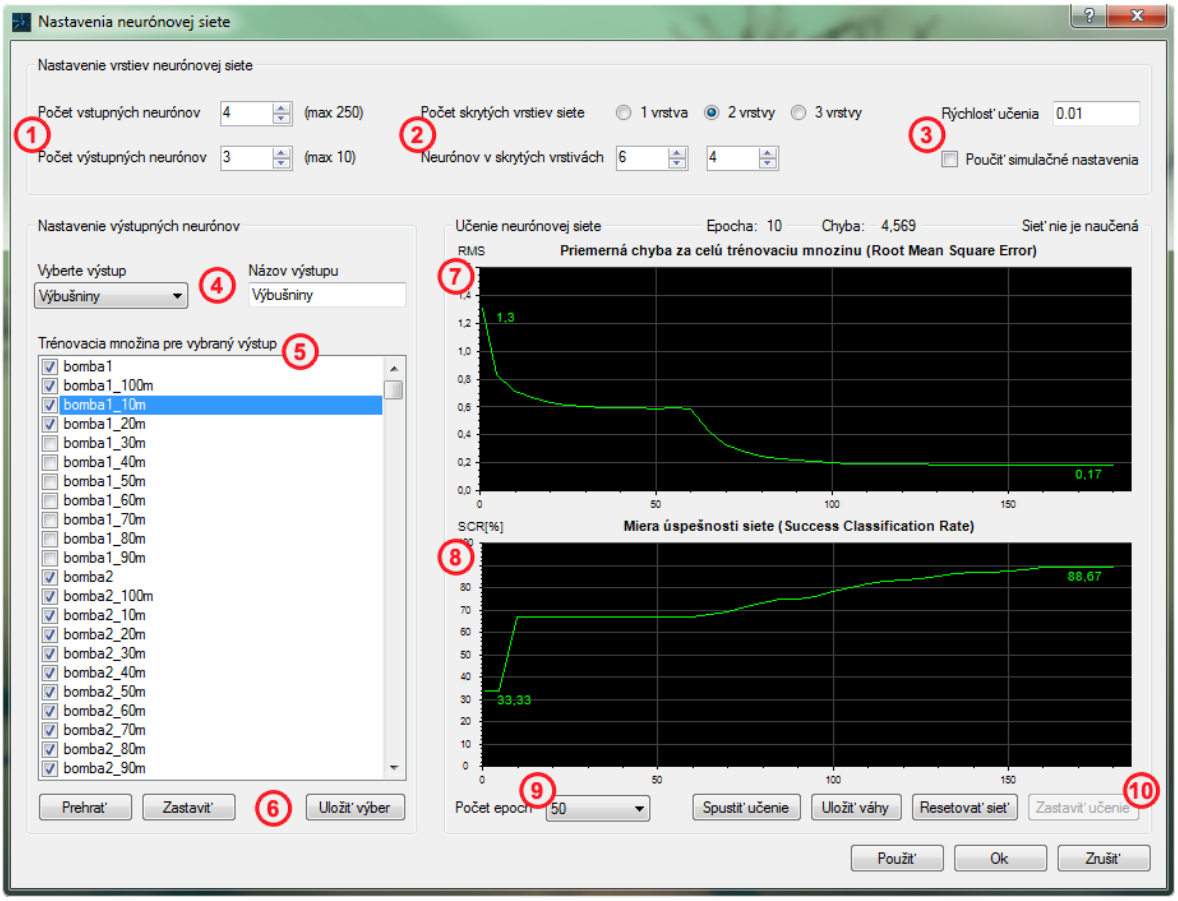
4.1.2 Formulár nastavenia neurónovej siete

Tento formulár slúžiaci pre nastavenie a učenie neurónovej siete je možné spustiť z formuláru ovládania aplikácie kliknutím v menu na položku *Nastavenia » Neurónová sieť*. Pri prvom zobrazení formuláru je vyvolaná udalosť *NetworkSettings_Load()*, ktorá obsahuje procedúry pre získanie nastavení formuláru a informácie o zvukových súboroch. Procedúry sú spúšťané v nasledovnom poradí:

- uloženie ovládacích prvkov *CheckedListBox*, *TextBox* a *Panel* do premenných
- spustenie procesu bežiaceho na pozadí pre načítanie zvukových dát uložených v adresári *training* na pevnom disku. Operácia je časovo náročná, pretože pri načítaní zvukov je pre každý súbor vykonaná aj Fourierova transformácia.
- vytvorenie grafických objektov pre zobrazovanie chyby a úspešnosti siete

Ovládacie a zobrazovacie prvky vo formulári rozdelené do skupín sú zobrazené na obrázku 31 a sú označené číslami v nasledujúcom poradí:

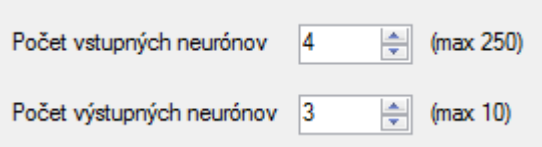
1. Nastavenie vstupných a výstupných neurónov siete
2. Nastavenie skrytých vrstiev siete a ich neurónov
3. Nastavenie ďalších parametrov neurónovej siete
4. Výber výstupnej kategórie a jej nastavenie
5. Trénovacia množina pre vybranú kategóriu
6. Ovládacie prvky pre trénovaciu množinu
7. Grafické zobrazenie priebehu priemernej chyby učenia siete
8. Grafické zobrazenie úspešnosti učenia neurónovej siete
9. Výber počtu epoch pre jeden cyklus učenia
10. Ovládacie tlačidlá učenia neurónovej siete



Obr. 31 – Formulár nastavenia neurónovej siete

4.1.2.1 Nastavenie vstupných a výstupných neurónov siete

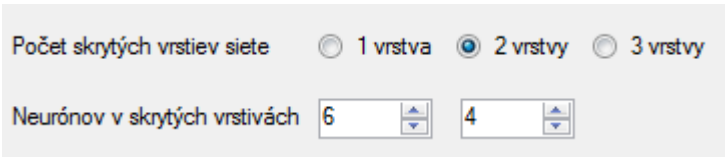
Nastavenie neurónovej siete prebieha pomocou ovládacích prvkov nazývaných *NumericUpDown*. Pomocou nich je možné nastaviť počet vstupných neurónov od 1 do 250 a počet výstupných neurónov v sieti od 1 do 10. Maximálny počet 250 vstupných neurónov som zvolil kvôli tomu, že viac neurónov by bolo zbytočných pre rozpoznanie zvuku a učenie neurónovej siete by príliš zaťažovalo procesor a bolo by tak časovo náročné (radovo v hodinách). Počet výstupných neurónov priamo určuje počet kategórií, do ktorých môže byť zvuk zaradený. Každý výstupný neurón reprezentuje jednu kategóriu.



Obr. 32 – Nastavenie vstupných a výstupných neurónov

4.1.2.2 Nastavenie skrytých vrstiev siete a ich neurónov

Počet skrytých vrstiev neurónovej siete je možné nastaviť na jednu, dve alebo tri skryté vrstvy. Toto nastavenie sa mení pomocou ovládacieho prvku nazývaného *RadioButton*, kde môže byt označená vždy len jedna možnosť. Pri zmene počtu skrytých vrstiev sa zobrazujú alebo skrývajú ďalšie *NumericUpDown* polia, ktorými nastavujeme počet neurónov v danej skrytej vrstve. Napríklad ak máme označenú len jednu skrytú vrstvu zobrazí sa pod ňou len jeden ovládací prvok pre nastavenie počtu neurónov ak však mame označený počet vrstiev tri, tak sú pod nimi zobrazené tri ovládacie prvky pre nastavenie počtu neurónov. Prvky môžeme podrobnejšie vidieť na obrázku 33.



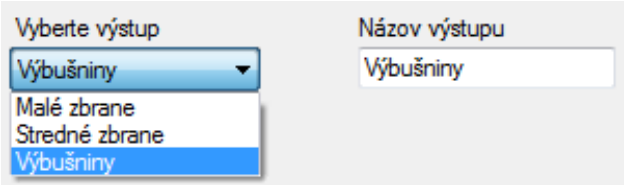
Obr. 33 – Nastavenie skrytých vrstiev neurónovej siete

4.1.2.3 Nastavenie ďalších parametrov neurónovej siete

Tu sa nastavuje parameter rýchlosť učenia neurónu definovaná ako *LearningRate*, ktorá určuje ako rýchlo a presne sa bude neurónová sieť učiť. Ďalej sa tu nachádza aj zaškrtávacie políčko, ktorým je možné určiť či má sieť pre učenie použiť trénovaciu množinu zvukov alebo testovaciu množinu používanú pri testovaní siete.

4.1.2.4 Výber výstupnej kategórie a jej nastavenie

Tieto ovládacie prvky slúžia pre výber a nastavenie výstupnej kategórie neurónovej siete. Pri zmene počtu výstupov neurónovej siete sa zmení aj počet kategórii, ktoré môžeme upravovať. Výber kategórie sa uskutočňuje pomocou prvku *ComboBox* a po vybraní určitej kategórie je možné nastaviť názov kategórie a zmeniť množinu zvukov, ktoré budú reprezentovať trénovaciu množinu pre túto kategóriu. Názov tohto výstupu bude zobrazený ako výstup z neurónovej siete v hlavnom formulári ovládania aplikácie.



Obr. 34 – Výber výstupnej kategórie

4.1.2.5 Trénovacia množina pre vybranú kategóriu

Je zoznam všetkých zvukových súborov WAV, ktoré sú umiestnené v adresári *training* na pevnom disku. Tento ovládací panel sa nazýva *CheckedListBox* a umožňuje výber viacerých záznamov. Vedľa názvu zvukového súboru je zaškrtávacie políčko. Po označení tohto políčka, čo sa robí dvojitým kliknutím na meno súboru, je tento zvuk zaradený do trénovacej množiny pre vybraný výstup.

Takto je možné z celej databáze súborov vybrať len tie zvuky, ktoré chceme aby boli zaradené do vybranej kategórie. Napríklad ak vyberieme výstup *Výbušniny* v tomto poli označíme len zvuky reprezentujúce výbušniny a takto ďalej postupujeme pri výbere vhodných zvukov pre všetky výstupné kategórie. Vytvoríme tak trénovaciu množinu, ktorú je možné použiť pre učenie neurónovej siete.

4.1.2.6 Ovládacie prvky pre trénovaciu množinu

Tlačidlá slúžia pre ovládanie prehrávaču zvukov. Po vybraní súboru z trénovacej množiny je možné zvuk prehrať kliknutím na tlačidlo *Prehrať*. Tlačidlo *Uložiť výber* zaznamená zvolený výber zvukov z trénovacej množiny do súboru aby sa zvuky nemuseli označovať znovu pri každom reštartovaní formuláru nastavenia neurónovej siete.

4.1.2.7 Grafické zobrazenie priebehu priemernej chyby učenia siete

Pomocou tohto zobrazovacieho prvku je graficky zobrazený vývoj priemernej chyby za celú trénovaciu množinu počas učenia neurónovej siete. Priemerná chyba za trénovaciu množinu, anglicky root mean square error (RMS), je veličina, ktorá sa používa pre sledovanie priebehu učenia neurónovej siete. RMS sa vypočíta podľa vzorcu:

$$RMS = \sqrt{\frac{E}{n}}$$

kde E je globálna chyba pre jednu epochu učenia a n je počet všetkých prvkov v trénovacej množine. Pre zobrazenie priebehu v reálnom čaše som použil knižnicu ZedGraph, ktorá je voľne dostupná z internetu. Na ose x je zobrazený počet epoch a na ose y je priemerná chyba za trénovaciu množinu. Táto chyba sa prepočítava každých 5 epoch učenia siete.

4.1.2.8 Grafické zobrazenie úspešnosti učenia neurónovej siete

Tento zobrazovací prvok reprezentuje grafický priebeh úspešnosti učenia neurónovej siete. Počas učenia je každých 5 epoch sieť testovaná koľko prvkov dokáže správne zaradiť do kategórie a z toho je vypočítané percentuálne vyjadrenie úspešnosti siete. Veličina sa nazýva miera úspešnosti neurónovej siete, anglicky success clasification rate (SCR), je vyjadrená v percentách a vypočíta sa podľa vzorcu:

$$SCR = \frac{S}{n}.100$$

kde S je počet správne zaradených prvkov množiny a n je počet všetkých prvkov v trénovacej množine.

4.1.2.9 Výber počtu epoch pre jeden cyklus učenia

Pomocou tohto ovládacieho prvku je možné určiť, koľko epoch má byť vykonaných za jeden učiaci cyklus. Z prvku *ComboBox* je možné označením vybrať hodnoty 1, 10, 50, 100, 200, 500, 1000. Po spustení učenia sa vykoná trénovací cyklus zo zadaným počtom epoch a potom sa učenie siete automaticky zastaví. Zastaviť učenie je možné aj pred uplynutím zadaného počtu epoch pomocou tlačidla *Zastaviť učenie*.

4.1.2.10 Ovládacie tlačidlá učenia neurónovej siete

Táto skupina ovládacích tlačidiel slúži pre ovládanie učenia neurónovej siete. Tlačidlo *Spustiť učenie* začne učiaci cyklus, ktorý je možné zastaviť kliknutím na tlačidlo *Zastaviť učenie* alebo skončí automaticky po uplynutí počtu epoch, ktoré sú nastavené prvkom *Počet epoch*.

Pri učení neurónovej siete sa môže stať, že sa sieť dostane do takzvaného lokálneho minima a už sa nedokáže viac naučiť. Pre tento prípad slúži tlačidlo *Resetovať sieť*. Pomocou tohto tlačidla sa sieť znovu načíta s náhodnými váhami a učenie sa tak môže spustiť znovu.

Po ukončení učiaceho cyklu môžeme podľa grafu vidieť ako je sieť naučená na daný problém. Ak sme s týmto výsledkom spokojný môžeme uložiť na pevný disk váhy neurónov tlačidlom *Uložiť váhy*. Pri ďalšom spustení programu budú tieto váhy načítané zo súboru a sieť nebude potrebné zvonu učiť na danú problematiku.

Obr. 35 – Ovládanie učenia neurónovej siete

4.2 Jadro aplikácie - neurónová sieť

Súčasťou programu je viacvrstvová neurónová sieť využívajúca učiaci algoritmus backpropagation, ktorá je považovaná za jadro aplikácie. Princíp funkcie algoritmu backpropagation môžeme popísať v niekoľkých ľahko pochopiteľných krokoch:

1. Inicializácia siete s nastavením malých náhodných váh neurónov.
2. Predloženie vstupného vektoru na vstupnú vrstvu siete.
3. Posúvanie vstupných dát dopredne celou sieťou aby sa vypočítala aktivačná hodnota všetkých neurónov.
4. Výpočet rozdielu medzi požadovaným výstupom a aktivačnou hodnotou výstupných neurónov aby sa zistila aktivačná chyba neurónovej siete.
5. Zmena nastavenia váh výstupných neurónov aby sa znížila aktivačná chyba pre tento vstupný vektor.

6. Šírenie chybovej hodnoty späť na každý skrytý neurón tak, že hodnota je úmerná ich prínosu k aktivačnej chybe siete.
7. Nastavenie váh pre každý skrytý neurón tak aby sa znížil jeho prínos k aktivačnej chybe pre dané vstupné dáta.
8. Opakovanie krokov 2 až 7 pre každý vstupný vektor z trénovacej množiny.
9. Opakovanie kroku 8 až kým nie je sieť dostatočne naučená.

Je dôležité si uvedomiť, že váhy siete sa musia upraviť len mierne pre aktuálny vstupný vektor pred prechodom na ďalší vektor v množine. Ak by sme sieť nechali dokonale opraviť chybu aktuálneho vektoru pred prechodom na ďalší, sieť by nikdy nedokázala nájsť všeobecné riešenie pre celú trénovaciu množinu.

Zdrojový kód neurónovej siete je rozdelený so piatich tried, ktoré obsahujú všetky potrebné metódy pre spustenie, naučenie aj testovanie tejto siete. Jedná sa o triedy *Network*, *Layer*, *Pattern*, *Neuron* a *Weight*.

4.2.1 Trieda Network

Hlavná trieda obsahujúca metódy pre pracú z neurónovou sieťou. Konštruktor tejto triedy prijíma parameter *dimensions*, ktorý definuje jednotlivé vrstvy siete. Premenná *dimensions* je vlastne pole integer hodnôt a definuje počet neurónov v jednotlivých vrstvách siete. Napríklad pole {5, 3, 3, 2} reprezentuje sieť, ktorá má 5 vstupov, 2 skryté vrstvy a 2 výstupné neuróny. Prvá skrytá vrstva má 4 neuróny a druhá 3 neuróny. Toto pole sa získava z ovládacích prvkov formuláru nastavenia neurónovej siete.

Trieda *Network* obsahuje metódu *Train()*, ktorá vykoná jednu trénovaciu epochu. Volaním tejto metódy sa vykoná aktivačná a adaptačná fáza pre každý prvok z trénovacej množiny a vráti sa globálna chyba za jednu epochu. Opakované volanie tejto metódy je vlastné učenie neurónovej siete.

Metóda *Test()* je podobná ako metóda *Train()*, ale obsahuje len aktivačnú fázu neurónovej siete a vráti poradie výstupného neurónu, ktorý ma najväčšiu hodnotu. Ako ďalšie obsahuje trieda metódy *Activate()* a *AdjustWeights()*. Prvá aktivuje každý neurón v neurónovej sieti a druhá zmení nastavenia váh každého neurónu v sieti.

4.2.2 Trieda Layer

Táto trieda slúži ako zoznam všetkých neurónov v jednej vrstve. Jedná sa o jednoduchý zoznam typu *List<Neuron>*, ktorý obsahuje jednotlivé neuróny. Trieda môže byt vytvorená dvomi spôsobmi zadaním rôzneho počtu vstupných parametrov. Zadaním len počtu neurónov ak sa jedná o vstupnú vrstvu a zadaním počtu neurónov, poradia vrstvy a náhodnej váhy neurónu ak sa jedná o skryté a výstupnú vrstvu siete.

4.2.3 Trieda Pattern

Trieda slúži pre spracovanie dát do podoby, ktorej rozumie neurónová sieť. Jedná sa o vektor vstupov a im priradených výstupov. Konštruktor triedy rozdelí zadané dáta do premenných *inputs* a *outputs.* Sieť ďalej pracuje z danými premennými a dotazuje sa na *Pattern.Inputs* a *Pattern.Outputs* pre získanie dát z trénovacej množiny.

4.2.4 Trieda Neuron

Táto trieda reprezentuje neurón a obsahuje metódy pre aktivačnú a adaptačnú fázu neurónu. Pri vytváraní novej inštancie neurónu je zadaná vrstva, v ktorej bude neurón umiestnený. Pre každý neurón vo vrstve je vytvorená nová synaptická váha, ktorej hodnota je získaná zo súboru ak je sieť naučená alebo je určená náhodne.

Pri aktivačnej fáze je spustená funkcia *Activate(),* počas ktorej sa vypočíta suma vstupných hodnôt neurónu *Neuron.input*. Pre každú synaptickú váhu neurónového spojenia je vynásobená hodnota váhy spojenia z výstupom neurónu odpovedajúceho spojenia.

Pri adaptačnej fáze je vykonaná funkcia *AdjustWeights()*, ktorá zmení hodnoty všetkých neurónových váh, ktoré odpovedajú danému neurónu. Ako prenosová funkcia neurónu je použitá derivácia sigmoidálnej funkcie.

4.2.5 Trieda Weight

Obsahuje tri premenné slúžiace pre definíciu synaptickej váhy neurónu. *InputId* je identifikačné číslo neurónu, premenná *Input* je ukazovateľ na neurón a *Value* je váhová hodnota daného spojenia neurónu.

4.3 Testovanie neurónovej siete

Testovanie neurónovej siete prebiehalo za použitia špeciálnej testovacej množiny, získanej z internetu zo stránky Machine Learning Repository, ktorá je určená pre ukladanie testovacích dát pre učenie umelých systémov. Použil som databázu, ktorá sa často vyskytuje v literatúre o systémovom rozpoznávaní. Súbor údajov obsahuje 3 triedy o 50 prvkoch pre každú triedu, kde každá trieda odkazuje na typ kosatcových rastlín. Jedna trieda je lineárne oddeliteľná od ostatných a ďalšie dve nie sú lineárne oddeliteľné jedna od druhej. Atribúty rastlín v trénovacej množine sú uložené v danom poradí:

1. dĺžka kališných lístkov v cm
2. šírka kališných lístkov v cm
3. dĺžka okvetných lístkov v cm
4. šírka okvetných lístkov v cm
5. trieda kosatcovitej rastliny - Iris Setosa, Iris Versicolour, Iris Virginica

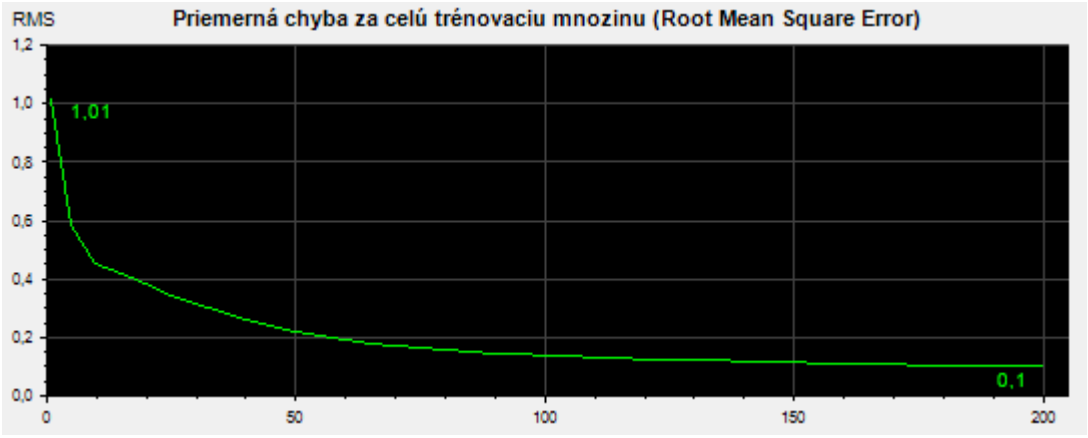
Trénovacia množina sa skladá z vektorov o piatich členoch, kde prvé 4 členy sú atribúty rastliny a piaty definuje triedu, do ktorej rastlina patrí. Preto som použil neurónovú sieť, ktorá má 4 vstupné neuróny a 3 výstupné. Zvolil som 2 skryté vrstvy kde prvá mala 6 neurónov a druhá 4 neuróny.

Testovanie prebiehalo v troch testovacích cykloch, pri ktorých bolo vždy vykonaných 200 učiacich epoch neurónovej siete. Výsledky sú zobrazené v tabuľke a graficky pomocou zobrazovacích prvkov aplikácie.

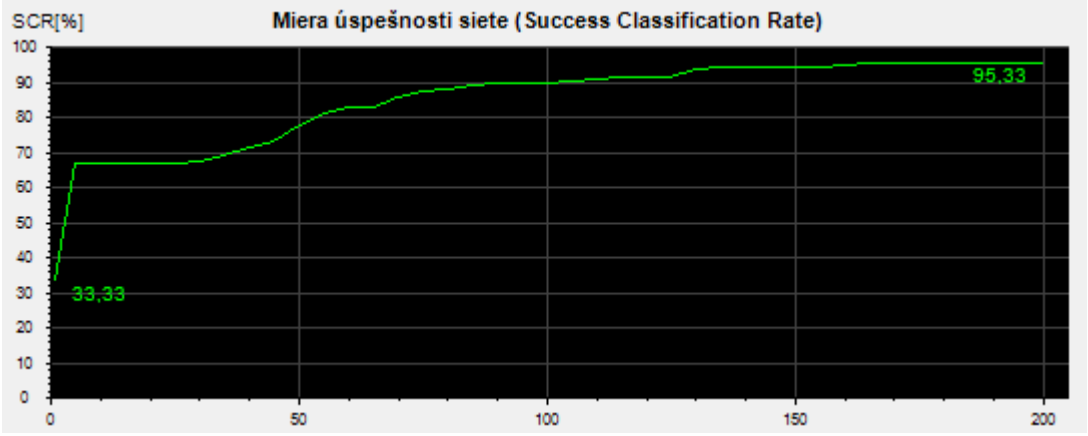
4.3.1 Testovací cyklus 1

Epocha	Globálna chyba	RMS	SCR
1	154,37	1,014	33,33
5	50,24	0,579	66,67
10	30,45	0,451	66,67
50	6,97	0,216	77,33
100	2,73	0,135	89,33
150	1,89	0,112	94,1
200	1,48	0,099	95,33

Tabuľka 1 – Testovací cyklus neurónovej siete 1



Obr. 36 – Priemerná chyba za trénovaciu množinu pri testovacom cykle 1



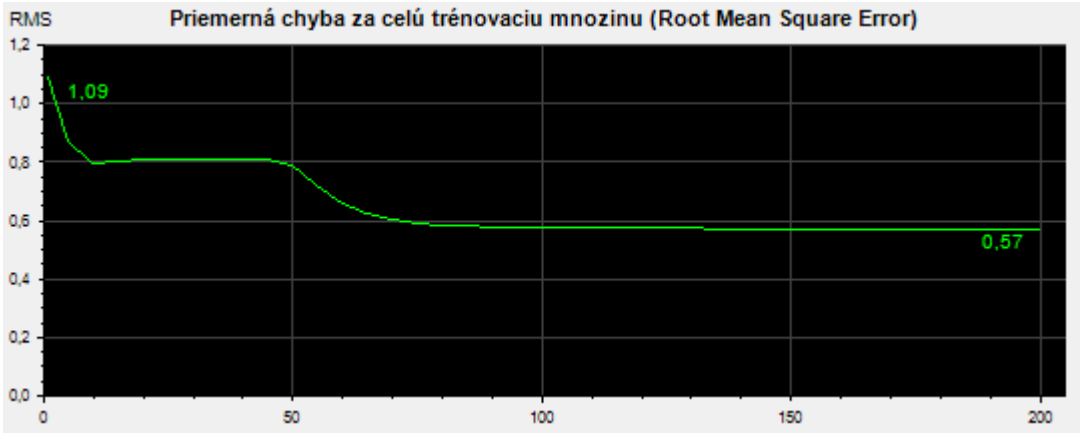
Obr. 37 – Miera úspešnosti siete pri testovacom cykle 1

Z tabuľky a grafov je zrejmé, že sa sieť dokázala dostatočne dobre naučiť na klasifikáciu rastlín do správnych kategórií. Priemerná chyba za trénovaciu množinu (RMS) exponenciálne klesala až dosiahla hodnoty menšej ako 0,1 a sieť tak môžeme považovať za naučenú na danú problematiku. Miera úspešnosti neurónovej siete (SCR) dosiahla v poslednej epoche 95,33% čo je veľmi dobra úspešnosť pri klasifikácii.

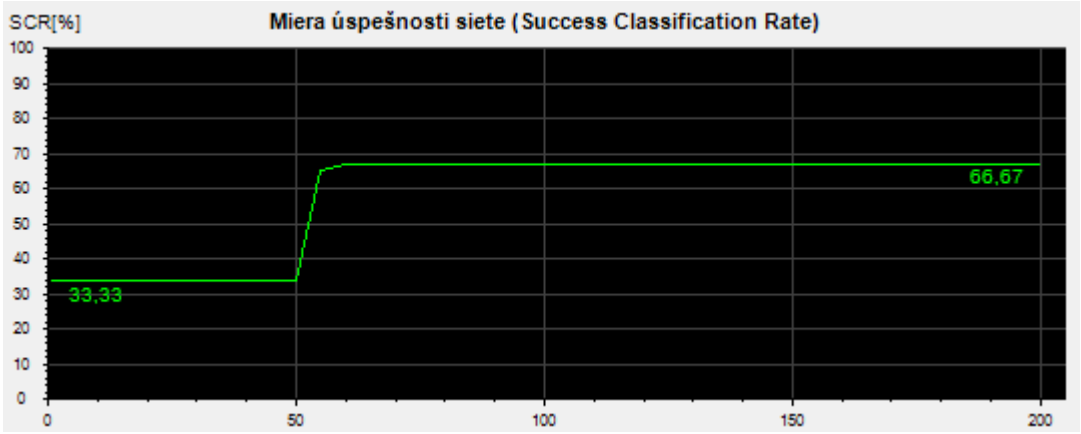
4.3.2 Testovací cyklus 2

Epocha	Globálna chyba	RMS	SCR [%]
1	176,96	1,086	33,34
5	112,94	0,868	33,34
10	94,95	0,796	33,34
50	92,54	0,785	33,34
100	49,22	0,573	66,67
150	48,57	0,569	66,67
200	48,35	0,568	66,67

Tabuľka 2 – Testovací cyklus neurónovej siete 2



Obr. 38 – Priemerná chyba za trénovaciu množinu pri testovacom cykle 3



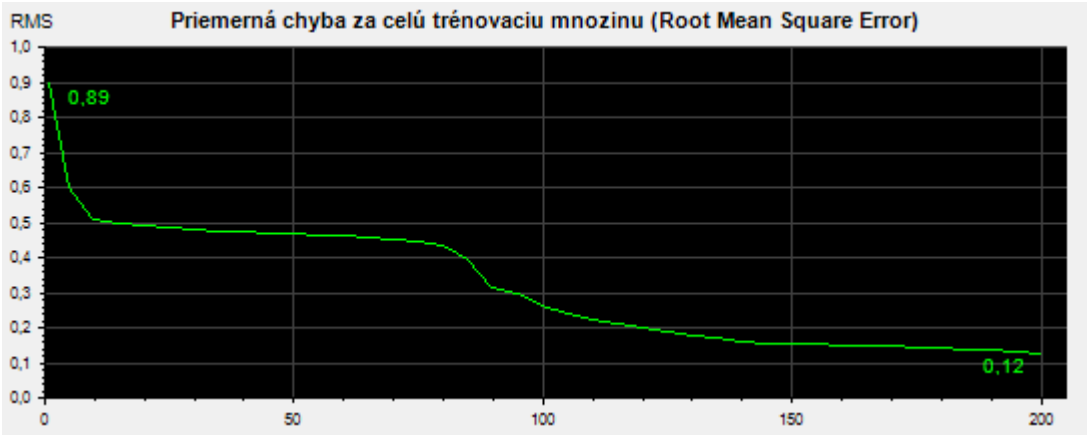
Obr. 39 – Miera úspešnosti siete pri testovacom cykle 3

Pri tomto testovacom cykle sa neurónovú sieť nepodarilo naučiť za 200 epoch. Ako môžeme vidieť z grafov sieť sa dostala do lokálneho minima, kde priemerná chyba prestala klesať. Chyba prudko klesala počas prvých 100 epoch kde sieť odlíšila prvé dve triedy, ktoré sú lineárne separabilné. Ďalších 100 epoch chyba neklesala a úspešnosť siete sa nezvýšila, takže sieť nedokázala oddeliť posledné dve lineárne neseparabilné triedy. Pre dokončenie testovania som sieť reštartoval z náhodné nastavenými váhami.

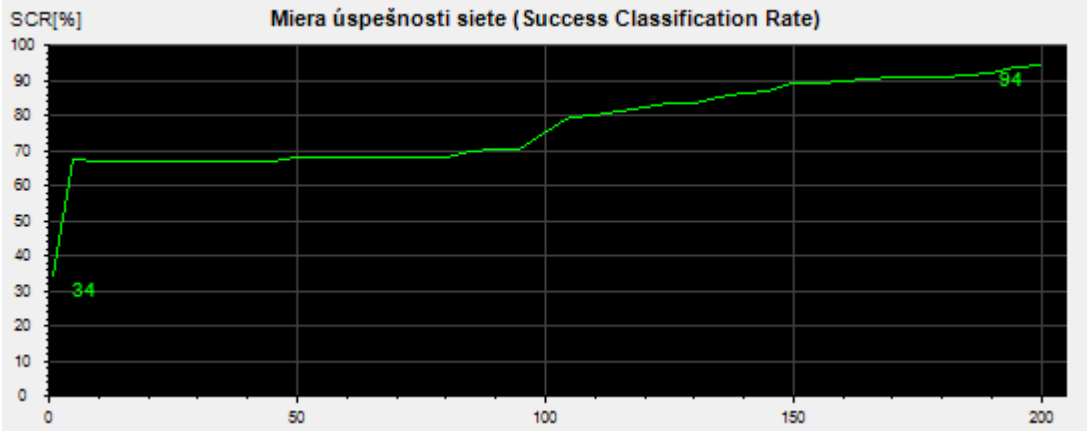
4.3.3 Testovací cyklus 3

Epocha	Globálna chyba	RMS	SCR [%]
1	120	0,894	34
5	53,68	0,598	67,34
10	38,4	0,506	66,67
50	32,39	0,465	68
100	10,12	0,26	74,67
150	3,43	0,151	88,67
200	2,31	0,124	94

Tabuľka 3 – Testovací cyklus neurónovej siete 3



Obr. 40 – Priemerná chyba za trénovaciu množinu pri testovacom cykle 3



Obr. 41 – Miera úspešnosti siete pri testovacom cykle 3

Pri poslednom testovacom cykle sa sieť opäť dokázala naučiť na klasifikáciu daného problému. Počiatočné prudké klesanie priemernej chyby je z dôvodu riešenia lineárne separabilného problému a nasledovné pozvoľné klesanie chyby a rast úspešnosti siete ukazuje, že sieť dokázala vyriešiť aj lineárne nesaparabilný problém.

Z testovania môžem usúdiť, že tento model siete je správne naprogramovaný a je možné ho použiť aj pre klasifikáciu zvukových signálov.

4.4 Učenie neurónovej siete

Učenie neurónovej siete na problematiku klasifikácie zdrojov zvuku prebiehalo v troch učiacich cykloch. Pre každý cyklus bolo vykonaných 200 učiacich epoch. Použil som štvorvrstvovú neurónovú sieť s dvomi skrytými vrstvami. Pre jednotlivé učiace cykly som použil rozdielne nastavenia počtu neurónov vo vstupnej a skrytých vrstvách. Počet neurónov v jednotlivých vrstvách som vypočítal podľa univerzálneho vzorcu pre výpočet topológie siete:

$$N_{s1} = N_{vy} * \left(\sqrt[3]{\frac{N_{vs}}{N_{vy}}} \right)^2 \qquad N_{s2} = N_{vy} * \left(\sqrt[3]{\frac{N_{vs}}{N_{vy}}} \right) \tag{8}$$

kde N_{s1} je počet neurónov v prvej skrytej vrstve, N_{s2} je počet neurónov v druhej skrytej vrstve N_{vs} je počet vstupných neurónov a N_{vy} je počet výstupných neurónov. Počas celého učenia neurónovej siete sa nebude meniť počet výstupných neurónov. Celá trénovacia množina je rozdelená do 3 kategórií a tak bude mať sieť 3 výstupné neuróny.

V prvom učiacom cykle som zvolil 50 vstupných neurónov, na ktoré sa privedie vstupný vektor zvukového signálu. Pomocou vzorcov (8) som vypočítal pre prvú skrytú vrstvu 20 neurónov a pre druhú skrytú vrstvu 8 neurónov. Neurónová sieť tak bude vytvorená pomocou poľa *dimensions* s hodnotami {50, 20, 8, 3}.

V druhom cykle som zvolil 25 vstupných neurónov a zo vzorca som vypočítal 12 neurónov pre prvú skrytú vrstvu a 6 neurónov pre druhú skrytú vrstvu. Sieť tak bude mať dimenzie {25, 12, 6, 3}.

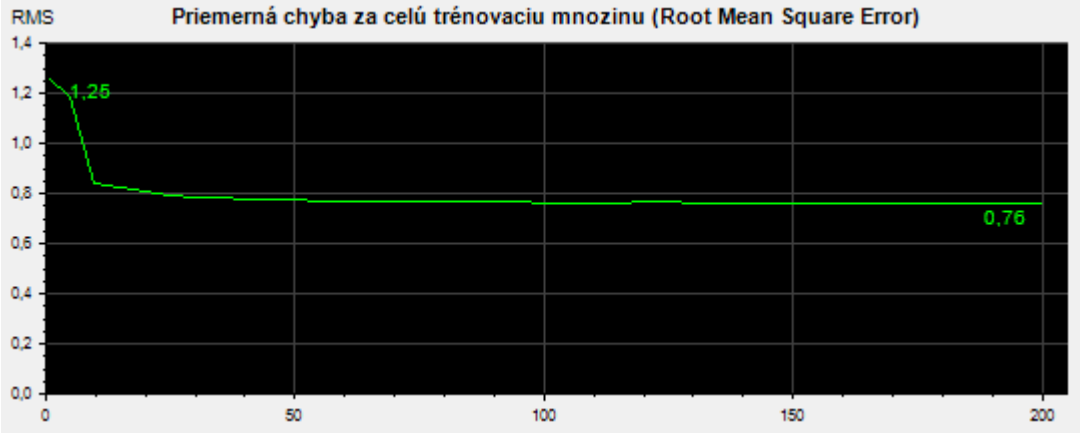
Pre posledný učiaci cyklus použijem neurónovú sieť, ktorá bude mať 10 vstupných neurónov, 7 neurónov v prvej skrytej vrstve, 4 neuróny v druhej skrytej vrstve a sieť tak inicializujem s dimenziami {10, 7, 4, 3}.

Trénovacia množina pozostáva zo zvukov výstrelov zo zbraní, ktoré som si predtým upravil do správnej podoby ako bolo spomenuté v kapitole 3 - Databáza zvukov. Pre každú kategóriu je v trénovacej množine určených približné 80 zvukových záznamov. Snažil som sa vybrať tie zvuky z daných kategórií, ktoré mali čo najviac podobné charakteristiky v danej kategórii a čo najviac sa odlišovali od ostatných kategórií. Výsledky sú zobrazené v tabuľke a graficky pomocou zobrazovacích prvkov aplikácie.

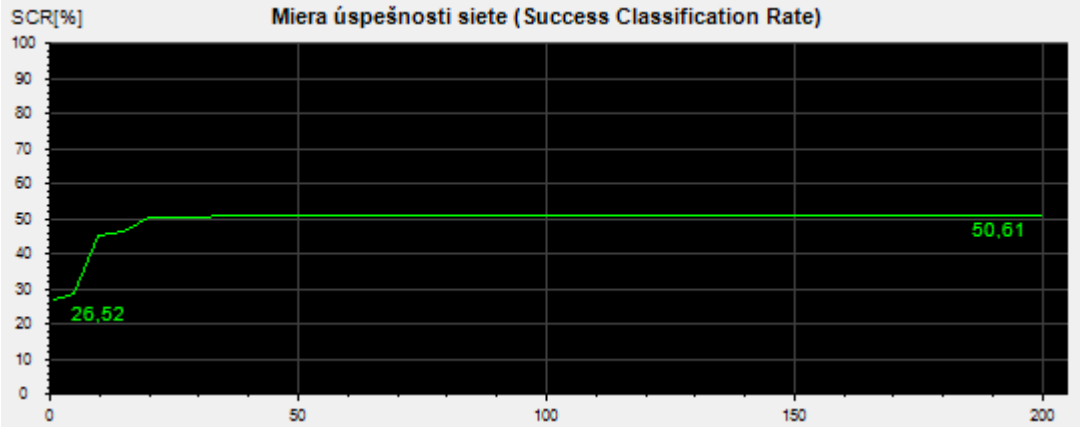
4.4.1 Učiaci cyklus 1

Epocha	Globálna chyba	RMS	SCR [%]
1	516,52	1,255	26,524
5	457,48	1,181	28,354
10	230,91	0,839	45,122
50	194,12	0,769	50,61
100	189,39	0,76	50,61
150	188,99	0,759	50,61
200	187,82	0,757	50,61

Tabuľka 4 – Učiaci cyklus neurónovej siete 1



Obr. 42 – Priemerná chyba za trénovaciu množinu pri učiacom cykle 1



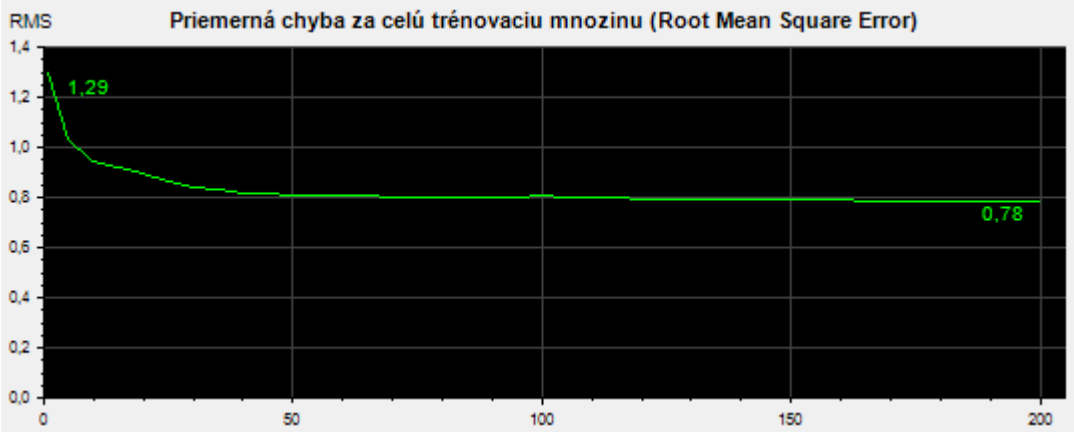
Obr. 43 – Miera úspešnosti siete pri učiacom cykle 1

Pri tomto učiacom cykle sa sieť nedokázala naučiť na danú problematiku. Zo začiatku úspešnosť siete stúpala avšak stúpanie sa zastavilo po 50 epochách a sieť sa už ďalej neučila. Priemerná chyba za trénovaciu množinu klesla na hodnotu 0,76, kde sa sieť dostala do lokálneho minima. Úspešnosť siete bola len málo nad 50%. Pre ďalší učiaci cyklus nastavím dimenzie siete na {25, 12, 6, 3} a resetujem sieť s náhodnými váhami.

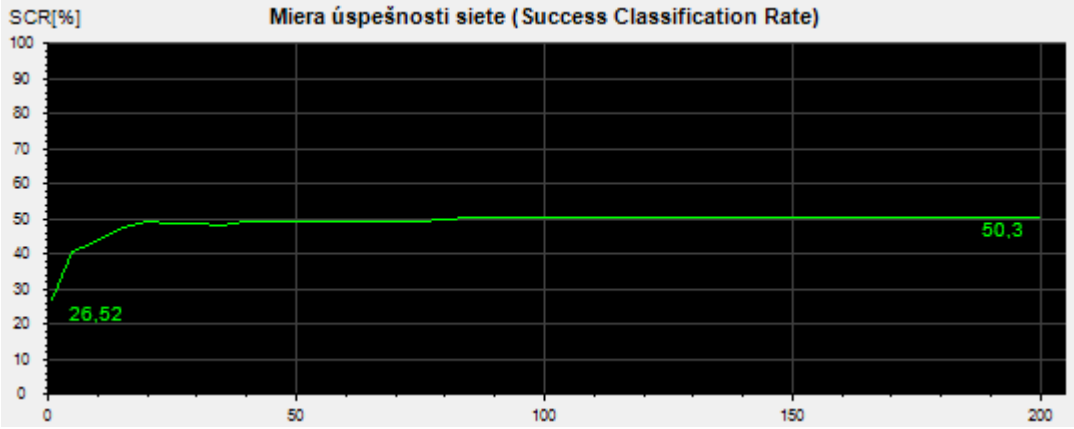
4.4.2 Učiaci cyklus 2

Epocha	Globálna chyba	RMS	SCR [%]
1	546,02	1,29	26,52
5	346,02	1,027	40,24
10	290,34	0,941	43,29
50	214,19	0,808	48,78
100	210,53	0,801	50,31
150	203,66	0,788	50,31
200	200,45	0,782	50,31

Tabuľka 5 – Učiaci cyklus neurónovej siete 2



Obr. 44 – Priemerná chyba za trénovaciu množinu pri učiacom cykle 2



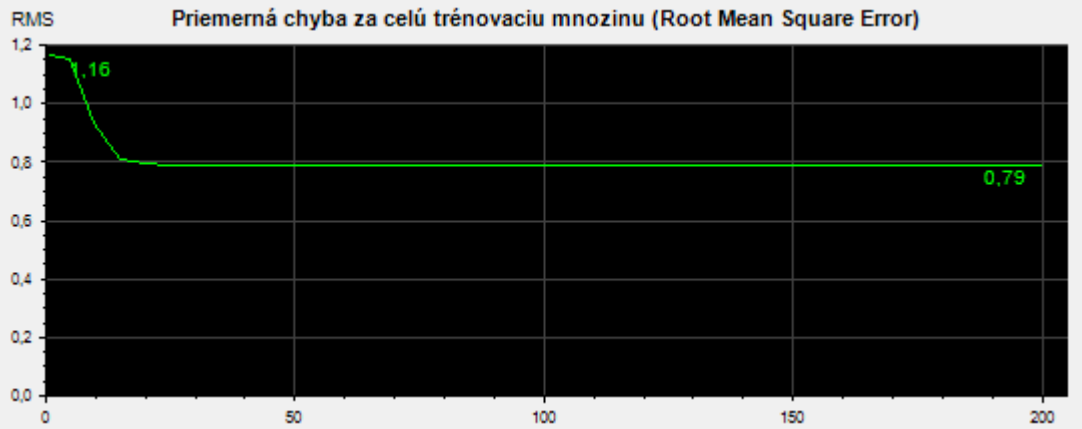
Obr. 45 – Miera úspešnosti siete pri učiacom cykle 2

Ani so zmenenými nastaveniami neurónovej siete sa počas druhého učiaceho cyklu sieť nedokázala naučiť rozpoznávať kategórie zvukových signálov. Ako pri prvom pokuse zo začiatku úspešnosť siete stúpala kde sieť dokázala rozpoznať zvuky, ktoré boli dostatočné odlišné od ostatných. Po 50 epochách sa stúpanie zastavilo a sieť sa už ďalej neučila. Priemerná chyba neklesla pod hodnotu 0,7. Sieť sa viac neučila ani po ďalších 1000 epochách.

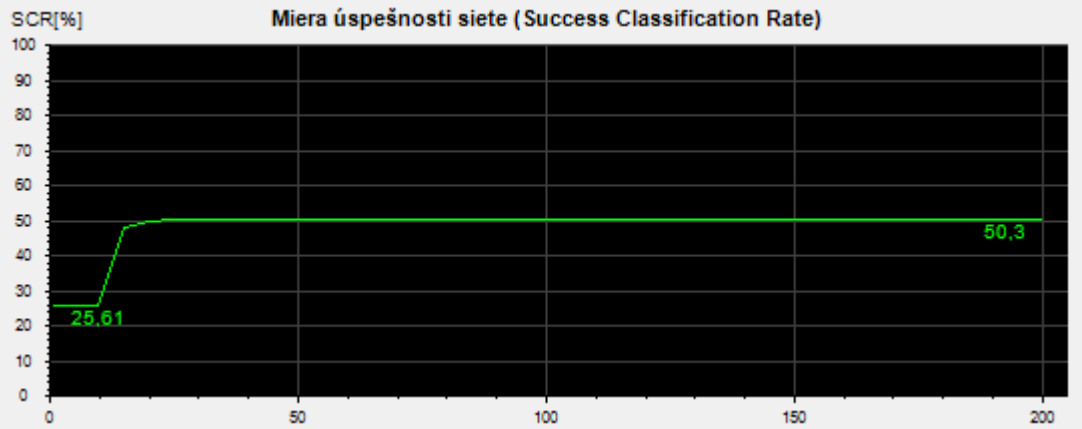
4.4.3 Učiaci cyklus 3

Epocha	Globálna chyba	RMS	SCR [%]
1	441,06	1,16	25,61
5	431,97	1,148	25,61
10	283,24	0,929	25,61
50	202,9	0,787	50,305
100	202,94	0,787	50,305
150	203,06	0,787	50,305
200	203,23	0,787	50,305

Tabuľka 6 – Učiaci cyklus neurónovej siete 3



Obr. 46 – Priemerná chyba za trénovaciu množinu pri učiacom cykle 3

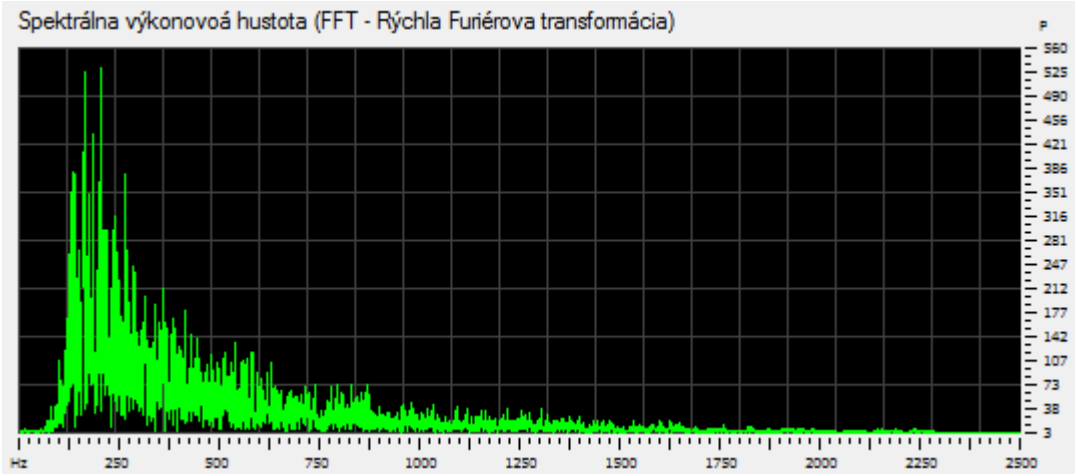


Obr. 47 – Miera úspešnosti siete pri učiacom cykle 3

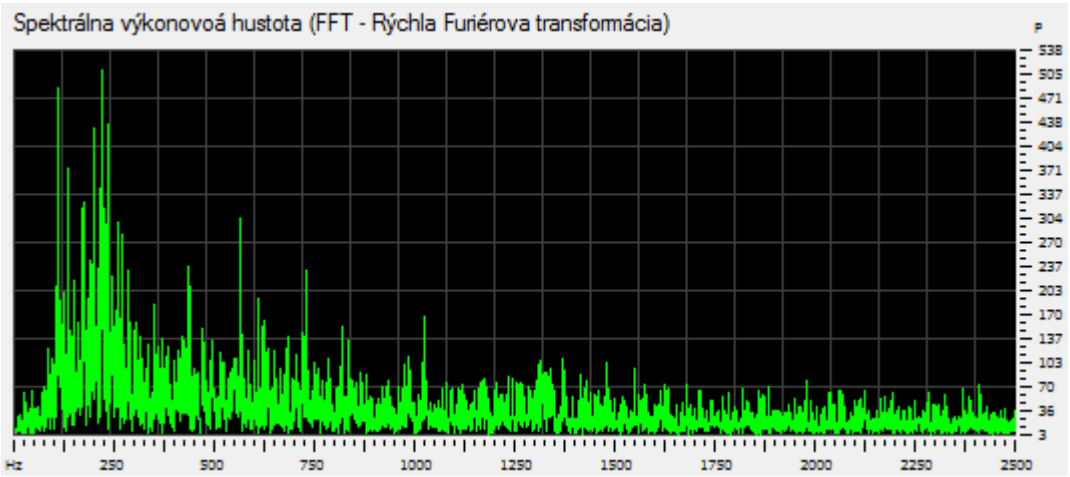
Z grafu je zrejmé, že sieť nedokázala správne klasifikovať zvuky ani pri poslednom učiacom cykle. Vykonal som ešte ďalšie učiace cykly s rôznymi nastaveniami siete, ale sieť sa aj tak nedokázala úspešne naučiť. Na záver môžem povedať, že sa sieť nepodarilo naučiť úspešne klasifikovať viac ako 50% zvukových vzoriek. Sieť nedokáže rozlíšiť zvuky, pretože vstupné vzorky sú príliš podobné vzorkám z ostatných kategórií. Tento problém ako aj jeho možné riešenia vysvetlím v nasledujúcej kapitole.

5 MOŽNOSTI PROKRAČOVANIA V DANEJ PROBLEMATIKE

Výsledky pokusu naučiť neurónovú sieť na klasifikáciu zdrojov zvuku poukázali na nedostatky extrahovania zvukových charakteristík pomocou Fourierovej transformácie. Pred pokusom som predpokladal, že frekvenčné spektrá jednotlivých zvukových signálov budú odlišné pre jednotlivé kategórie zbraní. Tento predpoklad sa však nepotvrdil a zistilo sa, že frekvenčné charakteristiky sú si veľmi podobné aj pri zvukoch z rôznych kategórií. Pre ukážku som zvolil dve charakteristiky spektrálnej výkonovej hustoty. Na obrázku 48 je charakteristika zvuku vybuchujúcej bomby teda zvuku z kategórie výbušniny a na obrázku 49 je zvuk výstrelu z 9 milimetrovej ručnej zbrane z kategórie malých zbraní.



Obr. 48 – Spektrálna výkonová hustota signálu z kategórie výbušniny



Obr. 49 – Spektrálna výkonová hustota signálu z kategórie malé zbrane

Z obrázkov je zrejmé, že tieto dve charakteristiky sú veľmi podobné a preto nedokázala neurónová sieť správne kategorizovať niektoré zvuky z trénovacej množiny. Jednotlivé kategórie tak nie sú jednoznačne odlíšiteľné od ostatných a preto algoritmus

backpropagation nedokázal nájsť také neurónové váhy, ktoré by boli riešením pre tento problém. Pri programovaní neurónovej siete a testovaní jeho učenia som došiel k záveru, že sieť je naprogramovaná správne, pretože sa dokázala naučiť na testovacie dáta. Problém je teda jednoznačne v trénovacej množine zvukov výstrelov zbraní a extrahovaní charakteristík, ktoré by boli spoločné len pre danú kategóriu.

Ako možnosť pokračovania v riešení tejto problematiky by som preto navrhol použiť miesto Fourierovej transformácie odlišné metódy extrahovania správnych charakteristík zvukových signálov. Tieto metódy by mali vyzdvihnúť spoločné charakteristiky v rámci kategórie a nájsť odlišnosti od ostatných kategórií. V tejto práci som využil analýzu signálu vo frekvenčnej oblasti. Ako ďalšie metódy analýzy by som navrhoval použiť metódy v časovej oblasti signálu ako je časová korelácia alebo metódy v oblasti časovo-frekvenčnej ako je diskrétna vlnková transformácia (DWT - discrete wavelet transform).

5.1 Časová korelácia

Existencia vzájomnej súvislosti medzi dvoma alebo viacerými signálmi v rôznych časoch sa nazýva časová korelácia. V užšom slova zmysle znamená korelácia vzájomný lineárny vzťah medzi veličinami x a y. Mieru korelácie vyjadruje korelačný koeficient, ktorý môže nadobúdať hodnôt od −1 až po +1.

Pri výpočte korelačného koeficientu r sa používajú hodnoty odhadu smerodajnej odchýlky s_x a s_y.

$$r = \frac{\sum_{i=1}^{n} \left(x_i - \bar{x}\right)\left(y_i - \bar{y}\right)}{(n-1).s_x.s_y}$$

Pre niektoré výpočty korelačného koeficientu je jednoduchšie použiť vzorec:

$$r = \frac{\overline{x.y} - \bar{x}.\bar{y}}{\sigma_x.\sigma_y}$$

v ktorom sú vyššie uvedené hodnoty vypočítané podľa nasledujúcich pravidiel:

$$\overline{x.y} = \frac{\sum_{i=1}^{n} x_i.y_i}{n} \qquad \bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \qquad \bar{y} = \frac{\sum_{i=1}^{n} y_i}{n} \qquad \sigma_x = \sqrt{\frac{1}{n}.\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2} \qquad \sigma_y = \sqrt{\frac{1}{n}.\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2}$$

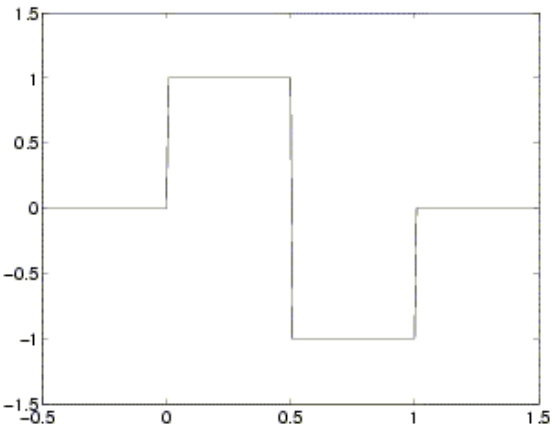
5.2 Diskrétna vlnková transformácia (DWT)

Vlnková (waveletová) transformácia je relatívne nový nástroj pre analýzu signálov v časovo-frekvenčnej oblasti. Vykonáva rozklad signálu $f(t)$ do priestoru generovaného bázou waveletu. Wavelety sú funkcie impulzného charakteru. Je ich možné obecne definovať ako odvodenia od matičného waveletu (mother wavelet) $\psi(t)$:

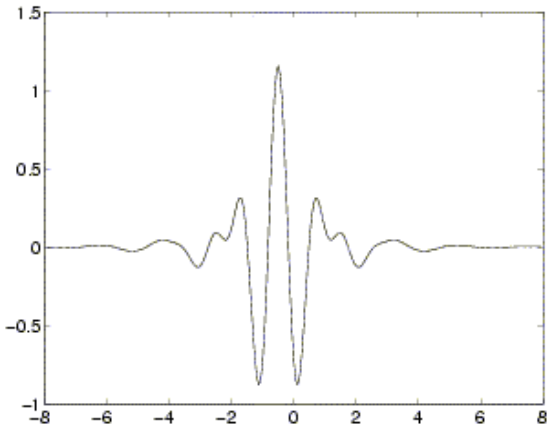
$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{t-b}{a}\right), \qquad a,b \in \Re, \qquad a \neq 0$$

Vlnková transformácia predstavuje alternatívu ku krátkodobej Fourierovej transformácii a poskytuje široké možnosti analýzy a spracovania viacrozmerných dát. Hlavná nevýhoda krátkodobej Fourierovej transformácie je v konštantnej šírke okennej funkcie behom celého výpočtu. Musíme sa rozhodnúť, či je pre naše účely výhodnejšie dobré rozlíšenie v čase, alebo vo frekvencii. Problém s rozlíšením rieši diskrétna vlnková transformácia, ktorá v priebehu výpočtu mení šírku okennej funkcie – waveletu.

Diskrétna vlnková transformácia (DWT) je transformácia odvodená od vlnkovej transformácie pre diskrétne vlnky (wavelety). Prvá DWT bola objavená maďarským matematikom Alfrédom Haarom. Pre vstup reprezentovaný zoznamom 2^n čísel je Haarová vlnková transformácia považovaná za najjednoduchšie spárovanie vstupných hodnôt uložením rozdielu a predaním súčtu do ďalšieho stupňa transformácie. Tento proces je opakovaný rekurzívne (na súčty). Waveletových funkcií je veľké množstvo, priebehy dvoch typov matičných waveletov sú zobrazené na obrázkoch.



Obr. 50 – Haarová vlnka



Obr. 51 – Meyerová vlnka

ZÁVER

Cieľom tejto diplomovej práce bolo oboznámiť čitateľa s využitím neurónových sietí pre klasifikáciu zvukových signálov. Vysvetlil som výhody aj nevýhody ich používania pre riešenie klasifikačných úloh. Oboznámil som čitateľa s históriou a vývojom neurónových sietí a priblížil som aj najpoužívanejšie typy týchto sietí.

V teoretickej časti som podrobne vysvetlil princíp učenia neurónových sietí na zadanú problematiku. Rozobral som teóriu učenia s učiteľom aj bez učiteľa. Vysvetlil som aktivačnú a adaptačnú fázu činnosti neurónovej siete. Spomenul som typy sietí s dopredným šírením informácie ako je viacvrstvová perceptrónová sieť ako aj Hopfieldové rekurentné siete, ktorými môže vstupná informácia putovať v oboch smeroch.

V praktickej časti som vytvoril databázu zvukov, ktorá pozostáva z výstrelov zbraní. Uskutočnil som meranie v teréne, pri ktorom bola táto databáza zvukov rozšírená desať násobne o výstrely zbraní z rôznych vzdialeností. V programovacom jazyku C# som naprogramoval viacvrstvovú neurónovú sieť, ktorá pre učenie využíva algoritmus backpropagation zo spätným šírením chyby. Pre učenie a testovanie tejto neurónovej siete som vytvoril windows aplikáciu pozostávajúcu z dvoch hlavných formulárov.

Pri programovaní som sieť testoval pomocou trénovacej množiny kosatcových rastlín, ktorá sa často vyskytuje v literatúre rozpoznávania. Výsledky testovania som zobrazil v tabuľkách aj graficky. Z tohto testovania vyplynul záver, že sieť je správne naprogramovaná a dokáže sa naučiť úspešné klasifikovať rastliny do kategórií.

Pri učení neurónovej siete som miesto testovacej množiny použil databázu zvukov výstrelov zbraní, ktorú som si predtým vytvoril. Jednotlivé výstrely som rozdelil do troch kategórií a to malé zbrane, stredné zbrane a výbušniny. Z učenia neurónovej siete, ale vyplynulo, že sa sieť nedokáže naučiť správne klasifikovať jednotlivé zvuky.

Pred pokusom som predpokladal, že charakteristiky jednotlivých zvukových signálov budú odlišné pre jednotlivé kategórie zbraní. Tento predpoklad sa však nepotvrdil a zistilo sa, že frekvenčné charakteristiky sú si veľmi podobné a sieť ich tak nedokáže zaradiť do správnej kategórie. Tento problém by bolo možné vyriešiť použitím inej metódy analýzy zvukových signálov, pretože Fourierova transformácia sa pri riešení tohto problému neosvedčila.

ZÁVER V ANGLIČTINE

The aim of this thesis was to familiarize the reader with usage of neural networks for classification of audio signals. I explained the advantages and disadvantages of their usage for solving classification tasks. I informed the reader with the history and development of neural networks and I mentioned the most common types of networks.

In the theoretical part of this work I explain in detail the principle of training neural networks to a specified task. I informed about the theory of supervised and unsupervised network learning style. I explained the activation and adaptation phase of the neural network. I mentioned the feed forward types of networks such as multilayer perceptron network and also I mentioned the Hopfield recurrent neural network in which an input information can be distributed in both directions.

In the practical part I created a database of sounds, which consists of weapon shots sounds. I have made measurements in field, in which the database was expanded ten times with sounds of weapon shots from various distances. In C# programming language I created a multilayer neural network which uses the backpropagation algorithm for learning. It's an algorithm which propagates the error back to input network. For learning and testing of the neural network I created a windows application, consisting of two forms.

I tested the network using a training set of iris plants, which often occurs in the recognition topic literature. The test results were displayed both in tables and graphically. This testing faze concluded that the network is properly programmed and it can learn to successfully classify the plants into categories.

In the learning faze of neural network, I used a database of weapon shots sounds, which I have previously created. I divided the sounds into three categories: small arms, medium weapons and explosives. Unfortunately the learning faze showed that the network can not learn properly to classify the different sounds.

I assumed, before the experiment, that the characteristics of sound signals will be different between weapon categories. This assumption was not confirmed and it was found that the frequency characteristics are very similar so the network can not classify the sounds in the right category. This problem could be solved by using another audio signal analysis method because the Fourier transform didn't proved to solve this problem.

ZOZNAM POUŽITEJ LITERATÚRY

[1] HLAVÁČ, Václav; ŠONKA, M.. *Počítačové vidění*. [cit. 2010-04-17] Grada, Praha, 1992.

[2] MAŘÍK, V.; ŠTEPÁNKOVÁ, O.; LAŽANSKÝ, J. a kolektiv. *Umělá inteligence 2*. [cit. 2010-04-17]. Academia, Praha, 1997.

[3] KATRÁK, Miroslav. *Rozpoznávanie reči použitím neurónových sietí*. Košice, 2007. 50 s. [cit. 2010-04-21]. Dizertační práce. Technická univerzita v Košiciach.

[4] MATOUŠEK, Václav. *Rozpoznávání a klasifikace. Umělá inteligence a rozpoznávání* [online]. 2010, [cit. 2010-04-21]. Dostupný z WWW: <http://www.kiv.zcu.cz/studies/predmety/uir/predn/P4/FThema4.pdf>

[5] KOTEK, Z.; MAŘÍK, V.. *Metody rozpoznávaní a jejich aplikace*. [cit. 2010-04-24]. Academia, Praha, 1993

[6] *Wikipedia.sk* [online]. 2010 [cit. 2010-04-24]. Mikrofón. Dostupné z WWW: <http://sk.wikipedia.org/wiki/Mikrof%C3%B3n>.

[7] KOPEČEK, Ivan. *Zpracování digitalizovaného signálu.* Fakulta informatiky Masarykovy univerzity [online]. 2010, 4, [cit. 2010-04-28]. Dostupný z WWW: <www.fi.muni.cz/~kopecek/upzr3.ppt>.

[8] HLAVÁČ, Václav. *Rozpoznávání s Markovskými modely* [online]. Praha, 2010 [cit. 2010-05-05]. Dostupné z WWW: <http://cmp.felk.cvut.cz/ ~hlavac/TeachPresCz/31Rozp/61MarkovianPR.pdf>.

[9] PSUTKA, Jozef. *Komunikace s počítačem mluvenou řeči*. [cit. 2010-05-08]. Akademie věd České republiky, Praha 1995.

[10] KUFNER, Alois; KADLEC, Jan. *Fourierovy řady*. [cit. 2010-05-08]. Academia 1969

[11] ČÍŽEK, Václav. *Diskrétní Fourierova transformace a její použití*. [cit. 2010-05-10]. SNTL 1981

[12] ZELINKA, I., *Umělá inteligence I.* [cit. 2010-05-10]. *Neuronové sítě a genetické algoritmy*. Brno: VITIUM, 1998.

[13] VOLNÁ, Eva. *Umělá inteligence : Vícevrstvá neuronová síť jako univerzální aproximátor.* [cit. 2010-05-10]. Automatizace. 2009, 52, 11, s. 658-660.

[14] *Wikipedia.cz* [online]. 2010 [cit. 2010-05-11]. Neuronová síť. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Neuronov%C3%A1_s%C3%AD%C5%A5>.

[15] NEUDERT, Leoš. *www.fi.muni.cz* [online]. 2000 [cit. 2010-05-11]. *Historie neuronových počítačů a sítí.* Dostupné z WWW: <*http://www.fi.muni.cz/usr/jkucera/pv109/2000/xneudert.html*>.

[16] ŠTĚPÁNOVSKÝ, Jan. *Úvod do neuronových sítí* [online]. 2010 [cit. 2010-05-13]. Dostupné z WWW*: <www.lopikus.cz/temp/BMI-NAN.ppt*>.

[17] DERENÍK, Dávid. *Matematické modely v technickej analýze cenných papierov. Brno, 2006. 37 s.* [cit. 2010-05-14]. *Bakalářská práce.* MASARYKOVA UNIVERZITA.

[18] FORIŠEK, Michal. *People.sk* [online]. 2006 [cit. 2010-05-19]. *UMELÉ NEURÓNOVÉ SIETE*. Dostupné z WWW: <http://people.ksp.sk/~misof/skola/!to%20process/Neuronove%20siete/uns.pdf>.

[19] *Wikipedia.cz* [online]. 2010 [cit. 2010-05-22]. *Přenosové funkce.* Dostupné z WWW: <http://cs.wikipedia.org/wiki/Neuronov%C3%A1_s%C3%AD%C5%A5#P.C5.99enosov.C3.A9_funkce>.

[20] VOJÁČEK, Antonín. *Samoučící se neuronová síť - SOM, Kohonenovy mapy* [online]. 2006, [cit. 2010-05-18]. Dostupný z WWW: <http://www.kiv.zcu.cz/studies/predmety/uir/NS/Samouc_NN2.pdf>.

[21] HLAVÁČ, Václav. *Umělé neuronové sítě z pohledu rozpoznávání* [online]. 2007, [cit. 2010-05-24]. Dostupný z WWW: <http://cmp.felk.cvut.cz/~hlavac/TeachPresCz/31Rozp/72UmeleNN.ppt>.

[22] WILSON, Scott. *https://ccrma.stanford.edu* [online]. 2003 [cit. 2010-05-24]. WAVE PCM soundfile format . Dostupné z WWW: <https://ccrma.stanford.edu/courses/422/projects/WaveFormat/>.

[23] *www.fftw.org* [online]. 2009 [cit. 2010-05-25]. FFTW. Dostupné z WWW: <http://www.fftw.org/>.

[24] *Dynamic Notions* [online]. 2008 [cit. 2010-05-27]. Training Neural Networks Using Back Propagation in C#. Dostupné z WWW: <http://dynamicnotions.blogspot.com/2008/09/training-neural-networks-using-back.html>.

[25] *UCI Machine Learning Repository* [online]. 1988 [cit. 2010-06-01]. Iris Data Set . Dostupné z WWW: <http://archive.ics.uci.edu/ml/datasets/Iris>.

[26] *Wikipedia.cz* [online]. 2010 [cit. 2010-06-01]. Korelácia. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Korelace>.

[27] HEJDA, Tomáš. *Vybrané partie matematiky* [online]. 2010 [cit. 2010-06-01]. Waveletová transformace, s. . Dostupné z WWW: <http://www.tomashejda.cz/soubory/X17VPM-reserse-WT.pdf>.

[28] *Wikipedia.cz* [online]. 2010 [cit. 2010-06-01]. Diskrétní vlnková transformace. Dostupné z WWW: <http://cs.wikipedia.org/wiki/Diskr%C3%A9tn%C3%AD_vlnkov%C3%A1_transformace>.

[29] *Obecná klasifikačná úloha* [online]. 2010 [cit. 2010-06-02]. Dostupné z WWW: <http://www.kiv.zcu.cz/studies/predmety/uir/predn/P4/FThema4.pdf>.

[30] *Kondenzátorový mikrofón* [online]. 2010 [cit. 2010-06-02]. Dostupné z WWW: <http://upload.wikimedia.org/wikipedia/commons/6/60/Kondensatormikrofon_SK.svg>.

[31] *Graf a kosínusovej vlny* [online]. 2010 [cit. 2010-06-02]. Dostupné z WWW: <http://klobouk.fsv.cvut.cz/~vrf/diplom/image74.gif>.

[32] *Spektrum kosínusovej vlny* [online]. 2010 [cit. 2010-06-02]. Dostupné z WWW: <http://klobouk.fsv.cvut.cz/~vrf/diplom/image75.gif>.

[33] *Model umelého neurónu* [online]. 2010 [cit. 2010-06-02]. Dostupné z WWW: <http://weboit.sk/app/files/subory/312_perc.png>.

[34] *Model biologického neurónu* [online]. 2010 [cit. 2010-06-02]. Dostupné z WWW: <http://programujte.com/galerie/2005/06/200506080117_ai1.jpg>.

[35] *Grafické znázornenie funkcie XOR* [online]. 2010 [cit. 2010-06-02]. Dostupné z WWW: <http://www.ai-cit.sk/source/publications/books/NS1/html/img131.gif>.

[36] *Hopfieldova sieť s tromi neurónmi* [online]. 2010 [cit. 2010-06-02]. Dostupné z WWW: <http://weboit.sk/app/files/subory/322_siet.png>.

[37] *Skoková funkcia* [online]. 2010 [cit. 2010-06-02]. Dostupné z WWW: <http://upload.wikimedia.org/wikipedia/commons/a/ac/HardLimitFunction.png>.

[38] *Sigmoidálna funkcia* [online]. 2010 [cit. 2010-06-02]. Dostupné z WWW: <http://upload.wikimedia.org/wikipedia/commons/9/96/SigmoidFunction.svg>.

[39] *Trojvrstvová perceptrónová sieť* [online]. 2010 [cit. 2010-06-02]. Dostupné z WWW: <http://programujte.com/galerie/2009/03/200903110844_ui_1.png>.

[40] *Možné štruktúry usporiadania neurónov Kohonenových máp* [online]. 2010 [cit. 2010-06-02]. Dostupné z WWW: <http://www.kiv.zcu.cz/studies/predmety/uir/NS/Samouc_NN2.pdf>.

[41] *Štruktúra Kohonenovej mapy s víťazným neurónom BMU* [online]. 2010 [cit. 2010-06-02]. Dostupné z WWW: <http://www.kiv.zcu.cz/studies/predmety/uir/NS/Samouc_NN2.pdf>.

[42] *Hopfieldova sieť so 6 neurónmi* [online]. 2010 [cit. 2010-06-02]. Dostupné z WWW: <http://hq.alert.sk/~mandos/fmfi-uk/Informatika/Neuronove%20Siete/NS2/img15.gif>.

[43] *Grafická reprezentácia RIFF štandardu* [online]. 2010 [cit.2010-06-02].Dostupné z WWW: <https://ccrma.stanford.edu/courses/422/projects/WaveFormat/wav-sound-format.gif>.

[44] *Význam jednotlivých bytov pre konkrétny WAVE súbor* [online]. 2010 [cit. 2010-06-02]. Dostupné z WWW: <https://ccrma.stanford.edu/courses/422/projects/WaveFormat/wave-bytes.gif>.

[45] *Haarová vlnka* [online]. 2010 [cit. 2010-06-02]. Dostupné z WWW: <http://measure.feld.cvut.cz/usr/staff/smid/wavelets/figures/haarwave.gif>.

[46] *Meyerová vlnka* [online]. 2010 [cit. 2010-06-02]. Dostupné z WWW: <http://www.uamt.feec.vutbr.cz/~richter/vyuka/0910_mpov/tmp/integral_tr_wavelety_html_m850f957.gif>.

ZOZNAM POUŽITÝCH SYMBOLOV A SKRATIEK

DTW	Dynamic Time Warping - dynamické skreslenie času
HMM	Hidden Markov Models - skryté Markové modely
FFT	Fast Fourier Transform - rýchla Fourierova transformácia
DFT	Discrete Fourier Transform - diskrétna Fourierova transformácia
ADALINE	Adaptive Linear Neuron - adaptívny lineárny neurón
XOR	Kombinačný logický obvod exclusive OR
DARPA	Defense Advanced Research Projects Agency
PDP	Parallel Distributed Processing Group
IEEE	Institute of Electrical and Electronics Engineers
INNS	International Neural Network Society
ART	Adaptive Resonance Theory
SOM	Self Organizing Maps - samoorganizujúce sa mapy
GPS	Global Positioning System - globálny polohový družicový systém
WAV (WAVE)	Waveform audio format - zvukový formát
RIFF	Resource Interchange File Format - souborový formát firmy Microsoft
FFTW	Fastest Fourier Transform in the West
ECMA	European Computer Manufacturers Association
ISO	International Organization for Standardization
GUI	Graphical User Interface - grafické užívateľské rozhranie
XML	Extensible Markup Language
RMS	Real Mean Square Error - priemerná chyba za trénovaciu množinu
SCR	Success Clasification Rate - miera úspešnosti neurónovej siete
DWT	Discrete Wavelet Transform - diskrétna vlnková transformácia

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ZOZNAM PRÍLOH

PŘÍLOHA P I: ZDROJOVÝ KÓD TRIEDY NETWORK

```
/**
* Hlavna trieda obsahujuca metody pre pracu z neuronovou sietou
* Dimenzie siete sa zadavaju ako pole integer hodnot napr. {10, 6, 3, 5}
* To reprezentuje siet ktora ma 10 vstupov 2 skryte vrstvy prva skryta
* vrstva ma 6 neuronov, druha 3 neurony a siet ma 5 vystupnych neuronov
*/

public class Network : List<Layer>
{
    private int[] dimensions;
    public List<Pattern> patterns;
    public static int neuronCounter = 0;

    //getter pre pole vstupnych neuronov
    private Layer Inputs { get { return base[0]; } }

    //getter pre pole vystupnych neuronov
    private Layer Outputs { get { return base[base.Count - 1]; } }

    //construktor triedy Network
    public Network(int[] dimensions)
    {
        this.dimensions = dimensions;
        Initialise();
    }

     //funkcia volana pri spusteni triedy
    public void Initialise()
    {
        Network.neuronCounter = 0;
        base.Clear();
        base.Add(new Layer(dimensions[0]));
        for (int i = 1; i < dimensions.Length; i++)
        {
            base.Add(new Layer(dimensions[i], base[i - 1],
                               new Random()));
        }
    }

    //nacitanie vsetkych trenovacich vzorov do siete
    public void LoadPatterns(List<PatternData> patternDatas)
    {
        object sett = MainForm.neuronNetworkSettings["loadTrainingSet"];
        this.patterns = new List<Pattern>();
        if (sett.ToString() == "no")
        {
            foreach (PatternData patternData in patternDatas)
            {
                patterns.Add(new Pattern(patternData.data,
                                         patternData.categoty - 1));
            }
        }
        else
        {
            //nacitanie simulacnych dat zo suboru data.csv
            StreamReader reader = File.OpenText("data.csv");

            //parameter pre parsovanie string hodnoty
            NumberFormatInfo culture =
            CultureInfo.InvariantCulture.NumberFormat;
```

```
            //precitame kazdy riadok suboru
            while (!reader.EndOfStream)
            {
                //jeden riadok je jedna poloyka v trenovacej mnoyine
                string line = reader.ReadLine();
                string[] value = line.Split(',');

                double[] inputs = new double[4];
                for (int i = 0; i < 4; i++)
                {
                    inputs[i] = double.Parse(value[i], culture);
                }

                //pridame vzor pre vstup do siete
                this.patterns.Add(new Pattern(inputs, int.Parse(value[4],
                                              culture) - 1));
            }

            reader.Close();
        }
    }


    //spusti jednu epochu trenovanie siete
    public double Train()
    {
        double error = 0;

        //pre kazdy polozku v trenovacich vzoroch
        foreach (Pattern pattern in this.patterns)
        {
            //aktivacia trenovacieho vzrou
            Activate(pattern);
            for (int i = 0; i < Outputs.Count; i++)
            {
                //ziskame rozdiel vo vystupe
                double delta = pattern.Outputs[i] - Outputs[i].Output;

                //spocitame chybu
                Outputs[i].CollectError(delta);
                error += Math.Pow(delta, 2);
            }

            //prenastavy vahy
            AdjustWeights();
        }
        return error;
    }


    //zisti uspesnost siete
    public double GetSuccessRate()
    {
        double success = 0;
        foreach (Pattern pattern in this.patterns)
        {
            int temp = pattern.MaxOutput;
            int output = this.Test(pattern.Inputs) - 1;
            if (output == temp) success++;
        }
        return Math.Round((success / (this.patterns.Count + 0.00001)) *
                          100, 5);
    }
```

```
        //otestuje siet pomocou zadanych vstupov
        public int Test(double[] inputData)
        {
            //aktivacna faza
            this.Activate(new Pattern(inputData, -1));

            int nr = 0;
            double max = 0;
            int winner = 0;

            //najdeme neuron ktory ma najvacsiu hodnotu
            foreach (Neuron neuron in Outputs) {
                nr++;
                if (neuron.Output > max) {
                    max = neuron.Output;
                    winner = nr;
                }
            }
            return winner;
        }



        //aktivacna faza neuronovej siete
        private void Activate(Pattern pattern)
        {
            for (int i = 0; i < Inputs.Count; i++)
            {
                Inputs[i].Output = pattern.Inputs[i];
            }
            for (int i = 1; i < base.Count; i++)
            {
                foreach (Neuron neuron in base[i])
                {
                    //aktivujeme kazdy neuron v sieti
                    neuron.Activate();
                }
            }
        }


        //adaptacna faza neuronovej siete
        private void AdjustWeights()
        {
            for (int i = base.Count - 1; i > 0; i--)
            {
                foreach (Neuron neuron in base[i])
                {
                    //pre kazdy neuron prenastavime vahy
                    neuron.AdjustWeights();
                }
            }
        }
    }
```

PŘÍLOHA P II: ZDROJOVÝ KÓD TRIEDY LAYER

```
/**
 * Vytvorenie vrstvy neuronov za pomoci jednoducheho
 * zoznamu List<Neuron>, ktory obsahuje jednotlive neurony
 */

public class Layer : List<Neuron>
{
    //vytvorenie vrsy o danej velkosti
    public Layer(int size)
    {
        for (int i = 0; i < size; i++)
            base.Add(new Neuron());
    }

    //vytvorenie specifickej Layer vrstvy danej velkosti
    public Layer(int size, Layer layer, Random rnd)
    {
        for (int i = 0; i < size; i++)
            base.Add(new Neuron(layer, rnd));
    }
}
```

PŘÍLOHA P III: ZDROJOVÝ KÓD TRIEDY PATTERN

```
/**
 *  Trieda sluzi pre spracovanie dat do podoby,
 *  ktorej rozumie neuronova siet.
 *  Jedna sa o vektor vstupov a im priradenym vystupov
 */

public class Pattern
{
    private double[] inputs;    //vstupy siete
    private double[] outputs;   //vystupy siete
    //getter pre vstupy siete
    public double[] Inputs { get { return inputs; } }
    //getter pre vystupy siete
    public double[] Outputs { get { return outputs; } }
    //getter pre neuron s maximalnou hodnotou
    public int MaxOutput
    {
        get
        {
            int item = -1;
            double max = double.MinValue;
            for (int i = 0; i < Outputs.Length; i++)
            {
                if (Outputs[i] > max)
                {
                    max = Outputs[i];
                    item = i;
                }
            }
            return item;
        }
    }
    //konstruktor triedy pripravy vstupne data pre neuronovu siet
    public Pattern(double[] inputData, int outputNeuron)
    {
        //nastavenia siete
        SortedList sett = MainForm.neuronNetworkSettings;
        //pocet vstupov a vystupovziskame z nasvaveni vo formulari
        int inputDims = int.Parse(sett["inputNeurons"].ToString());
        int outputDims = int.Parse(sett["outputNeurons"].ToString());

        //do premennej inputs vlozive vstupne data
        inputs = new double[inputDims];
        for (int i = 0; i < inputDims; i++)
        {
            inputs[i] = (double)inputData[i];
        }
        //ak urcujeme aj vystupnu kategoriu
        if (outputNeuron > -1)
        {
            //do premennej outputs vlozime vystupne data
            outputs = new double[outputDims];
            for (int i = 0; i < outputDims; i++)
            {
                if (i == outputNeuron)
                    outputs[i] = (double)1;
                else
                    outputs[i] = (double)0;
            }
        }
    }
}
```

PŘÍLOHA P IV: ZDROJOVÝ KÓD TRIEDY NEURON A WEIGHT

```
/**
 * Tato trieda reprezentuje neuron a obsahuje
 * aktivacnu a adaptacnu fazu neuronu
 */

public class Neuron
{
    private double bias;                    //Skreslenie neuronu.
    private double error;                   //Suma chyby
    private double input;                   //Suma vstupnych hodnot
    private double gradient = 5;            //Strmost sigmoidalnej krivky
    private double learnRate = 0.01;        //Rychlost ucenia
    private double output = double.MinValue;//Vystup neuronu
    public List<Weight> weights;            //Zoznam vah pre dany neuron

    //getter pre vahy neuronu
    public List<Weight> Weights { get { return weights; } }

    //konstruktor neuronu
    public Neuron() { }

    //konstruktor pre vybranu vrstvu a nahodnou vahovou hodnotou
    public Neuron(Layer inputs, Random rnd)
    {
        //nastavenia siete
        SortedList sett = MainForm.neuronNetworkSettings;
        SortedList savedWeights = MainForm.trainedNeuonWeights;
        string appDir =
        Path.GetDirectoryName(Application.ExecutablePath);
        NumberFormatInfo culture =
        CultureInfo.InvariantCulture.NumberFormat;

        //ak je nastavena rychlost ucenia
        if (sett["learnRate"] != null)
        {
            this.learnRate = double.Parse(sett["learnRate"].ToString(),
                                          culture);
        }

        //vytvorenie zoznamu vah
        weights = new List<Weight>();

        //pre kazdy vstup zo vsetkych vstupon
        foreach (Neuron input in inputs)
        {
            //pocitadlo neuronov
            Network.neuronCounter++;

            //vaha je nastavena nahodne
            double weight = rnd.NextDouble() * 2 - 1;

            //ak je siet naucena zoskame vahy z ulozeneho suboru
            if (sett["networkTrained"].ToString() == "1" &&
                File.Exists(appDir + "\\settings\\neuon_weights.xml") &&
                savedWeights.ContainsKey("weight" +
                Network.neuronCounter))
            {
                weight = double.Parse(savedWeights["weight" +
                                      Network.neuronCounter].ToString());
            }
```

```
            //do zoznamu pridame novy objekt triedy Weight
            Weight w = new Weight();
            w.InputId = Network.neuronCounter;
            w.Input = input;
            w.Value = weight;
            weights.Add(w);
        }
    }

    //aktivacna faza neuronu
    public void Activate()
    {
        error = 0;
        input = 0;

        //vynasobime kazdu vahu s vystupom neuronu
        foreach (Weight w in weights)
        {
            input += w.Value * w.Input.Output;
        }
    }

    //vypocet chyby
    public void CollectError(double delta)
    {
        if (weights != null)
        {
            error += delta;

            //vypocet chyby pre kazdu vahovu hodnotu
            foreach (Weight w in weights)
            {
                w.Input.CollectError(error * w.Value);
            }
        }
    }

    //prenastavime vahy pomocou backpropagation
    public void AdjustWeights()
    {
        for (int i = 0; i < weights.Count; i++)
        {
            weights[i].Value += error * Derivative * learnRate *
                                weights[i].Input.Output;
        }

        bias += error * Derivative * learnRate;
    }

    //derivacia
    private double Derivative
    {
        get
        {
            double activation = Output;
            return activation * (1 - activation);
        }
    }
```

```
    //getter a setter pre vystu neuronu
    public double Output
    {
        get
        {
            if (output != double.MinValue)
            {
                return output;
            }
            return 1 / (1 + Math.Exp(-gradient * (input + bias)));
        }

        set
        {
            output = value;
        }
    }
}


/**
 * Trieda obsahuje ukazovatel na neuron a vahu spojenia neuronu
 */

public class Weight
{
    public int InputId;
    public Neuron Input;
    public double Value;
}
```