Slovenská technická univerzita v Bratislave
Ústav informatizácie, automatizácie a matematiky

Fakulta chemickej a potravinárskej technológie
Akademický rok: 2012/2013
Evidenčné číslo: FCHPT-5415-44230

STU
FCHPT

ZADANIE BAKALÁRSKEJ PRÁCE

Študent:	**Martin Krippel**
ID študenta:	44230
Študijný program:	automatizácia, informatizácia a manažment v chémii a potravinárstve
Kombinácia študijných odborov:	5.2.14 automatizácia, 5.2.52 priemyselné inžinierstvo
Vedúci práce:	RNDr. Štefan Boor, PhD.

Názov práce: **Klasifikácia a predikcia vlastností chemických látok pomocou neurónových sietí**

Špecifikácia zadania:

Cieľom práce je naštudovať a použiť rôzne metódy učenia viacvrstvovej neurónovej siete s dopredným šírením a vyhodnotiť ich vplyv na schopnosti neurónovej siete rýchle a presne klasifikovať, predikovať vlastnosti chemických látok v závislosti od ich vstupných charakteristík. Pre výsledné neurónové siete treba vybrať vhodné dáta na overenie časovej náročnosti učiaceho procesu, ako aj schopnosti natrénovanej neurónovej siete predikovať, či klasifikovať vlastnosti látok.
Predpokladom riešenia problematiky bakalárskej práce je schopnosť realizácie vlastného programu v programovacom jazyku, C, C++.

Riešenie zadania práce od: 18. 02. 2013
Dátum odovzdania práce: 25. 05. 2013

Slovenská technická univerzita v Bratislave Fakulta chemickej a potravinárskej technológie dekanát - 2 -

L. S.

Martin Krippel
študent

prof. Ing. Miroslav Fikar, DrSc.
vedúci pracoviska

prof. Ing. Miroslav Fikar, DrSc.
garant študijného programu

Rád by som sa týmto spôsobom poďakoval môjmu školiteľovi a vedúcemu práce RNDr. Štefanovi Boorovi, PhD. za odborné vedenie, inšpiratívne návrhy, cenné pripomienky a rady a za čas, ktorý si našiel na zodpovedanie mojich otázok.

Abstrakt

Autor: Martin Krippel

Názov bakalárskej práce: Klasifikácia a predikcia vlastností chemických látok pomocou neurónových sietí

Škola: Slovenská technická univerzita v Bratislave

Fakulta: Fakulta chemickej a potravinárskej technológie

Ústav: Ústav informatizácie, automatizácie a matematiky

Študijný program: Automatizácia, informatizácia a manažment v chémii a potravinárstve

Vedúci práce: RNDr. Štefan Boor, PhD.

Rozsah práce: 40 strán

Cieľom tejto bakalárskej práce je konštrukcia viacvrstvovej doprednej neurónovej siete za účelom predikcie zvolených vlastností chemických látok na základe ich vstupných charakteristík. Práca sa v teoretickej časti zaoberá matematickým opisom neurónovej siete a učiacich algoritmov, v praktickej časti opisuje voľbu dát, nastavenie parametrov siete, proces trénovania a porovnanie efektivity použitých algoritmov. Práca obsahuje aj vybrané časti zdrojového kódu programu napísaného v programovacom jazyku C, ako aj grafy zobrazujúce priebeh trénovania prostredníctvom účelovej funkcie trénovacej a testovacej množiny.

Kľúčové slová: neurónova sieť, trénovací algoritmus, predikcia, Backpropagation algoritmus, genetický algoritmus

Abstract

Author: Martin Krippel

Thesis title: The Classification and Prediction of Properties of Chemical Substances Using Neural Networks

University: Slovak University of Technology in Bratislava

Faculty: Faculty of Chemical and Food Technology

Institute: Institute of Information Engineering, Automation, and Mathematics

Study program: Automation, information engineering a management in chemical and food technologies

Advisor: RNDr. Štefan Boor, PhD.

Thesis lenght: 40 pages

The objective of this thesis is construction of multilayer feedforward neural network in purpose of prediction of desired properties of chemical substances based on their input characteristics. In theoretical section, the thesis deals with mathematical description of neural network and the learning algorithms. In practical section, it describes selection of input data, setting up parameters of neural network, training process and compares efficiency of used algorithms. The thesis also contains fragments of source code of computer program written in C programming language. Graphs representing cost function of training set and test set are also included in the thesis. These graphs are useful for presenting progress of network training process.

Keywords: neural network, learning algorithm, prediction, Backpropagation algorithm, genetic algorithm

Obsah práce

1. Úvod

Teória neurónových sietí je inšipirovaná živými nervovými sústavami a vychádza teda z neurofyziologických poznatkov. Jej snahou je vysvetliť správanie sa v závislosti na spracovávaní informácií v nervových bunkách. Niekedy sú umelé neurónové siete označované aj ako modely mozgu bez mysle (brain without mind), keďže ich cieľom je pochopiť nervový systém, ale pri tom sa nezaoberajú psychikou.

Inšpirácia na použitie modelu l'udského mozgu pre tvorbu programu na nás kladie požiadavku, aby sa program dokázal aj niečo naučiť a nielen vykonával presné sekvencie inštrukcií. Medzi takéto modely patria aj neurónové siete, ktoré sú v podstate súbormi navzájom prepojených jednoduchých prvkov – neurónov, ktoré vykonávajú jednoduché matematické operácie a ich spojenia sú ohodnotené reálnymi číslami - váhami. Keď sieti poskytneme dostatok vzorov (úlohy a ich riešenia), z ktorých sa môžu učit' daný problém, takáto sieť sa časom naučí rozpoznávat' a sama riešit' úlohy daného typu. Tento proces sa volá trénovanie siete a jeho podstata je modifikácia intenzít váhových prepojení takým spôsobom, aby neurónová sieť adekvátne reagovala na vstupy, tj. aby produkovala požadované výstupy..

Z inžinierskeho hľadiska sú neurónové siete populárne a často používané modely, a to najmä vzhľadom na jednoduchosť ich použitia a schopnosť rýchlo dosiahnuť uspokojivé výsledky.

V súčasnosti už existuje veľký počet rôzných architektúr neurónových sietí s rôznymi smermi toku informácie, ja sa však budem zaoberať len s modelom viacvrstvovej neurónovej siete s dopredným šírením informácie.

Mojim cieľom v tejto práci bude skonštruovať viacvrstvovú doprednú neurónovú sieť (ako počítačový program) a následne ju natrénovať, aby bola schopná predpovedať zvolené chemické vlastnosti látok na základe ich vstupných charakteristík.

V prvej časti práce opíšem nervový systém človeka, práve ktorým bol inšpirovaný vznik neurónových sietí a trochu aj načriem do histórie vývoja neurónových sietí.

V ďalšej časti sa budem venovať teoretickému opisu viacvrstvovej neurónovej siete a aj trénovacích algoritmov, ktoré použijem. V tejto sekcii uvediem aj matematické vzťahy výpočtov, ktoré bude sieť realizovať.

Nasledujúca časť bude venovaná samotnej práci s neurónovou sieťou. Tá zahŕňa výber dát, konštrukciu neurónovej siete a jej správne natrénovanie. Túto sekciu doplním aj o ukážky zdrojového kódu vybraných procedúr programu a grafy zobrazujúce priebeh trénovacieho procesu.

V záverečnej časti práce porovnám efektivitu všetkých zvolených trénovacích algoritmov, pričom budem vychádzať z dvoch kritérii – presnosť predikcie výstupných hodnôt na základe netrénovaných vstupov a časovú náročnosť algoritmu.

2. Základy

2.1 Nervový systém človeka

Nervový systém obsahuje obrovské množstvo buniek. V princípe sa rozdeľuje na centrálny nervový systém (CNS – mozog a miecha) a periférny nervový systém (PNS – gangliá).

Všetky informácie, ktoré CNS dostáva o svete naokolo, sú založené na signáloch vznikajúcich v zmyslových (senzorických) bunkách. Tieto signály sa šíria do príslušných ústredí v CNS cestou aferentných (dostredivých) nervových dráh. Hlavnou funkciou nervového systému je riadiť organizmus. Podkladom tejto funkcie je schopnosť nervového systému spracovávať informácie. Informácie sa v nervovom systéme prenášajú vo forme zmien membránového potenciálu nervových buniek, neurónov.

Neurón je bunka špecializovaná na získavanie, prenos, spracovanie a ukladanie informácií. Informáciu prenáša vo forme elektrického signálu. Časť neurónov prenáša informácie z vonkajšieho a vnútorného prostredia do riadiacich centier CNS, časť regulačnú odpoveď z týchto centier do tkanív a orgánov. Časť neurónov tvorí tieto regulačné centrá, ktoré signály spracúvajú, ukladajú a plánujú odpoveď.

Neuróny sú ohraničené bunkovou membránou. Ide o membránu polopriepustnú. Medzi bunkou a jej okolím existuje elektrický gradient. Nervová bunka sa skladá z tela (lat. soma), z dendritov (z informatického hľadiska predstavujú vstupnú časť) a z jedného axónu (vzruch sa šíri s pomocou axónu k iným bunkám). Axón býva často obalený myelínovou pošvou. Ide o bunkovú membránu tzv. gliovej bunky, ktorá tesne obtáča axón. Na hraniciach medzi dvoma susednými gliovými bunkami sa na axóne nachádzajú neobalené úseky, tzv. Ranvierove zárezy. [3, str.41] Zmeny membránového napätia potom preskakujú myelínovou pošvou izolované úseky a šíria sa priamo od jedného Ranvierovho zárezu k druhému. Neuróny vytvárajú funkčné spojenia v mieste tesného kontaktu axónu jednej bunky s membránou inej bunky. Tieto spojenia sa nazývajú synapsy (ako prvý názov zaviedol Charles Sherrington od latinského slova synapto).

Synapsy delíme podľa toho, že ktoré časti nervových buniek ich tvoria, teda delíme ich na axo-dendritické, axo-somatické a axo-axonálne. Na jednom neuróne sú

stovky, tisícky a na niektorých dokonca mnoho desiatok tisíc synáps. Synapsa sa skladá z troch častí: z membrány presynaptického terminálu (rozšírené zakončenie axónu), synaptickej štrbiny a postsynaptickej membrány.

Miecha vykonáva elementárnu analýzu hmatových podnetov, vnímania teploty a bolesti. Ústredia pre tieto činnosti sa nachádzajú aj na vyšších úrovniach CNS. Okrem toho vykonáva reguláciu niektorých reflexných funkcií pohybového aparátu (*motorika*) a niektorých vegetatívnych reflexov. Obsahuje aferentné a eferentné dráhy. Najvyššou úrovňou miechy je *predĺžená miecha*, ktorá pokračuje štruktúrami nazývanými *Varolov most* a *mesencephalon*. Tieto tri štruktúry sa súborne označujú ako *mozgový kmeň*. Zaisťujú *vegetatívne funkcie* (t.j. vedomím priamo neovládané funkcie zabezpečujúce stabilitu vnútorného prostredia organizmu). Ide napr. o reguláciu srdcovej frekvencie, dýchania, kyslosť vnútorného prostredia, koncentrácie iónov v krvi, atď.

Okrem toho mozgový kmeň zabezpečuje základnú analýzu sluchových, chuťových ako aj hmatových podnetov z oblasti tváre. Podobne riadi motoriku tváre. Je centrom niektorých reflexov (napr. zabezpečujúcich rovnováhu, pohyby očí, rozširovanie a zužovanie zreníc a pod.).

Mozoček hrá významnú úlohu pri modifikácii starých a učení nových pohybových stereotypov. Súčasne porovnáva vykonávaný pohyb s jeho plánom a prípadné odchýlky medzi nimi koriguje.

Mozog je jedna z najzložitejších sústav, vďaka čomu sa už dosť dlho vedú spory o jeho skutočnej zložitosti. [3, str.19] Skladá sa z dvoch polovíc, tzv. *hemisfér*. Väčšina štruktúr patriacich k mozgu je zastúpená dvakrát, v oboch hemisférach. *Talamus* predstavuje veľmi zložitú štruktúru, ktorou prechádzajú takmer všetky aferentné dráhy. Ide o nahromadenie nervových buniek v centrálnych častiach mozgu; takéto zhluky sú typické tým, že na reze majú sivú farbu. Miesta, kde sa nenachádzajú telá nervových buniek, majú bielu farbu. Okolo talamu sa nachádza niekoľko zhlukov neurónov, ktoré sa súborne označujú ako *bazálne gangliá*. Hrajú dôležitú úlohu pri selekcii programu pre pohybový vzorec zvoleného pohybu. Pri ich poruchách vznikajú zložité poruchy motoriky (napr. Parkinsonova choroba). Emotívne podfarbené správanie ovplyvňuje systém skupín nerých buniek nazvaný *limbický systém*. Jeho jedna časť, tzv. *hippocampus*, má zrejme centrálnu úlohu vo fyziológii pamäti. [2, str.11-20]

2.2 História

Za začiatok modernej éry neurónových sietí sa považuje priekopnícka práca psychiatra a neuro-anatóma McCullocha a matematického génia Pittsa z roku 1943. V tejto prevratnej práci opisujú logické výpočty neurónových sietí, ktoré zjednocovali fyziológiu nervovej sústavy a matematickú logiku. Aplikovali sieť zloženú z tzv. formálnych neurónov na symbolickú logiku, na výroky zložené z elementárnych logických operácií (x AND y, x OR y, NOT x). Ich formálny model neurónu sa riadil pravidlom "všetko alebo nič“. Ukázali, že takto vytvorený systém (neurónová sieť'), s dostatočným počtom takýchto jednotiek (neurónov) a ich správnym prepojením (zvolením synáps), dokáže v podstate vypočítat' akúkol'vek vypočítatel'nú funkciu. V prípade, že by takáto sieť' bola neobmedzená, teda nekonečne vel'ká, ukázali ekvivalenciu s turingovým strojom. Táto práca ukázala významné výsledky, na ktorých d'alší mohli stavat' nové poznatky ohl'adne neurónových sietí a umelej inteligencie. McCulloch a Pitts ukázali, že všetky procesy, ktoré sa dajú popísať konečným počtom symbolických výrazov, teda jednoduchá aritmetika, klasifikácia, záznam konečnej množiny dát, rekurzívna aplikácia logických pravidiel a pod., sa dajú realizovať sieťou zloženou z takýchto elementov.

V roku 1948 vyšla Wienerova kniha "Cybernetics“ ktorá opisovala spôsoby kontroly, komunikácie a spracovania štatistických signálov. V druhej edícii sa zameral na učenie a samo-organizáciu.

Ďalším významným míľnikom v obore neurónových sietí bola Hebbova kniha "The organization of bahaviour“ (1949). Hebb v nej navrhol, že účinnosti spojení medzi neurónmi v mozgu sa permanentne menia ako sa jedinec adaptuje a učí nové veci, a to podľa pravidla, že keď má axón bunky A excitačný účinok na bunku B, a opakovane alebo stále sa zúčastňuje na jej aktivácii, v jednej alebo v oboch bunkách prebehne nejaký rastový proces alebo metabolická zmena, takže účinnosť bunky A ako jednej z buniek, ktoré aktivujú B, vzrastie. To má za následok, že skupina neurónov, ktoré sú navzájom pospájané, a ktoré sú synchrónne aktivované, bude mať tendenciu vytvárať tzv. bunečné zoskupenie (angl. *cell assembly*), v ktorom sú neuróny pospájané silnejšie. Podstatná invencia Hebbovej predpovede, odvtedy experimentálne nespočetne veľakrát potvrdenej je, že nová informácia je reprezentovaná distribuovane prostredníctvom zmeny účinností mnohých synaptických spojení.

V roku 1956 matematik John von Neumann, významná postava vo vývoji digitálnych počítačov, vyriešil problém spoľahlivosti McCullochovych-Pittsovych sietí v prípade hardwarového poškodenia. Zaviedol redundanciu - viacero formálnych neurónov vykonáva tú istú prácu. Tak napr. jeden bit informácie (výber medzi 1 a 0) nie je signalizovaný výstupom jedného logického prepínača, ale synchrónnou aktiváciou mnohých neurónov: 1 dostávame, keď viac ako polovica neurónov je aktívnych a 0, keď je aktívnych menej ako polovica neurónov.

V r. 1958 Frank Rosenblatt ukázal, že McCullochove-Pittsove siete s modifikovateľnými synaptickými váhami sa dajú natrénovať tak, aby vedeli rozpoznávať a klasifikovať objekty. Vymyslel pre ne názov “perceptróny”. Hlavná myšlienka jeho trénovacej procedúry je takáto: najskôr zaznamenajme odpoveď každého formálneho neurónu na daný podnet. Ak je odpoveď správna, nemodifikujme váhy. Ak je odpoveď daného neurónu nesprávna, potom modifikujme váhy všetkých aktivovaných vstupných synáps, a to nasledovným spôsobom: ak má byť neurón aktívny a nie je, zväčšime ich, a naopak, ak má byť na výstupe neurónu 0 a nie je, zmenšime ich. Rosenblattova trénovacia procedúra je založená na znalosti toho, ktoré vzory (reprezentujúce objekty) patria do ktorej triedy. Jeho idea modifikácie váh spojení na základe korekcie chýb tvorí základ mnohých algoritmov učenia s pomocou učiteľa, ktoré sa používajú dodnes.

V r. 1969 Minsky a Papert vo svojej knihe "Perceptróny: úvod do výpočtovej geometrie" poukázali na obmedzenia perceptrónov. Ukázali, že tieto siete vôbec nie sú výpočtovo univerzálne a nedokážu riešiť všetky triedy problémov. Hlavne však išlo o to, že perceptróny nedokážu riešiť tzv. lineárne neseparovateľné problémy. Klasickým najjednoduchším príkladom zlyhania je logická funkcia XOR (vylučujúce alebo). V tom čase nebolo známe žiadne pravidlo učenia (t.j. modifikácie synaptických váh) v umelých neurónových sieťach, nevyhnutné na implementáciu výpočtov tohoto typu. Minsky a Papert usúdili, že bude užitočnejšie, keď sa výskum zameria iným smerom.

V roku 1982 J. J. Hopefield navrhol rekurentnú neurónovú sieť. K jednému vstupu viacero výstupov, v závislosti od časového kontextu. Viacvrstvová sieť by mala byť rozšírená o možnosť reprezentovať časový kontext, aby tak mohla na základe predloženého vstupu lepšie rozhodnúť o výstupe.

V roku 1983 Hinton a Sejnowski vymysleli algoritmus učenia pre ľubovoľné, úplne alebo čiastočne rekurentné stochastické siete, ktoré majú symetrické synaptické

spojenia. Takéto siete sa dajú považovať za zovšeobecnenie Hopfieldovej neurónovej siete, tak aby táto sieť obsahovala aj skryté neuróny na extrakciu príznakov. Takisto ako v dopredných sieťach, aj tu sa adaptujú hodnoty váh synaptických spojení v dôsledku pôsobenia vstupov a znalosti o tom, ktoré vstupné vzory sú asociované s ktorými výstupnými konfiguráciami aktivity. Keďže takáto sieť má tú vlastnosť, že pravdepodobnostná distribúcia jej možných stavov (konfigurácií aktivity) je totožná s Boltzmannovym rozdelením, nazvali ju jej autori Boltzmannov stroj (angl. *Boltzmann machine*).

Rok 1986 - Rumelhart, Hinton a Williams zaviedli pravidlo učenia metódou spätného šírenia chýb pre viacvrstvové perceptóny. Metóda učenia pomocou spätného šírenia sa chýb je aplikovateľná len na dopredné troj- a viacvrstvové perceptróny, v ktorých neuróny nie sú spojené každý s každým, ale vzájomne nespojené neuróny v rámci jednej vrstvy posielajú spojenia dopredným spôsobom (jednosmerne) ku všetkým neurónom v ďalšej vrstve. Stavy neurónov v prvej vrstve predstavujú vstup do siete a stavy neurónov v poslednej vrstve predstavujú výstup siete. Jedno- a viacvrstvové perceptróny tiež reprezentujú informáciu redundantne a distribuovane, a fungujú ako obsahom adresovaná pamäť. Sú schopné učiť sa na príkladoch, a samy nájsť príznaky spoločné prvkom (vzorom) patriacim do tej istej triedy. Extrakcia príznakov prebieha vo vnútornej, tzv. skrytej vrstve neurónov (skrytých vrstiev môže byť aj viac ako jedna). Avšak pri učení je potrebná znalosť toho, ktoré prvky (vzory) patria do ktorej triedy. Až po natrénovaní sú tieto siete schopné zovšeobecňovať, t.j. správne klasifikovať nové vzory. Viacvrstvové siete sú veľmi dobré aj na aproximáciu spojitých funkcií. Teoretické odvodenie učenia pomocou algoritmu spätného šírenia sa chýb je založené na minimalizácii objektívnej funkcie, ktorá vyjadruje celkovú chybu, t.j. rozdiel medzi želaným a skutočným výstupom siete.

V roku 1989 Ronald Williams a David Zipser použili metódu gradientovej minimalizácie chybovej funkcie na odvodenie algoritmu spätného šírenia sa chýb v čase na trénovanie rekurentných sietí. V rekurentnej sieti posielajú neuróny v každej vrstve spojenia nielen k nasledujúcim vrstvám, ale aj k predchádzajúcim, a môžu byť pospájané aj medzi sebou. Rekurentné spojenia umožňujú uchovať a vyvolať informáciu, ktorá sa vyskytla v minulosti a použiť ju pre súčasný výpočet. To znamená, že rekurentné siete sú dobré na zapamätávanie a predikciu časových postupností vzorov.

V súčasnosti sa skúmajú dopredné a rekurentné siete, ktorých prvky nemajú sigmoidálnu vstupno-výstupnú prechodovú funkciu, ale tzv. radiálnu bázovú funkciu (angl. *radial basis function*, RBF). Najčastejšie je to aktivačná funkcia v tvare gaussiánu. Takáto prechodová funkcia sa veľmi podobá na tvar recepčných polí v reálnom neurónovom systéme a v mnohých prípadoch RBF siete dosahujú lepšie výsledky ako siete s prvkami, ktoré majú sigmoidálnu prechodovú funkciu. [2, str.32-38]

3. Viacvrstvové dopredné neurónové siete

Viacvrstvová neurónová sieť zložená z perceptrónov (anglicky Multilayer Perceptron) je najbežnejší a najčastejšie používaný typ neurónových sietí. Prívlastok „dopredná“ (anglicky Feedforward) sa používa na udanie smeru šírenia informácie. Na rozdiel od jednoduchých neurónových sietí, ktoré dokážu klasifikovať len lineárne separovateľné problémy, viacvrstvové neurónové siete sú univerzálnym aproximátorom, t.j. sú schopné s požadovanou presnosťou aproximovať ľubovoľnú spojitú reálnu funkciu. [2, str.70]

3.1 Neurón

V oblasti umelých neurónových sietí pojmom neurón označujeme jednoduché elementy, z ktorých sú tieto neurónové siete zložené. Perceptrón je neurón, ktorý uskutočňuje jednoduchý matematický výpočet. Obsahuje n vstupov x_1 až x_n, ktoré sú na neho prepojené prostredníctvom váhových prepojení s koeficientami w_1 až w_n. Koeficient, čiže váha prepojenia určuje, ako výrazne daný vstup ovplyvňuje výslednú aktivitu neurónu. Výpočet, ktorý neurón realizuje, sa teda dá zapísať ako:

$$a = f\left(\sum_{i=1}^{n} w_i x_i + \Theta\right) \qquad (3.1)$$

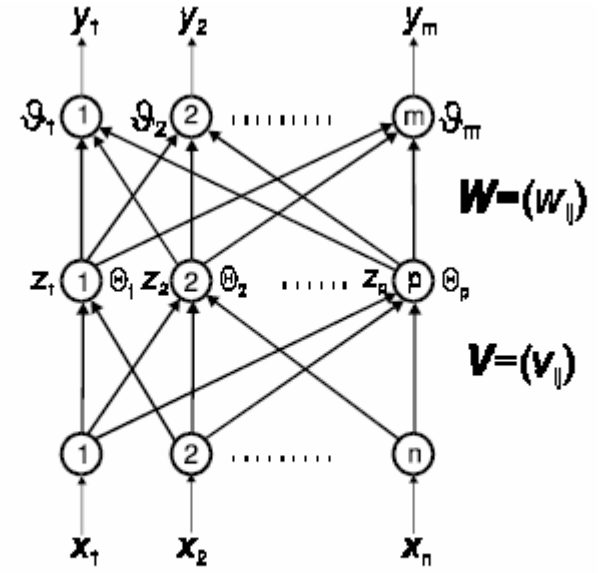
kde a označuje výstupnú aktivitu neurónu a Θ je prahový koeficient neurónu. Prahový koeficient môže byť podľa potreby vždy nastavený na konštantu 1 alebo sa môže vždy upravovať spolu s váhovými koeficientami (vtedy ho môžeme považovať za váhu neurónu). Tento prah môže byť aj záporný. Funkcia f je tzv. aktivačná alebo prechodová funkcia. Veľmi často používaná aktivačná funkcia je logistická sigmoida alebo iná sigmoidálna funkcia. Vzťah pre výpočet funkčnej hodnoty logistickej sigmoidy je nasledovný:

$$f(x) = \frac{1}{1 + e^{-x}} \qquad (3.2)$$

3.2 Viacvrstvová neurónová sieť

Napriek faktu, že vo všeobecnosti je možné zostrojiť doprednú neurónovú sieť so skoro ľubovoľne poprepájanými neurónmi, v praxi sa obyčajne používajú neurónové

siete s neurónmi organizovanými do vrstiev. Najčastejšie používaná architektúra je trojvrstvová neurónová sieť znázornená na obrázku 3.1. Vstupné neuróny neuskutočňujú žiadny výpočet, iba poskytujú vstupnú informáciu do siete (a preto sa aj niekedy táto architektúra označuje ako dvojvrstvová sieť, keďže obsahuje iba dve vrstvy „skutočných“ neurónov). Zo vstupnej vrstvy informácia postupuje na skrytú vrstvu obsahujúcu skryté neuróny. Najčastejšie je každý skrytý neurón prostredníctvom vstupných váh prepojený s každým vstupným neurónom. Z neurónov skrytej vrstvy informácia postupuje ďalej na výstupné neuróny na výstupnej vrstve. Znova, každý výstupný neurón je prostedníctvom výstupných váh prepojený s každým skrytým neurónom. Vo všeobecnosti by dopredná neurónová sieť mohla mať aj vyšší počet skrytých vrstiev, ktoré by boli medzi sebou poprepájané príslušnými váhami, avšak takáto sieť neprináša lepšie výsledky ako neurónová sieť s jednou vrstvou skrytých neurónov, pretože ďalšie skryté vrstvy je možné nahradiť vyšším počtom neurónov v prvej skrytej vrstve.



Obr. 3.1 Trojvrstvová neurónová sieť zložená z n vstupných, p skrytých a m výstupných neurónov

Neurónová sieť z obrázku 3.1 má *n* vstupných neurónov, na ktoré sa pred zahájením výpočtu umiestni vstupná informácia – n-tica reálnych čísiel $(x_1,x_2,...,x_n)$. Aktivity skrytých neurónov – p-tica reálnych čísiel $(z_1,z_2,...,z_p)$, sú vypočítané ako:

$$z_j = f\left(\sum_{i=1}^{n} v_{ji} x_i + \Theta_j\right) \qquad (3.3)$$

pre j = 1,2,...,*p*

Kde f je aktivačná funkcia (3.2) a Θ_j je prahový koeficient (váha) skrytého neurónu. Aktivity výstupných neurónov – m-tica reálnych čísiel (y_1,y_2,...,y_m), sú vypočítané ako:

$$y_k = f\left(\sum_{j=1}^{p} w_{kj} z_j + \vartheta_k\right) \qquad (3.4)$$

pre k = 1,2,...,*m*

Kde f je aktivačná funkcia (3.2) a θ_k je prahový koeficient (váha) výstupného neurónu. Tento výpočet predpokladá, že prahové koeficienty sú váhami neurónov, ktoré sa upravujú pri adaptácii siete.

3.3 Trénovanie neurónových sietí

Významnou vlastnosťou neurónových sietí je možnosť ich natrénovať na základe dopredu pripravených vstupno-výstupných hodnôt. Trénovacia množina vytvoríme z typických vstupov a z k nim prislúchajúcich očakávaných výstupov. Po správnom natrénovaní sa od neurónovej siete očakáva, že dokáže zovšeobecnovať, čiže správne predpovedať výstupy pre vstupy, ktoré trénovacia množina neobsahovala.

Typický postup trénovania neurónovej siete je nasledovný:

1. Príprava trénovacej a testovacej množiny

Riešená úloha je transformovaná do podoby, ktorá umožňuje jej riešenie pomocou neurónovej siete. Podľa typu úlohy je potrebné rozhodnúť, čím budú tvorené vstupy do neurónovej siete a čo budeme očakávať na výstupe neurónovej siete. Výstupom z tejto fázy je trénovacia a väčšinou aj testovacia množina, ktorá pozostáva z viacerých vstupno-výstupných vzorov (x,d), kde *x* je vstupný vektor a *d* je k nemu

prislúchajúci vektor očakávaných výstupných hodnôt. Takýchto vzorov obsahuje trénovacia aj testovacia množina dostatočný počet, aby bolo možné sieť dostatočne dobre natrénovať.

2. Inicializácia neurónovej siete

Inicializácia spočíva v samotnom vytvorení neurónovej siete. Hodnoty váhových prepojení sú generované náhodne na malé hodnoty, napr. z intervalu (0,0.5). Počet vstupných neurónov siete je daný rozmerom vstupného vektora a počet výstupných neurónov je rovný rozmeru vektora požadovaných výstupných hodnôt. Počet skrytých neurónov sa volí experimentálne, typicky sú to jednotky až desiatky neurónov usporiadané v jedinej vrstve. Dobrý štart pri hľadaní optimálneho počtu skrytých neurónov je obyčajne 1,5-násobok počtu vstupných neurónov.

3. Trénovanie neurónovej siete

Dopredným šírením informácie sa vypočítajú skryté a potom aj výstupné aktivity siete. Po doprednom šírení informácie cez jednotlivé vrstvy siete očakávame na výstupe požadované hodnoty $d=(d_1,d_2,...,d_K)$, ktoré sa zvyčajne od sieťou vypočítaných $y=(y_1,y_2,...,y_K)$ hodnôt líšia. Túto chybu, ktorá vznikla na výstupe, sa snažíme minimalizovať. Chybová (účelová) funkcia v danom kroku trénovacieho algoritmu definovaná ako

$$\boldsymbol{E} = \frac{1}{2}\sum_{\boldsymbol{k}=1}^{\boldsymbol{K}} (\boldsymbol{d_k} - \boldsymbol{y_k})^2 \qquad (3.5)$$

Spôsob, akým minimalizujeme účelovú funkciu, záleží od použitého trénovacieho algoritmu.

4. Ukončenie trénovania a otestovanie neurónovej siete

Algoritmus ukončíme, ak uplynul stanovený počet iterácií, prípadne keď v danom kroku iterácie je výsledná hodnota účelovej funkcie po prechode cez celú trénovaciu alebo testovaciu množinu uspokojivo malá. Ak chyba ešte neklesla pod požadovanú úroveň, pokračujeme v trénovaní. [1]

3.3.1 Trénovanie metódou Backpropagation

Najznámejším ako aj najčastejšie používaným algoritmom na trénovanie neurónových sietí je algoritmus spätného šírenia chybového signálu (anglicky Backpropagation). Umožňuje vhodným spôsobom organizovať výpočet derivácií, ktoré sú potrebné na zmenu váh pomocou gradientovej metódy najprudšieho spádu. Tieto derivácie sú vypočítavané postupne od neurónov na vyšších vrstvách smerom k neurónom na nižších vrstvách. [1, str.41]

Účelovú funkciu chceme minimalizovať metódou najprudšieho spádu, a teda váhy potrebujeme zmeniť pomocou vzťahov:

- váhy výstupných neurónov:

$$\vartheta_i(t+1) = \vartheta_i(t) - \lambda \frac{\partial E}{\partial \vartheta_i} + \mu(\vartheta_i(t) - \vartheta_i(t-1)) \tag{3.6}$$

pre $i = 1,2,...,m$

- výstupné váhy (medzi skrytou a výstupnou vrstvou):

$$w_{ij}(t+1) = w_{ij}(t) - \lambda \frac{\partial E}{\partial w_{ij}} + \mu(w_{ij}(t) - w_{ij}(t-1)) \tag{3.7}$$

pre $i = 1,2,...,m$ a $j = 1,2,...,p$

- váhy skrytých neurónov:

$$\Theta_i(t+1) = \Theta_i(t) - \lambda \frac{\partial E}{\partial \Theta_i} + \mu(\Theta_i(t) - \Theta_i(t-1)) \tag{3.8}$$

pre $i = 1,2,...,p$

- vstupné váhy (medzi vstupnou a skrytou vrstvou):

$$\boldsymbol{v_{ij}(t+1) = v_{ij}(t) - \lambda \frac{\partial E}{\partial v_{ij}} + \mu(v_{ij}(t) - v_{ij}(t-1))} \tag{3.9}$$

pre $i = 1,2,...,p$ a $j = 1,2,...,n$

kde λ je rýchlosť učenia, ktorá určuje mieru zmeny váhy, μ je momentum, ktoré pôsobí ako istá zotrvačnosť pri menení váh a pomáha zabrániť uviaznutiu v lokálnych minimách účelovej funkcie, *t* popisuje iteračný krok. Rýchlosť učenia λ, resp. momentum μ sa obvykle položia rovné malému kladnému číslu, napr. 0.1. Rýchlosť učenia je často vhodné aj dynamicky meniť v závislosti od hodnoty účelovej funkcie. Ak sa jej hodnota zväčší, parameter λ zmenšíme a naopak, keď sa účelová funkcia monotónne zmenšuje, parameter λ zväčšíme. [2, str.111-116]

Derivácie vo vyššie uvedených vzorcoch vypočítame pomocou vzťahov:

$$\boldsymbol{\frac{\partial E}{\partial \vartheta_i} = y_i(1 - y_i)(y_i - d_i)} \tag{3.10}$$

pre $i = 1,2,...,m$

$$\boldsymbol{\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial \vartheta_i} z_j} \tag{3.11}$$

pre $i = 1,2,...,m$ a $j = 1,2,...,p$

$$\boldsymbol{\frac{\partial E}{\partial \Theta_i} = z_i(1 - z_i)\sum_{j=1}^{m} \frac{\partial E}{\partial \vartheta_j} w_{ji}} \tag{3.12}$$

pre $i = 1,2,...,p$

$$\frac{\partial \boldsymbol{E}}{\partial \boldsymbol{v}_{ij}} = \frac{\partial \boldsymbol{E}}{\partial \Theta_i} \boldsymbol{x}_j \tag{3.13}$$

pre $i = 1,2,...,p$ a $j = 1,2,...,n$

3.3.2 Trénovanie genetickým algoritmom

Genetický algoritmus vychádza zo všeobecných predstáv Darwinovej teórie prirodzeného výberu a je vhodný na hľadanie globálneho minima zložitých úloh. Pri optimalizácii váhovych koeficientov neurónovej siete to nie je najtypickejšia metóda, pretože klasická metóda Backpropagation býva dostatočne presná a výpočtovo menej náročná. Pri tejto metóde sa snažíme minimalizovať účelovú funkciu náhodnými zmenami prvkov v populácii. Tieto zmeny zabezpečujú kríženie a mutácia. [2, str.255-256]

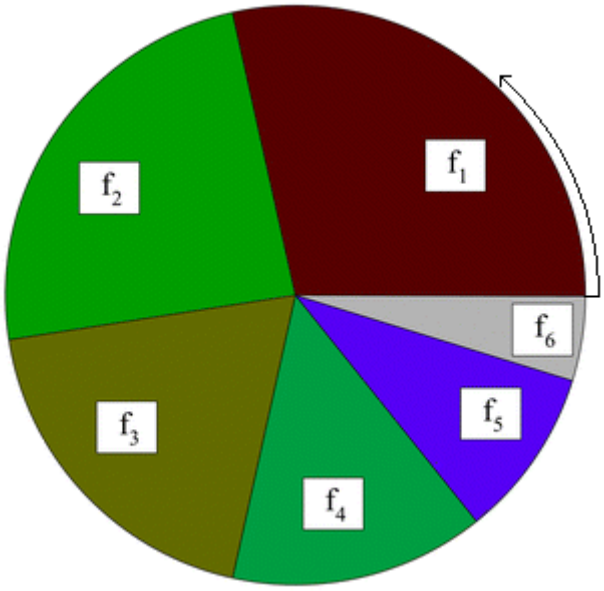
Chceme minimalizovať účelovú funkciu jedného člena populácie. Člen populácie obsahuje matice všetkých váh v neurónovej sieti. Váhy sa upravujú nasledovne:
1. Potrebujeme ohodnotiť jednotlivé členy populácie silou (fitness). Výpočet hodnoty sily dostaneme napr. z prevrátených hodnôt účelových funkcií jednotlivých členov populácie podľa nasledujúceho vzťahu:

$$F_i = \frac{\frac{1}{E_i}}{\sum_{j=1}^{p} \frac{1}{E_j}} \tag{3.14}$$

pre $j = 1,2,...,p$
$i \in \{1,2,..,p\}$

Kde *p* označuje počet členov populácie, *F* je hodnota sily daného členu a *E* je hodnota účelovej funkcie daného členu populácie.. Zo vzťahu (3.14) vyplýva, že hodnota sily môže byť z intervalu $(0;1\rangle$.

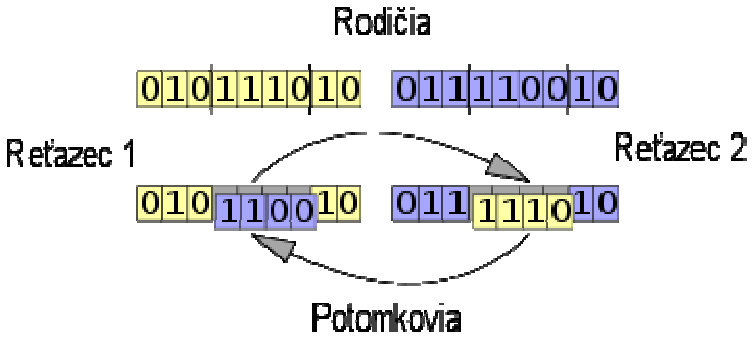
2. Náhodne vyberieme dva členy populácie, pričom pravdepodobnosť výberu je priamo úmerna sile člena populácie. Tento proces sa nazýva ruleta.



Obr. 3.2 Schématické znázornenie rulety

Schéma na obrázku 3.2 vysvetľuje princíp výberu pomocou rulety. Na myslenú kružnicu s obvodom veľkosti 1 sa postupne od štartovacieho bodu s hodnotou 0 nanesú hodnoty sily jednotlivých členov populácie (ktorých suma je samozrejme rovná 1). Každý člen populácie tak získa určitý podiel obvodu na kružnici, ktorý je priamo úmerný veľkosti jeho sily. Šípka na schéme zobrazuje smer, akým rastú hodnoty od 0 smerom k 1 (dá sa to predstaviť ako smer točenia rulety). Následne sa vygeneruje náhodné číslo od 0 po 1 a podľa toho, do ktorej časti rulety vygenerované číslo patrí, ten člen populácie je vybraný do reprodukčného procesu (podľa schémy na obrázku by bol vybraný prvý člen populácie). Postup opakujeme dvakrát, aby sme dostali dvojicu, ktorú budeme krížiť.

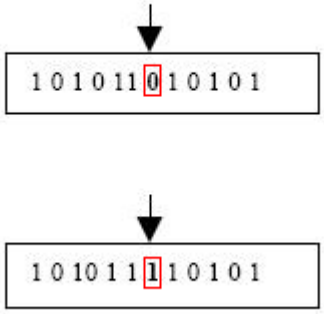
3. Nastane kríženie – členy populácie si navzájom vymenia svoje časti, pričom počet vymenených komponentov je náhodne zvolený.



Obr. 3.3 Schéma 2-bodového kríženia binárnych reťazcov

Kríženie je možné realizovať mnohými spôsobmi, na obrázku 3.3 je znázornené 2-bodové kríženie, kedy sa vymenia komponenty medzi dvoma náhodne zvolenými rezmi, ďalej možno použiť 1-bodové, kedy sa vymenia všetky komponenty pred alebo za rezom. Tieto 2 typy patria medzi najbežnejšie formy kríženia, ale v prípade potreby je možné praktizovať aj iné spôsoby výberu vymieňaných častí.

4. Skrížené členy populácie sa podrobia mutácii – v náhodne vybraných komponentoch nastane zmena. V prípade, že pracujeme s váhami v reálnom tvare, ide o nejakú malú zmenu číselnej hodnoty, ak pracujeme s váhami v binárnom tvare, ide o zmenu jednotky na nulu, resp. nuly na jednotku.



Obr. 3.4 Schéma mutácie vybraného komponentu v binárnom reťazci

Obrázok 3.4 zobrazuje mutáciu jedného náhodne zvoleného komponentu v binárnom reťazci. Pravdepodobnosť, že daný komponent zmutuje, sa obvykle volí dosť nízka, v opačnom prípade by mohol proces mutácie pôsobiť pri trénovaní siete kontraproduktívne.

5. Postup sa od bodu 2 opakuje zvyčajne dovtedy, kým nedostaneme celú populáciu nových členov.
6. Uplynula jedna generácia a celý postup sa opakuje znovu.

Spôsobov, akými možno vypočítať silu členov populácie, ako krížiť alebo mutovať existuje mnoho, ja som uviedol postup výhodný pre trojvrstvovú architektúru doprednej neurónovej siete.

4. Aplikácia neurónovej siete – predikcia vlastností chemických látok

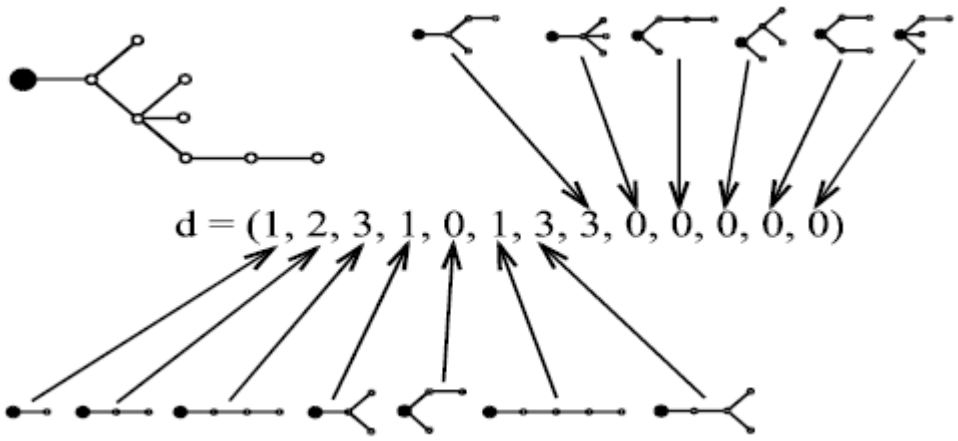
V tejto časti bakalárskej práce sa budem venovať praktickému využitiu neurónovej siete. Hoci existuje dosť dostupného softvéru zameraného na dopredné neurónové siete, táto práca vyžadovala realizáciu vlastného programu. Preto som celú trojvrstvovú neurónovú sieť naprogramoval v jazyku C. Program umožňuje nastaviť celý rad parametrov siete a obsahuje tri trénovacie algoritmy:

1. Backpropagation
2. Genetický algoritmus
3. Backpropagation využívajúci genetický algoritmus na počiatočné generovanie váh

Na vybraných dátach natrénujem neurónovú sieť všetkými troma algoritmami a potom porovnám ich časovú náročnosť ako aj schopnosť predikovať výstupné hodnoty. V tejto časti budem prezentovať aj časti zdrojového kódu svojho programu napísané v ja ktorý umožňuje ich takmer okamžitý zápis v bežnom programovacom jazyku. Zdrojové kódy môjho programu sa nachádzajú v samostatnej prílohe na CD (sources.rar).

4.1 Voľba dát, príprava trénovacej a testovacej množiny

Do neurónovej siete som zvolil dáta z práce [5], kde sa autori zaoberali konštrukciou neurónovej siete, ktorú by bolo možné použiť na predikciu ^{13}C NMR (jadrová magnetická rezonancia) chemických posunov (shiftov) nasýtených acyklických uhľovodíkov – alkánov. Chemický posun je lokálna vlastnosť uhlíkových atómov, preto daný atóm uhlíka býva popísaný jeho najbližším okolím. Tento popis okolia je tvorený deskriptorom ***d***, čo je vlastne vektor obsahujúci nezáporné celé čísla, ktorých počet je rovný počtu dopredu zvolených fragmentov v alkáne. Tieto čísla v deskriptore opisujú frekvenciu výskytu daného fragmentu v alkáne so špecifikovaným uhlíkom (obrázok 4.1).



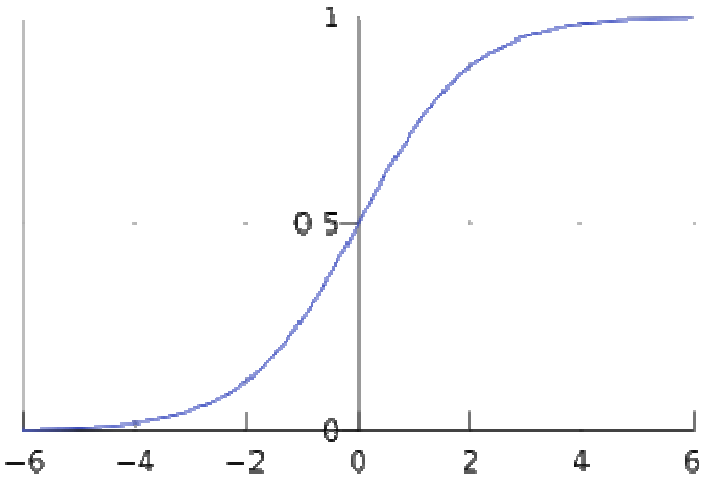
Obr. 4.1 Diagram popisu molekuly acyklického nenasýteného uhľovodíka pomocou deskriptora ***d***

Zvolené dáta obsahovali deskriptory s trinástimi číslami (skúmala sa teda prítomnosť trinástich rôznych fragmentov v alkáne). Vstupný vektor do neurónovej siete bude teda obsahovať 13 hodnôt. Vektor očakávaných výstupov bude obsahovať len jednu hodnotu, ide o experimentálne nameranú hodnotu chemického posunu pre daný deskriptor. [4] Tieto hodnoty boli tiež dostupné v spomínaných dátach. Mojím cieľom bude správne natrénovať neurónovú sieť, ktorá potom bude schopná predikovať hodnotu chemického posunu aj z deskriptorov, ktoré nemala k dispozícii pri procese trénovania. Textové súbory obsahujúce spomínané dáta sa nachádzajú v samostanej prílohe, ktorá sa nachádza na CD (data.rar).

Z týchto dát teraz potrebujeme zostaviť trénovaciu a testovaciu množinu. Dáta obsahujú 326 vstupných vektorov a očakávaných výstupov. Ich vhodné rozdelenie do trénovacej a testovacej množiny sa zisťuje experimentálne, v podstate ide o jeden z voliteľných parametrov siete. Ja som zvolil pomerne jednoduchú a bežnú metódu rozdelenia dát – prvých 250 vektorov a výstupov som umiestnil do trénovacej množiny a zvyšných 76 do testovacej množiny.

Použitá neurónová sieť používa ako aktivačnú funkciu už vyššie spomenutú (3.2) logistickú sigmoidu:

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



Obr. 4.2 Graf logistickej sigmoidy

Ako vidno z grafu na obrázku 4.2, táto funkcia má obor hodnôt v intervale (0;1), preto aj očakávané výstupy musia ležať v tomto intervale. Výstupy v použitých dátach túto podmienku nespĺňajú, preto ich potrebujeme ešte pred zahájením trénovacieho procesu siete normovať. Sieť je naprogramovaná tak, že dokáže rozpoznať, či sú očakávané výstupy z požadovaného intervalu a ak nie, bude ich normovať podľa nasledujúceho algoritmu:

1. Spomedzi všetkých výstupov nájde globálne minimum MIN a globálne maximum MAX.
2. Všetky hodnoty stlačí do intervalu (0;1) pomocou nasledujúceho vzťahu:

$$d_{norm} = \frac{d - MIN + a}{MAX - MIN + b} \tag{4.1}$$

Kde d je požadovaný výstup siete (načítaný z dát), a,b sú malé racionálne čísla, pričom platí, že a,b>0 ; a<b. Tieto konštanty slúžia na to, aby všetky hodnoty (vrátane extrémov) boli z intervalu (0;1), ktorý je otvorený a preto nemôže obsahovať ani hodnoty 0 a 1.

4.2 Inicializácia neurónovej siete

V podstate ide o vytvorenie samotnej siete a o generáciu počiatočných váh. Sieť je naprogramovaná všeobecne, takže počet neurónov na jednotlivých vrstvách si môžeme voliť. Keďže vstupné vektory v dátach obsahujú 13 celých čísel, nastavíme 13 vstupných neurónov, počet výstupných neurónov bude 1, pretože ku každému

vstupnému vektoru prislúcha jedna výstupná hodnota. Počet skrytých neurónov sa zisťuje experimentálne a pravdepodobne ho budeme ešte meniť. Ako štartovací bod môžeme dať 1,5-násobok počtu vstupných neurónov, čiže v našom prípade by to bolo 19 alebo 20 skrytých neurónov. Ďalej zadáme mohutnosť trénovacej množiny (250) a celkový počet vstupov, resp. očakávaných výstupov (v podstate trénovacia a testovacia množina spolu) – 326.

Ďalej sieť umožňuje nastaviť ďalšie parametre – ich počet sa líši v závislosti od použitého trénovacieho algoritmu:

1. Backpropagation – nastavujeme ešte počet iterácii, počiatočnú rýchlosť učenia a momentum.
2. Genetický algoritmus – tu treba nastaviť počet generácii, počet členov populácie, pravdepodobnosť, pri ktorej nastane kríženie v danej generácii a pravdepodobnosť, pri ktorej nastane mutácia.
3. Backpropagation vylepšený o generovanie váh cez genetický algoritmus – keďže ide v podstate o spojenie predchádzajúcich dvoch algoritmov, využíva všetky parametre spomenuté pri predchádzajúcich metódach.

Po nastavení parametrov je sieť pripravená na trénovanie, ale ešte pred tým vygeneruje počiatočné váhy. Pri prvých dvoch algoritmoch ide o náhodné generovanie. V prípade Backpropagation sa generujú reálne čísla z intervalu (0;x), kde za x volíme nejakú malú hodnotu (napr. 0.02 v mojom prípade). Pri genetickom algorimte sa generujú binárne reťazce, ktoré sú náhodne vyplnené nulami a jednotkami. Dĺžka reťazca závisí od zadaného počtu dekadických miest a je vypočítaná podľa nasledujúceho vzťahu:

$$k = \left\lceil \frac{\ln 10^n}{\ln 2} \right\rceil \tag{4.2}$$

Kde n je počet požadovaných dekadických miest a hranaté zátvorky znamenajú najbližšiu väčšiu celočíselnú hodnotu výrazu v nich.

Tretí algoritmus využíva na generovanie váh celý genetický algoritmus. V tomto prípade zvyčajne nastavíme počet generácii na malý počet a po dokončení genetického algoritmu sa sieť trénuje metódou Backpropagation. Výhoda tohto

postupu spočíva v tom, že počiatočné váhy nie sú úplne náhodné, ale už čiastočne optimalizované.

4.3 Trénovanie siete

Najskôr sa uskutoční prechod sieťou s počiatočnými váhami, z čoho dostaneme aktivity neurónov a hodnoty účelovej funkcie v trénovacej aj testovacej množine. Potom sa upravujú váhy a následne znova nastane prechod sieťou už s upravenými váhami. Tento postup sa potom opakuje dovtedy, kým neuplynie predpísaný počet iterácii, resp. generácii. Spôsob, akým sú upravované váhy, závisí od použitého trénovacieho algoritmu. Sieť si v každom iteračnom kroku overuje, či sú aktuálne váhy lepšie ako doposiaľ najlepšie nájdené váhy. Ak áno, prepíše ich novými, v prípade, že nie, ponechá predchádzajúce.

```
//aktivity
for(k=0;k<tstr;k++){
  for(i=0;i<sk;i++){
     aktiv=t[i];
     for(j=0;j<vs;j++){
        aktiv+=(v[i][j]*x[k][j]);
     }
     z[k][i]=af(aktiv);
  }
}

mtrain=0.0;
mtest=0.0;
for(k=0;k<tstr;k++){
  stvorec=0.0;
  for(i=0;i<vy;i++){
     aktiv=u[i];
     for(j=0;j<sk;j++){
        aktiv+=(w[i][j]*z[k][j]);
     }
     y[k][i]=af(aktiv);
     stvorec+=pow((d[k][i]-y[k][i]),2.0);
  }
if(k<train){
  mtrain+=(stvorec*0.5);
}
else{
    mtest+=(stvorec*0.5);
}

}
```

Obr. 4.3 C kód procedúry aktivity

Na obrázku 4.3 vidíme algoritmizáciu dopredného prechodu siete, najskôr sú vypočítané aktivity skrytej vrstvy *z*, následne aj výstupnej vrstvy *y*. Nakoniec je vypočítaná hodnota účelovej funkcie podľa vzorca (3.5), kde *d* sú požadované výstupy pre trénovaciu aj testovaciu množinu. Aktivačná funkcia (logistická sigmoida) je označená v pseudokóde ako funkcia *af*, *train* označuje mohutnosť trénovacej množiny, *tstr* mohutnosť trénovacej a testovacej množiny spolu a *pow* je funkcia na výpočet n-tej mocniny importovaná prostredníctvom hlavičkového súboru math.h. Hodnota účelovej funkcie trénovacej množiny je uložená do premennej *mtrain*, hodnota účelovej funkcie testovacej množiny do *mtest*. V prípade genetického algoritmu táto procedúra počas jednej generácie prebehne pre každého člena populácie zvlášť. Prahové koeficienty sú označené ako *t* (skrytá vrstva) a *u* (výstupná vrstva), váhové koeficienty ako *v* (medzi vstupnou a skrytou vrstvou) a *w* (medzi skrytou a výstupnou vrstvou).

Po každom prechode sieťou sa testuje, či sú aktuálne váhy lepšie, ako posledné najlepšie, ktoré má sieť zapísané. V prípade prvého prechodu sieťou s počiatočnými váhami sú tieto váhy automaticky zapísané ako optimálne váhy. Počas ďalších iteračných krokov sa overuje, či pri daných váhach je hodnota účelovej funkcie trénovacej množiny menšia ako pri posledných optimálnych váhach. V prípade genetického algoritmu berieme do úvahy toho člena populácie, ktorý má najmenšiu hodnotu účelovej funkcie trénovacej množiny a tú porovnávame s poslednou optimálnou hodnotou. Ak je táto podmienka splnená, ešte sa overuje, či aj hodnota účelovej funkcie testovacej množiny je menšia ako posledná optimálna. Ak áno, zapíšu sa nové optimálne váhy. Keďže hodnota účelovej funkcie trénovacej funkcie obyčajne stále klesá a účelová funkcia testovacej množiny zvykne vykazovať globálne minimum, podmienka porovnávajúca hodnoty v testovacej množine je dôležitejšia. Sieť totiž dokáže najlepšie predikovať výstupné hodnoty práve vtedy, keď je natrénovaná na váhy, s ktorými dosahuje toto globálne minimum v trénovacej množine.

Najdôležitejšou časťou trénovacieho procesu siete je úprava váh. Táto časť programu sa úplne odlišuje v závislosti od použitého trénovacieho algoritmu. V prípade Backpropagation je táto časť realizovaná výpočtom gradientu účelovej funkcie. Tento gradient pozostáva z viacerých komponentov – sú to parciálne derivácie účelovej funkcie vzhľadom k prahovým koeficientom neurónov výstupnej a skrytej vrstvy ako aj vzhľadom na váhové koeficienty medzi vrstvami – vzťahy

(3.10), (3.11), (3.12) a (3.13). Tento výpočet sa realizuje v smere zhora nadol (t.j. v opačnom smere ako pri prechode sieťou). Celkový gradient dostaneme ako sumu gradientov všetkých prvkov trénovacej množiny. Následne sa ešte v procese adaptácie upravia váhy pomocou vzťahov (3.6), (3.7), (3.8) a (3.9). Tieto vzťahy práve využívajú vypočítaný gradient. Po úprave váh je iteračný krok ukončený. Realizáciu adaptačnej časti neurónovej siete v programe zobrazuje kód na obrázkoch 4.4 a 4.5. V prvej fáze procesu (obr. 4.4) sa počítajú gradienty prahových, resp. váhových koeficientov (premenné *grad_u*, *grad_w*, *grad_t* a *grad_v*). Výsledné hodnoty gradientov prahových a váhových koeficientov sú sumy gradientov všetkých prvkov trénovacej množiny (premenné s názvami *grad_total_u*, *grad_total_w*, *grad_total_t* a *grad_total _v*). V druhej fáze (obr. 4.5) sa upravujú váhové a prahové koeficienty. Rýchlosť učenia je označená ako *lambda*, momentum ako *mi*. Premenná *speed* slúži na ukladanie hodnoty účelovej funkcie z predchádzajúceho iteračného kroku (iteračný krok je vyjadrený premennou *it*) z dôvodu implementácie dynamickej zmeny rýchlosti učenia počas trénovacieho procesu. Ak hodnota trénovacej množiny v nasledujúcom iteračnom kroku stúpne, rýchlosť učenia sa zmenší, v opačnom prípade sa rýchlosť učenia zväčší.

```
//gradienty

//vynulovanie totalneho gradientu
for(i=0;i<sk;i++){
   for(j=0;j<vs;j++){
      grad_total_v[i][j]=0.0;
   }
}
for(i=0;i<vy;i++){
   for(j=0;j<sk;j++){
      grad_total_w[i][j]=0.0;
   }
}
for(i=0;i<sk;i++){
   grad_total_t[i]=0.0;
}
for(i=0;i<vy;i++){
   grad_total_u[i]=0.0;
}

//backpropagation
for(k=0;k<train;k++){
   for(i=0;i<vy;i++){
      grad_u[i]=y[k][i]*(1-y[k][i])*(y[k][i]-d[k][i]);
      grad_total_u[i]+=grad_u[i];
   }
   for(i=0;i<vy;i++){
      for(j=0;j<sk;j++){
         grad_w[i][j]=grad_u[i]*z[k][j];
         grad_total_w[i][j]+=grad_w[i][j];
      }
   }
   for(i=0;i<sk;i++){
      aux=0.0;
      for(j=0;j<vy;j++){
         aux+=(grad_u[j]*w[j][i]);
      }
      grad_t[i]=z[k][i]*(1-z[k][i])*aux;
      grad_total_t[i]+=grad_t[i];
   }
   for(i=0;i<sk;i++){
      for(j=0;j<vs;j++){
         grad_v[i][j]=grad_t[i]*x[k][j];
         grad_total_v[i][j]+=grad_v[i][j];
      }
   }
}
```

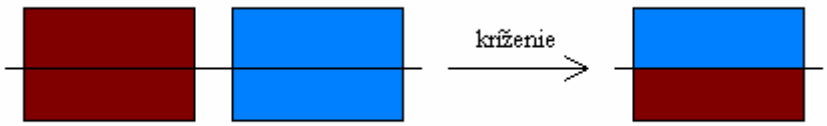
Obr. 4.4 C kód procedúry gradienty

```
//adaptacia
delta=0.0;
if((it>1) && (speed-mtrain)<0.){lambda/=2.0;}
if((it>1) && (speed-mtrain)>0.){lambda*=1.5;}
for(i=0;i<vy;i++){
   delta=-1*lambda*grad_total_u[i]+mi*delta_u[i];
   u[i]+=delta;
   delta_u[i]=delta;
}
for(i=0;i<vy;i++){
   for(j=0;j<sk;j++){
     delta=-1*lambda*grad_total_w[i][j]+mi*delta_w[i][j];
     w[i][j]+=delta;
     delta_w[i][j]=delta;
   }
}
for(i=0;i<sk;i++){
   delta=-1*lambda*grad_total_t[i]+mi*delta_t[i];
   t[i]+=delta;
   delta_t[i]=delta;
}
for(i=0;i<sk;i++){
   for(j=0;j<vs;j++){
     delta=-1*lambda*grad_total_v[i][j]+mi*delta_v[i][j];
     v[i][j]+=delta;
     delta_v[i][j]=delta;
   }
}
speed=mtrain;
```

Obr. 4.5 C kód procedúry adaptacia

Genetický algoritmus upravuje váhy pomocou kríženia a mutácii. V mojom programe som pracoval s váhami v binárnom tvare, ktorý tieto operácie uľahčuje. Matice váh a prahových koeficientov som previedol do binárneho tvaru, následne aplikoval procedúry kríženia a mutácie, na záver generácie som ešte využil tzv. elitizmus, ktorý zaručí, že hodnota účelovej funkcie trénovacej množiny nestúpne takým spôsobom, že jedného náhodne vybraného člena populácie nahradí optimálnym členom (pod členom rozumieme matice váh a prahových koeficientov). Proces kríženia je možné realizovať mnohými spôsobmi, v mojom prípade je postup nasledovný – najskôr sa vypočíta sila jednotlivých členov populácie podľa vzťahu (3.14) a následne sa pomocou rulety vyberú náhodné dva členy, pričom pravdepodobnosť výberu daného člena populácie je priamo úmerná jeho sile. Následne sa vygeneruje náhodné číslo, ktoré reprezentuje číslo riadku matice váh prvého z dvojice vybraných členov a všetky riadky pred týmto riadkom nahradí

hodnotami váh z druhého člena. Tento proces sa opakuje dvakrát, pretože každý člen populácie obsahuje 2 matice – prvá pozostáva z váh medzi vstupnou a skrytou vrstvou a prahových koeficientov skrytej vrstvy, druhá obsahuje váhy medzi skrytou a výstupnou vrstvou a prahové koeficienty výstupnej vrstvy. Vznikne tak jeden nový člen populácie. Počet opakovaní tejto procedúry je rovný počtu členov populácie, takže po skončení kríženia dostaneme novú populáciu, ktorá nahradila tú starú. Na obrázku 4.6 je ilustrovaná schéma zobrazujúca popísaný spôsob kríženia v programe. Následne sa na každého člena novej populácie uplatní mutácia. Táto procedúra spočíva v tom, že pre každý bit matíc člena populácie (váhy sú v binárnom tvare) sa vygeneruje náhodné číslo od 0 po 1 a ak je menšie ako nastavená pravdepodobnosť mutácie, zamení v danom bite 0 za 1, resp. 1 za 0. Po uplatnení vyššie spomínaného elitizmu sa generácia končí, váhy sa prevedú na reálny tvar (kvôli výpočtu účelovej funkcie) a začína nová generácia.



Obr. 4.6 Schéma spôsobu kríženia jednej matice v programe. Každý člen populácie obsahuje dve matice s váhami, ktoré sú krížené týmto spôsobom. Pri tomto krížení vzniká z náhodne 2 vybraných rodičov len jeden potomok, pričom proces sa opakuje dovtedy, kým nezískame celú novú populáciu potomkov

```
//krizenie

for(k=0;k<pop;k++){

//ruleta
sum=0.;
roulette=0.;
for(i=0;i<pop;i++){
   sum+=(1/mtrain[i]);
}
for(i=0;i<pop;i++){
   power[i]=roulette+((1/mtrain[i])/sum);
   roulette=power[i];
}

for(i=0;i<2;i++){
roulette=rand();
roulette=roulette/RAND_MAX;
   for(j=0;j<pop;j++){
      if(roulette<=power[j]){vyber[k][i]=j;break;}
   }
}

}

//rez
for(k=0;k<pop;k++){
if(vyber[k][0]!=vyber[k][1]){
rollcross=rand();
rollcross=(rollcross/RAND_MAX)*100.0;
if(rollcross<kriz){
rez=rand();
rez=rez%(sk+1);
for(j=0;j<sk;j++){
   for(l=0;l<(vs*dr);l++){
      if(j<rez){bv_cross[k][j][l]=bv[vyber[k][1]][j][l];}
   }
}
for(j=0;j<(sk*dr);j++){
   if(j<(rez*dr)){bt_cross[k][j]=bt[vyber[k][1]][j];}
}
rez=rand();
rez=rez%(vy+1);
for(j=0;j<vy;j++){
   for(l=0;l<(sk*dr);l++){
      if(j<rez){bw_cross[k][j][l]=bw[vyber[k][1]][j][l];}
   }
}
for(j=0;j<(vy*dr);j++){
   if(j<(rez*dr)){bu_cross[k][j]=bu[vyber[k][1]][j];}
}
}

}

}
```

Obr. 4.7 C kód procedúry krizenie

```
//mutacia
for(k=0;k<pop;k++){
for(i=0;i<sk;i++){
   for(j=0;j<(vs*dr);j++){
      rollmut=rand();
      rollmut=rollmut/RAND_MAX;
      if(rollmut<prob){bv[k][i][j]=1-bv[k][i][j];mutcounter++;}
   }
}
for(i=0;i<vy;i++){
   for(j=0;j<(sk*dr);j++){
      rollmut=rand();
      rollmut=rollmut/RAND_MAX;
      if(rollmut<prob){bw[k][i][j]=1-bw[k][i][j];mutcounter++;}
   }
}
for(i=0;i<(sk*dr);i++){
   rollmut=rand();
   rollmut=rollmut/RAND_MAX;
   if(rollmut<prob){bt[k][i]=1-bt[k][i];mutcounter++;}
}
for(i=0;i<(vy*dr);i++){
   rollmut=rand();
   rollmut=rollmut/RAND_MAX;
   if(rollmut<prob){bu[k][i]=1-bu[k][i];mutcounter++;}
}
}
```

Obr. 4.8 C kód procedúry mutacia

Na obrázku 4.7 je zobrazený kód procedúry kríženia z môjho programu. Najskôr sa vyberú dva členy populácie na kríženie pomocou rulety. Do premennej *sum* je uložená suma prevrátených hodnôt účelovej funkcie trénovacej množiny jednotlivých členov populácie. Premenná *power* obsahuje hodnoty sily členov v populácii, pričom počet týchto členov je označený ako *pop*. Funkcia *rand* vracia náhodné celé čísla od 0 po *RAND_MAX* (konštanta s hodnotou minimálne 32767). Funkcia aj konštanta sú do programu importované cez hlavičkový súbor *stdlib.h*. Vybrané dvojice členov populácie sú uložené do premennej *vyber*. V samotnom procese kríženia sa najprv pomocou vygenerovania náhodného čísla určí, či kríženie vôbec nastane. Pravdepodobnosť, že sa kríženie uskutoční sa nachádza v premennej *kriz*. Premenná *rez* obsahuje náhodne vygenerované číslo riadku. Všetky riadky s menším indexom ako je v premennej *rez*, sú nahradené riadkami z druhého člena vybraného do kríženia. Nový člen populácie má prahové a váhové koeficienty (v binárnom tvare) dočasne uložené v pomocných premenných *bv_cross*, *bt_cross*, *bw_cross* a *bu_cross*.

Obrázok 4.8 zobrazuje procedúru mutácie. V premennej *prob* sa nachádza hodnota pravdepodobnosti, s akou v každom jednom bite nastane zmena (z 0 na 1 alebo z 1 na 0). Premenná *mutcounter* slúži na počítanie uskutočnených mutácii.

4.4 Ukončenie trénovania a otestovanie neurónovej siete

Proces trénovania skončí, keď prejde stanovený počet iterácii. Jeho výstupom sú optimálne váhy a prahové koeficienty neurónov ako aj hodnoty účelovej funkcie trénovacej aj testovacej množiny v každom iteračnom kroku. Optimálne váhy potom môžeme využiť ako vstup do ďalšej procedúry programu zvanej *predikcia*, ktorá spraví prechod sieťou s optimálnymi váhami, denormuje výstupy siete a vypočíta odchýlky medzi výstupmi siete a požadovanými výstupmi. Hodnoty účelovej funkcie oboch množín možno použiť na vykreslenie grafu závislosti hodnoty účelovej funkcie od počtu iteračných krokov. Tento graf môžeme použiť napr. na overenie správnosti trénovacieho procesu a na jeho prípadnú ďalšiu analýzu. Ešte však treba dodať, že málokedy sa podarí získať optimálne váhy po uskutočnení jediného trénovacieho procesu. Na dosiahnutie požadovaných výsledkov často potrebujeme proces opakovať viackrát (často sú to aj desiatky opakovaní), pričom pri tom môžeme aj nemusíme (ak nie je treba)

Vytvorený program som použil najskôr na natrénovanie siete pre dáta uvedené v podkapitole 4.1. Uvádzam parametre siete, ktoré som použil pri úspešnom natrénovaní pri jednotlivých metódach:

Tab. 4.1 Základné parametre zhodné pre všetky algoritmy

Počet vstupných neurónov	13
Počet skrytých neurónov	17
Počet výstupných neurónov	1
Mohutnosť trénovacej množiny	250
Trénovacia a testovacia množina spolu	326

Tab. 4.2 Parametre algoritmu Backpropagation

Počet iterácii	20000
Počiatočná rýchlosť učenia	0.02
Momentum	0.3

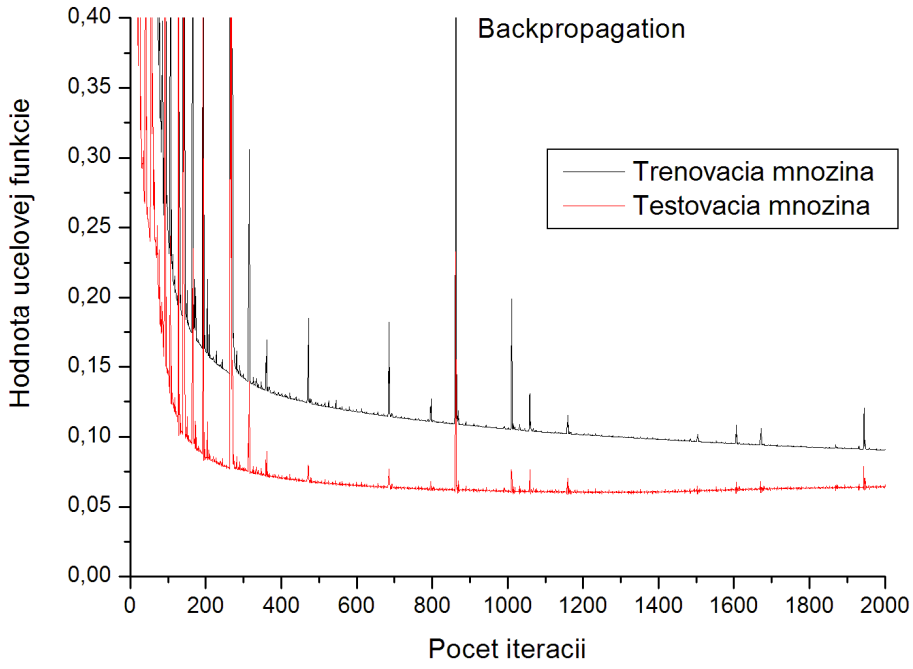
Tab. 4.3 Parametre genetického algoritmu

Počet generácii	100000
Počet členov v populácii	6
Pravdepodobnosť kríženia	0.5
Pravdepodobnosť mutácie	0.02

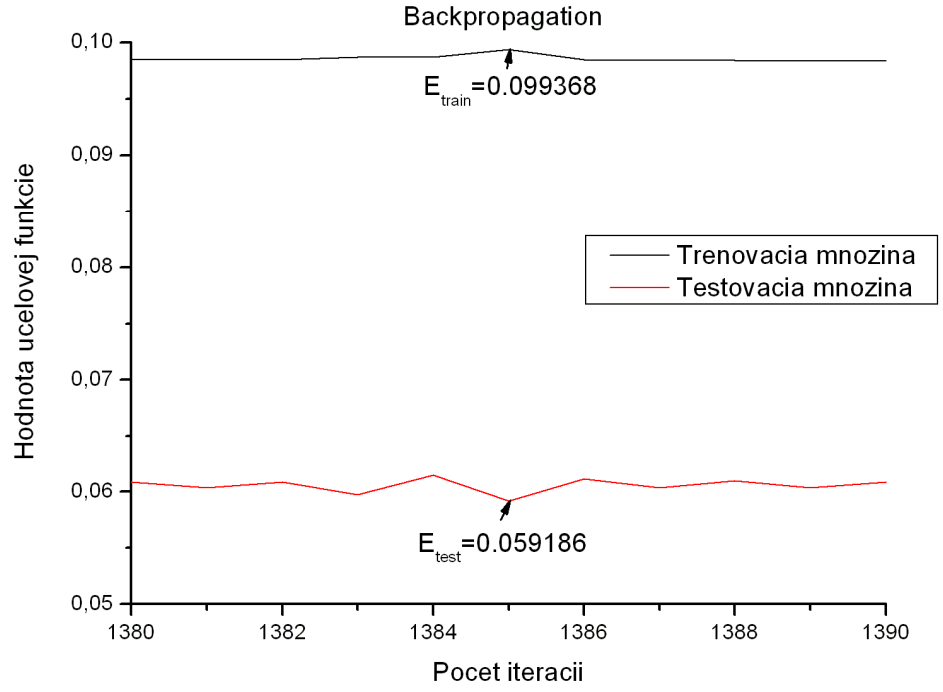
Tab. 4.4 Parametre algoritmu Backpropagation s generovaním počiatočných váh cez genetický algoritmus

Počet iterácii	15000
Počiatočná rýchlosť učenia	0.02
Momentum	0.3
Počet generácii	150
Počet členov v populácii	6
Pravdepodobnosť kríženia	0.5
Pravdepodobnosť mutácie	0.02

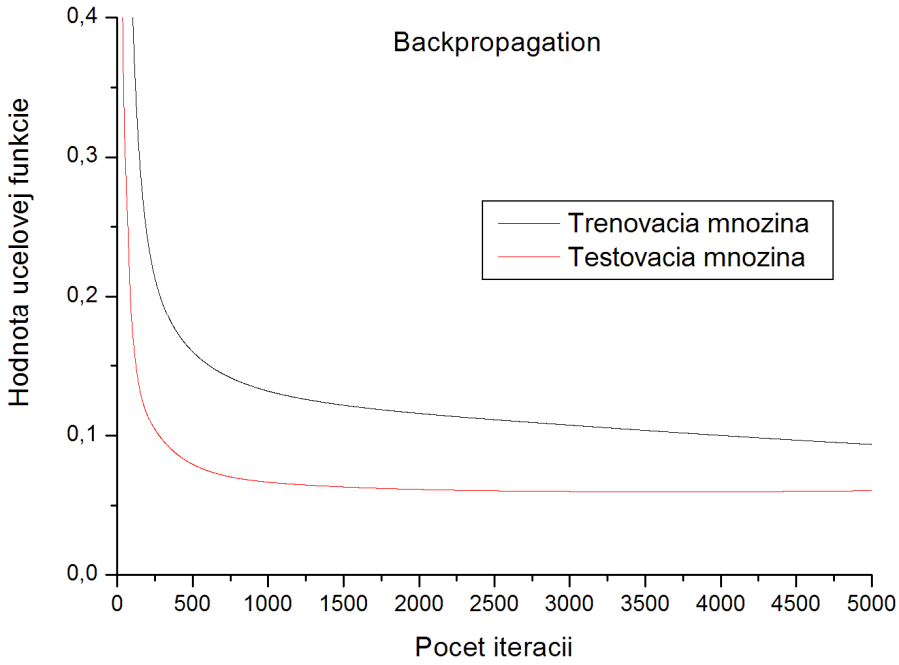
Po natrénovaní som dostal hodnoty účelovej funkcie pre trénovaciu a testovaciu množinu. Prikladám preto grafy, ktoré zobrazujú priebehy týchto účelových funkcií pre jednotlivé metódy, ako aj ukážkový graf (obrázok 4.10), zobrazujúci detailný záber globálneho minima testovacej množiny pri metóde Backpropagation:



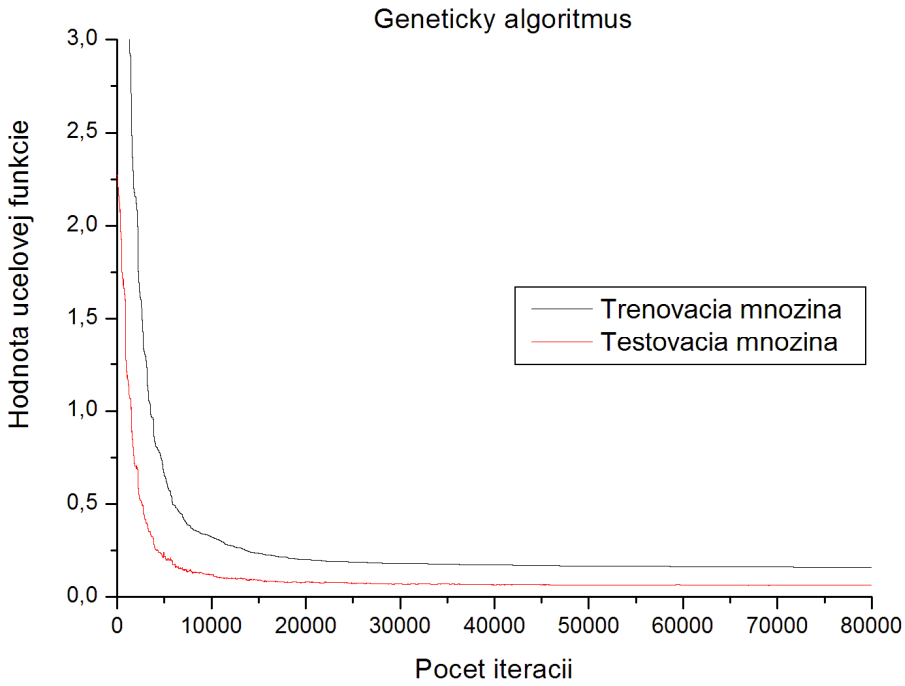
Obr. 4.9 Graf závislosti hodnoty účelovej funkcie od počtu iteračných krokov pri metóde Backpropagation – priebeh účelovej funkcie trénovacej a testovacej množiny



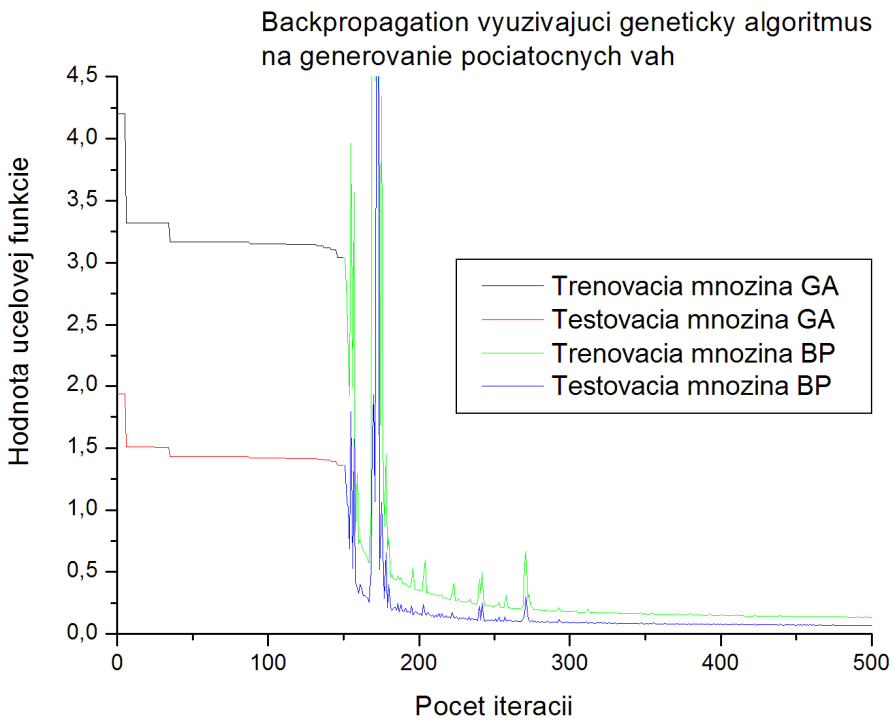
Obr. 4.10 Graf závislosti hodnoty účelovej funkcie od počtu iteračných krokov pri metóde Backpropagation - detailné zobrazenie globálneho minima testovacej množiny a hodnoty v trénovacej množine pri danom iteračnom kroku



Obr. 4.11 Graf závislosti hodnoty účelovej funkcie od počtu iteračných krokov pri metóde Backpropagation - priebeh účelovej funkcie trénovacej a testovacej množiny bez použitia dynamickej zmeny rýchlosti učenia (nepoužité pri vyhodnocovaní účinnosti metód)



Obr. 4.12 Graf závislosti hodnoty účelovej funkcie od počtu iteračných krokov pri metóde genetického algoritmu - priebeh účelovej funkcie trénovacej a testovacej množiny



Obr. 4.13 Graf závislosti hodnoty účelovej funkcie od počtu iteračných krokov pri metóde Backpropagation s využitím genetického algoritmu na generovanie počiatočných váh - priebeh účelovej funkcie trénovacej a testovacej množiny, pričom GA označuje priebeh účelových funkcií počas genetického algoritmu a BP ich priebeh počas Backpropagation

Ako môžme vidieť z grafu na obrázku 4.11, priebehy účelových funkcií správne natrénovanej neurónovej siete majú hladký priebeh aj s metódou Backpropagation. Dôvod, prečo účelové funkcie pri metódach využívajúce algoritmus Backpropagation (grafy na obrázkock 4.10 a 4.13) nemali hladký priebeh počas toho, ako som trénoval sieť týmito metódami, je využitie dynamickej zmeny rýchlosti učenia spomínanej v podkapitole 4.3. Avšak pre porovnanie uvádzam výsledky procesu trénovania:

- Backpropagation s dynamickou zmenou rýchlosti učenia dosiahla optimum účelovej funckie testovacej množiny v 1385. iterácii, optimálna hodnota účelovej funkcie testovacej množiny bola 0.059186 pri hodnote účelovej funkcie trénovacej množiny 0.099368.
- Backpropagation bez dynamickej zmeny rýchlosti učenia dosiahla optimum účelovej funckie testovacej množiny v 3720. iterácii, optimálna hodnota účelovej funkcie testovacej množiny bola 0.059613 pri hodnote účelovej funkcie trénovacej množiny 0.102006.

5. Komparácia použitých trénovacích algoritmov

Po úspešnom natrénovaní neurónovej siete všetkými troma učiacimi algoritmami zostáva už len posledná úloha – zistiť, ktorý z použitých algoritmov by bol na riešenie podobných úloh najvhodnejší, pričom pri výbere treba zohľadniť dve základné kritéria:

1. Časovú náročnosť algoritmu
2. Presnosť predikcie natrénovanej siete pri vstupoch, ktoré neboli súčasťou trénovacej množiny

5.1 Časová náročnosť algoritmov

Jedná sa o čas, ktorý potrebuje daný algoritmus na uspokojivé natrénovanie neurónovej siete, t.j. nájdenie globálneho minima účelovej funkcie testovacej množiny. Tento čas závisí od rôznych faktorov akými sú napr. hardvérové a softvérové vybavenie počítača, ktorý realizuje výpočet, zložitosť samotného algoritmu (čas potrebný na uskutočnenie jedného iteračného kroku) alebo hodnoty počiatočných váh.

Časovú náročnosť použitých algoritmov zobrazuje nasledujúca tabuľka:

Tab. 5.1 Časová náročnosť algoritmov

Algoritmus	Počet iterácií potrebných na nájdenie optima	Čas potrebný na nájdenie optima
BP	1385	10.411 s
GA	66909	1 h 19 min 12.129 s
GA-BP	1641	21.078 s

Pozn. BP značí Backpropagation algoritmus, GA genetický algoritmus a GA-BP Backpropagation algoritmus s využitím genetického algoritmu na generovanie počiatočných váh.

Pre úplnosť týchto údajov ešte musím dodať, že hodnoty uvedené pri Backpropagation algoritme využivajúceho genetický algoritmus na generovanie počiatočných váh (GA-BP) v sebe zahŕňajú obe časti algoritmu, t.j. hodnota počtu iterácií v sebe obsahuje aj 150 generácií genetického algoritmu a časový údaj v sebe ukrýva aj čas, za ktorý sa uskutočnil genetický algoritmus (13.273 s). Technické

parametre počítača, na ktorom bežal môj program, sú uvedené v prílohe bakalárskej práce.

5.2 Presnoť predikcie algoritmov

Hlavný účel trénovania neurónej siete je dosiahnutie jej schopnosti správne (s určitou presnosťou) predpovedať výstupné hodnoty zo vstupných údajov, ktoré sieť nepozná, t.j. nenachádzali sa v trénovacej množine. Najrýchlejším spôsobom, ako overiť presnosť predikcie je porovnanie výstupných hodnôt a požadovaných hodnôt zo vstupov, ktoré boli umiestnené do testovacej množiny. Dobre natrénovanú sieť môžeme samozrejme použiť aj na predikciu výstupov na základe vstupov, ktoré sa vôbec nenachádzali v trénovacej ani testovacej množine. Keďže princípom trénovania siete je minimalizácia účelových funkcií oboch množín, ich hodnota v bode globálneho minima testovacej množiny nám už napovie, ako dobre je sieť natrénovaná. Vo všeobecnosti môžeme povedať, že čím menšie budú tieto funkčné hodnoty, tým presnejšie bude sieť predikovať výstupné hodnoty.

V nasledujúcej tabuľke sa nachádzajú hodnoty účelovej funkcie trénovacej aj testovacej množiny v optime (globálnom minime testovacej množiny) pri jednotlivých algoritmoch:

Tab. 5.2 Hodnoty účelovej funkcie trénovacej a testovacej množiny v bode globálneho minima testovacej množiny pri jednotlivých algoritmoch

Algoritmus	Počet iterácií potrebných na nájdenie optima	Hodnota účelovej funkcie trénovacej množiny v bode optima	Hodnota účelovej funkcie testovacej množiny v bode optima
BP	1385	0.099368	0.059186
GA	66909	0.162698	0.062289
GA-BP	1641	0.084288	0.051104

Pozn. BP značí Backpropagation algoritmus, GA genetický algoritmus a GA-BP Backpropagation algoritmus s využitím genetického algoritmu na generovanie počiatočných váh.

Po prechode sieťou s optimálnymi váhami jednotlivých metód (procedúra programu *predikcia*) som dostal výstupy siete ako aj ich odchýlky od požadovaných hodnôt pre všetky vstupné dáta. Keďže z pohľadu predikcie je zaujímavá len testovacia množina

(posledných 76 vstupov a požadovaných výstupov z celkového počtu 326), nasledujúce údaje zohľadňujú len prvky testovacej množiny:

Tab. 5.3 Presnosť predikcie výstupov na základe prvkov testovacej množiny pri jednotlivých algoritmoch

Algoritmus	Najmenšia odchýlka od požadovanej hodnoty (%)	Najväčšia odchýlka od požadovanej hodnoty (%)	Priemerná odchýlka od požadovanej hodnoty (%)
BP	0.09	65.31	8.47
GA	0.09	36.72	9.32
GA-BP	0.07	33.62	7.14

Tab. 5.4 Ukážka predikcie prvých 5 členov testovacej množiny pri jednotlivých algoritmov

Algoritmus	Poradie	Požadovaná výstupná hodnota	Výstupná hodnota siete	Rozdiel medzi požadovanou a výstupnou hodnotou
BP	1.	5	4,93404	0,06596
	2.	7	6,82913	0,17087
	3.	7	6,43430	0,56570
	4.	7	6,73196	0,26804
	5.	3	2,93798	0,06202
GA	1.	5	4,58624	0,41376
	2.	7	6,95128	0,04872
	3.	7	6,54018	0,45982
	4.	7	6,64697	0,35303
	5.	3	3,15685	-0,15685
GA-BP	1.	5	4,78817	0,21183
	2.	7	7,18983	-0,18983
	3.	7	6,52906	0,47094
	4.	7	6,88285	0,11715
	5.	3	3,10954	-0,10953

Pozn. BP značí Backpropagation algoritmus, GA genetický algoritmus a GA-BP Backpropagation algoritmus s využitím genetického algoritmu na generovanie počiatočných váh.

5.3 Zhodnotenie

Ako najvhodnejší algoritmus by som označil Backpropagation algoritmus s využitím genetického algoritmu na generovanie počiatočných váh (GA-BP) z nasledujúcich dôvodov:

- presnosť predikcie – dosiahol najnižšiu priemernú odchýlku výstupov siete od požadovaných hodnôt v testovacej množine, čo bolo primárnym cieľom tejto úlohy.
- časovo bol síce o niečo pomalší ako klasický Backpropagation algoritmus, avšak vyše polovicu času zabral genetický algoritmus, ktorý je výpočtovo náročnejší ako Backprogation. Navyše vďaka čiastočne optimalizovaným počiatočným váham cez genetický algoritmus je možné, že pri niektorých úlohach bude stačit menší počet iterácií na nájdenie optima ako pri bežnom Backpropagation.

Najmenej výhodný je naopak čistý genetický algoritmus, pretože vďaka svojej výpočtovej náročnosti je čas potrebný na nájdenie optima neporovnateľne dlhší ako pri ostatných metódach. Taktiež dosiahol najmenšiu presnosť predikcie pri riešenej úlohe.

6. Záver

Cieľom práce bolo použiť viac typov metód učenia viacvrstvovej neurónovej siete s dopredným šírením informácie a vyhodnotiť ich vplyv na schopnosť neurónovej siete predikovať vlastností chemických látok v závislosti od ich vstupných charakteristík. Taktiež bolo treba vyhodnotiť aj časovú náročnosť použitých algoritmov. Na riešenie tejto úlohy bolo potrebné naprogramovať vlastný program a taktiež vybrať vhodné dáta.

Za účelom trénovania neurónovej siete som zvolil tri algoritmy – Backpropagation (spätné šírenie chybového signálu), genetický algoritmus a Backpropagation vylepšený o generovanie počiatočných váh siete cez genetický algoritmus.

Prvá časť úlohy pozostávala z realizácie samotného programu. Napísal som ho v programovacom jazyku C, táto neurónová sieť je naprogramovaná všeobecne, čo umožňuje jej využitie aj na iné úlohy, keďže počet neurónov v jednotlivých vrstách ako aj rada ďalších parametrov je nastaviteľná pred každým spustením siete.

Ďalšia časť úlohy bolo natrénovať sieť na zvolených dátach všetkými troma trénovacím algoritmami. V dátach, ktoré som mal k dispozícii, bola skúmaná výstupná chemicka vlastnosť chemický posun v alkánoch a vstupné charakteristiky boli frekvencie výskytu zvolených fragmentov uhlíkového reťazca, ktoré boli usporiadané do vektora nazývaného deskriptor. So všetkými troma algoritmami sa mi podarilo natrénovať sieť dostatočne na to, aby mohla predikovať výstupné hodnoty aj na základe vstupov, ktoré nemala v procese trénovania k dispozícii.

Poslednou časťou úlohy bola samotná komparácia jednotlivých trénovacích algoritmov. Za najvhodnejší som označil posledný algoritmus - Backpropagation s generovaním počiatočných váh cez genetický algoritmus, pretože dosiahol najlepšie výsledky v predikcii a ani z hľadiska časovej náročnosti príliš nezaostával za bežným Backpropagation algoritmom, ktorý bol najrýchlejší zo všetkých. Naopak najnevhodnejším algoritmom na trénovanie bol genetický algoritmus, hlavne kvôli svojej neporovnateľne väčšej časovej náročnosti spôsobenej svojou výpočtovou náročnosťou.

Výsledok komparácie nebol prekvapením, naopak bol očakávaný. Kombinácia rýchlosti algoritmu Backpropagation s čiastočnou optimalizáciou počiatočných váh

cez genetický algoritmus už dopredu sľubovala potenciál dosahovania lepších výsledkov ako klasické verzie týchto algoritmov.

Použitá literatúra

[1] Kvasnička V., Pospíchal J., Kozák Š., Návrat P., Paroulek P.: Umelá Inteligencia a Kognitívna Veda I, Slovenská Technická Univerzita v Bratislave, 2009, str. 37-43

[2] Kvasnička V., Benušková Ľ., Pospíchal J., Farkaš I.,Tiňo P., Kráľ A.: Úvod do teórie neurónových sietí, Vydavateľstvo IRIS, 1997, str. 11-20, 32-38, 70, 111-116, 255-256

[3] Novák M., Faber J., Kufudaki O.: Neurónové sítě a informační systémy živých organismů, Grada a.s. v Praze, 1993, str. 19, 41

[4] Boor, Š.: Maximum Coverage Method for Feature Subset Selection for Neural Network Training, Computing and Informatics č. 5 zv. 30, 2011, str. 901–912

[5] Svozil D., Pospíchal J., Kvasnička V.: Neural network prediction of carbon-13 NMR chemical shifts of alkanes, Journal of Chemical Information and Computer Sciences 35, 1995, str. 924-928

Prílohy

Príloha 1: Technické parametre počítača použité na výpočet

Procesor: Intel® Core™ 2 Duo E6600 s frekvenciou každého jadra 2.40GHz

RAM pamäť: 2x 1GB DDR2 SDRAM

Operačný systém: Windows® XP Professional SP2

Intel® a Core™ sú obchodnou alebo registrovanou obchodnou známkou spoločnosti Intel Corporation

Windows® je registrovanou obchodnou známkou spoločnosti Microsoft Corporation

Príloha 2: CD

Priložené CD obsahuje nasledujúce súbory:

Bakalárska práca.pdf (text bakalárskej práce)

data.rar (archív obsahujúci textové súbory .txt s použitými dátami)

sources.rar (archív obsahujúci zdrojové kódy .cpp naprogramovanej neurónovej siete)