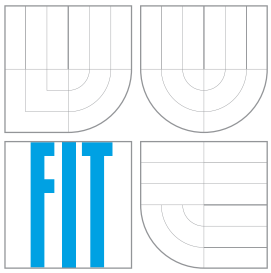
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ
FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

GENETICKÝ NÁVRH KLASIFIKÁTORU S VYUŽITÍM NEURONOVÝCH SÍTÍ

NEURAL NETWORKS CLASSIFIER DESIGN USING GENETIC ALGORITHM

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. MICHAL TOMÁŠEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VOJTĚCH MRÁZEK

BRNO 2016

Zadání diplomové práce/18049/2015/xtomas27

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačových systémů Akademický rok 2015/2016

Zadání diplomové práce

Řešitel: **Tomášek Michal, Bc.**
Obor: Inteligentní systémy
Téma: **Genetický návrh klasifikátoru s využitím neuronových sítí**
Neural Networks Classifier Design using Genetic Algorithm
Kategorie: Umělá inteligence

Pokyny:

1. Nastudujte problematiku genetických algoritmů, evoluce umělých neuronových sítí pomocí metody NeuroEvolution of Augmenting Topologies (NEAT) a klasifikace.
2. Navrhněte program, který umožní tvorbu klasifikátorů pomocí genetického algoritmu.
3. Program z bodu 2 implementujte.
4. Ověřte funkčnost programu v zadaných úlohách.
5. Zhodnoťte dosažené výsledky a diskutujte přínos práce.

Literatura:

- Dle pokynů vedoucího práce.

Při obhajobě semestrální části projektu je požadováno:

- Splnění bodu 1 a 2 zadání.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Mrázek Vojtěch, Ing.**, UPSY FIT VUT
Datum zadání: 1. listopadu 2015
Datum odevzdání: 25. května 2016

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačových systémů a sítí
612 66 Brno, Božetěchova 2

L.S.

Kotásek

doc. Ing. Zdeněk Kotásek, CSc.
vedoucí ústavu

Abstrakt

Cílem této práce je genetický návrh neuronových sítí, jenž budou schopné provádět klasifikaci v rámci různých klasifikačních úloh. K vytváření těchto neuronových sítí je použit algoritmus vycházející z algoritmu NeuroEvolution of Augmenting Topologies (zkráceně známého jako NEAT). Dále je představena myšlenka předzpracování, která je v implementovaném výsledku rovněž zahrnuta. Cílem předzpracování je snížení výpočetních nároků pro zpracování datové sady daného klasifikačního problému. Výsledkem této práce je množina experimentů provedených nad datovou sadou pro detekci rakovinných buněk a databází ručně psaných číslic MNIST. Klasifikátory vytvořené pro rakovinné buňky pak dosahují více jak 99% přesnosti a u experimentu MNIST dochází ke snížení výpočetních nároků o více jak 10% se zanesením zanedbatelné chyby o velikosti 0,17%.

Abstract

The aim of this work is the genetic design of neural networks, which are able to classify within various classification tasks. In order to create these neural networks, algorithm called NeuroEvolution of Augmenting Topologies (also known as NEAT) is used. Also the idea of preprocessing, which is included in implemented result, is proposed. The goal of preprocessing is to reduce the computational requirements for processing of benchmark datasets for classification accuracy. The result of this work is a set of experiments conducted over a data set for cancer cells detection and a database of handwritten digits MNIST. Classifiers generated for the cancer cells exhibits over 99 % accuracy and in experiment MNIST reduces computational requirements more than 10 % with bringing negligible error of size 0.17 %.

Klíčová slova

Evoluční algoritmus, Genetický algoritmus, Neuron, Neuronová síť, Umělá neuronová síť, TWEANN, NEAT, Klasifikace, Předzpracování, Úspora energie, Klasifikace rakoviny prsu, MNIST

Keywords

Evolutionary algorithm, Genetic algorithm, Neuron, Neural network, Artificial neural network, TWEANN, NEAT, Classification, Preprocessing, Energy saving, Breast cancer classification, MNIST

Citace

TOMÁŠEK, Michal. *Genetický návrh klasifikátoru s využitím neuronových sítí*. Brno, 2016. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Mrázek Vojtěch.

Genetický návrh klasifikátoru s využitím neuronových sítí

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Vojtěcha Mrázka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Michal Tomášek
20. května 2016

Poděkování

Rád bych tímto poděkoval vedoucímu práce Ing. Vojtěchovi Mrázkovi za konzultace, zpětnou vazbu a odbornou pomoc.

Obsah

Kapitola 1

Úvod

V současné době se za pomoci výpočetní techniky stále častěji řeší problémy, které nejsou řešitelné pomocí deterministických algoritmů. Tyto problémy se typicky vyznačují velmi rozsáhlým stavovým prostorem, jenž by se deterministicky bez jakékoliv heuristiky prohledával velmi dlouho. Řešením těchto a dalších problémů se zabývá oblast nazývaná *softcomputing*. V této oblasti mimo jiné nachází uplatnění právě evoluční algoritmy a neuronové sítě, kterými se tato práce zabývá.

Neuronové sítě jsou obecným výpočetním modelem a umožňují řešit různorodé problémy. Jsou reprezentovány množinou uzlů (neuronů) a spojení mezi nimi. Topologie takových sítí pak bývá statická a empiricky vytvářená. Jejich slabinou je ale to, že s růstem komplexnosti problému roste jejich velikost, a tím i složitost jejich učení a případného návrhu topologie. Standardní techniky učení pak selhávají a je zapotřebí provádět proces učení jinak.

Jedním z alternativních způsobů učení je použití evoluce. Při tomto procesu je neuronová síť zakódována do řetězce, který podléhá evolučnímu procesu. Tímto procesem jsou sítě postupně zlepšovány, přičemž je jím nahrazeno jak učení, tak i tvorba vhodné topologie sítě. Tento způsob tvorby a učení neuronových sítí je velmi výhodný pro složité či měnící se problémy, kde by byl empirický návrh a tradiční metody učení neefektivní.

V této práci je popsána teorie týkající se evolučních algoritmů (kapitola 2), konkrétně pak základní pojmy a obecný průběh evolučního algoritmu. Následně je popsána problematika genetických algoritmů. Ty jsou charakterizovány i z teoretického hlediska funkčnosti. Dále je podkapitola zaměřena na všechny důležité součásti těchto algoritmů. Kapitola 3 se zaměřuje na umělé neuronové sítě. Nejprve je popsána struktura a činnost biologického neuronu, který je vzorem pro neuron umělý. Následně je zadefinován neuron umělý včetně jeho matematického modelu. Tento neuron se používá pro formování tzv. umělých neuronových sítí, které se mohou učit pomocí popsaných metod tradičního učení. Také jsou popsána jejich možná využití v praxi včetně klasifikace, na kterou je tato práce zaměřena. Alternativní přístup k vytváření a učení neuronových sítí je popsán v kapitole 4. Tento způsob je demonstrován především na algoritmu NEAT, ale jsou zde i alternativní metody, které lze pro tyto účely použít. Následuje kapitola 5 popisující návrh řešení. Kapitola se zaměřuje na použitý princip předzpracování. Implementace algoritmu NEAT v kombinaci s principem předzpracování je popsána v kapitole 6. Mimo jiné obsahuje tato kapitola i pseudokód popisující konkrétní fitness funkci používanou v experimentech. Ty jsou rozebrány v kapitole 7, která je rozdělena na dvě podkapitoly zaměřující se na experimenty nad dvěma klasifikačními problémy. Poslední kapitola (kapitola 8) shrnuje výsledky dosažené v této práci, a také možné výhody a nevýhody zvoleného řešení.

Kapitola 2

Evoluční algoritmy

Tato kapitola popisuje evoluční algoritmy a čerpá zejména z následujících zdrojů [7, 19, 32], kde je možné nalézt více informací. Evoluční algoritmy patří mezi základní nástroje moderní informatiky pro hledání řešení v extrémně složitých problémech, kdy nelze použít standardní deterministické metody založené na technikách úplného prohledávání. Jedná se o nástroje provádějící hledání řešení v prohledávacím prostoru, ve kterém jsou přítomna všechna povolená řešení problému.

Pojem evoluční algoritmus zastřešuje stochastické prohledávací algoritmy, které jsou inspirovány Darwinovou evoluční teorií a genetikou obecně. Tyto algoritmy se pak dále dělí na genetické algoritmy, evoluční strategie, evoluční programování a genetické programování. Všechny algoritmy, jenž termín evoluční algoritmus zastřešuje, mají tyto společné rysy:

- Populace – používá se populace kandidátních řešení umožňující paralelní přístup k prohledávání.
- Reprodukce – pro vytváření nových kandidátních řešení se používají biologií inspirované operátory.
- Přirozený výběr – jedinci s vyšším ohodnocením vstupují do procesu reprodukce s vyšší pravděpodobností.
- Genetický drift – náhodná událost v životě jedince může ovlivnit celou populaci; může se jednat o mutaci, nebo o náhlou smrt jedince s vysokým ohodnocením.

Evoluční algoritmy jsou v současnosti velmi populární optimalizační technikou a přestože se původně využívaly pouze pro optimalizaci parametrů vyvíjeného systému, nyní jsou často používány i pro návrh jeho struktury.

2.1 Základní pojmy

V této sekci jsou popsány základní pojmy týkající se evolučních algoritmů. Přestože je mnoho z těchto termínů používáno i v biologii, jejich význam se může mírně lišit. Pojmy jsou zde definovány s ohledem na evoluční algoritmy, tedy z informatického pohledu a bez zaměření na jakoukoliv skupinu algoritmů, kterou pojem evoluční algoritmy zastřešuje.

Gen Vloha pro určitou vlastnost. V oblasti evolučních algoritmů se také jedná o funkční jednotku dědičnosti, která zakódovává nějakou vlastnost fenotypu. Představuje určitou část chromozomu.

Alela Konkrétní varianta genu, jenž se nachází na definované pozici v chromozomu. Alela reprezentuje pouze jednu z možných hodnot, kterých daný gen může nabývat.

Genotyp (Chromozom) Kombinace alel, jejichž nositelem je konkrétní jedinec. Obsahuje všechny informace potřebné k vytvoření odpovídajícího fenotypu. Jedná se o zakódování fenotypu. V evolučních algoritmech jsou označení genotyp a chromozom považována za synonyma, přestože tomu tak v biologii není.

Fenotyp Jedinec se všemi jeho znaky, vlastnostmi a projevy. Představuje kandidáta řešení dané úlohy v konkrétním prostředí. Mezi fenotypem a genotypem je u jednodušších problémů přímé zobrazení, kdy každému genu odpovídá jeden parametr fenotypu. U složitějších problémů je genotypem předpis pro sestavení fenotypu.

Populace Populace je reprezentována multimnožinou chromozomů. Jedná se o jednotku evoluce, protože jedinci jsou statické objekty, které se nemění ani neadaptují; populace ano. Ve většině algoritmů je velikost populace konstantní a je jedním z parametrů inicializace. Každá populace je součástí nějaké generace. V rámci dané generace pak typicky dochází k obměně určité části populace.

Fitness funkce Funkce poskytující heuristický odhad kvality řešení, reprezentovaného hodnoceným jedincem. Podle způsobu definice této funkce, může být hodnota fitness vyjádřena různými způsoby:

- Hrubá – udávána v hodnotách přirozených problémové doméně.
- Standardizovaná – vyjádření hrubé hodnoty fitness do podoby, kdy je nižší numerická hodnota žádanější.
- Přizpůsobená – převrácená hodnota součtu standardizované hodnoty fitness a čísla 1.
- Normalizovaná – podíl hrubé hodnoty fitness jedince a sumy všech hrubých hodnot fitness jedinců v populaci.

Mutace Unární operátor, který po aplikaci na jedince produkuje (mírně) modifikovaného mutanta, potomka nebo více potomků. Operátor mutace je vždy stochastický. Jeho využití se může velmi lišit v závislosti na typu evolučního algoritmu. Hlavním účelem mutace je vytváření jedinců, jenž jsou pro danou populaci z hlediska křížení netypičtí. Díky tomu je sníženo riziko uváznutí v lokálním optimu prohledávaného prostoru.

Křížení (Rekombinace) Binární operátor spojující informaci od dvou rodičovských chromozomů do jednoho nebo dvou chromozomů potomků. Opět je jeho využití různé v závislosti na typu evolučního algoritmu. Cílem křížení je vhodná kombinace chromozomů rodičů tak, aby vznikli potomci s kvalitními chromozomy, a tedy i vysokou hodnotou fitness. Mezi základní metody křížení patří:

- Jednobodové – náhodně je stanoven bod křížení; první potomek vzniká použitím první části (do bodu křížení) chromozomu prvního rodiče a druhé části druhého rodiče, druhý potomek naopak.

- Vícebodové – náhodně je vytvořeno více bodů křížení; princip vytváření potomků je stejný jako u jednobodového křížení.
- Uniformní – každý gen se do potomka přenese od prvního nebo druhého rodiče náhodným výběrem.

Operátor selekce Unární operátor, jehož cílem je vybrat jedince na základě jejich kvality, a tím umožnit lepším z nich stát se rodiči následující generace. Jedinec se stává rodičem, jestliže byl vybrán selekčním operátorem, aby podstoupil variaci za účelem vytvoření potomstva. Společně s nahrazovací strategií je operátor selekce zodpovědný za zvyšování kvality populací. Zpravidla je selekce pravděpodobnostní, s přímou závislostí na fitness hodnotě jedinců. Vyšší hodnota fitness implikuje vyšší pravděpodobnost výběru pro reprodukci. Přesto mohou mít i jedinci s nízkou hodnotou fitness nenulovou pravděpodobnost výběru, což může být klíčové z pohledu uváznutí v lokálním optimu prohledávaného prostoru. Nejznámější algoritmy selekce jsou:

- Ruleta – každý jedinec má pravděpodobnost výběru odpovídající jeho normalizované hodnotě fitness; suma normalizovaných hodnot fitness je rovna jedné; výběr je následně proveden vygenerováním náhodného čísla z intervalu $\langle 0; 1 \rangle$.
- Výběr podle pořadí – jedinci jsou seřazeni dle hodnoty fitness; jejich pořadí jim určuje pravděpodobnost výběru; následně je provedena ruleta s vypočtenými pravděpodobnostmi.
- Turnaj – z populace je náhodně vybráno X (typicky 2) jedinců; jedinec s nejvyšší fitness hodnotou vítězí a je vybrán.
- Deterministicky – výběr několika nejlepších jedinců.

Nahrazovací strategie Jedná se o velmi podobný mechanismus jakým je operátor selekce s tím rozdílem, že nahrazovací strategie se používá až po vytvoření potomků. Jejím úkolem je vybrat jedince do nové populace. Opět je využívána hodnota fitness, ale výběr je často deterministický. Modely nahrazovacích strategií jsou:

- Generační – všichni jedinci původní populace jsou nahrazeni svými potomky.
- Inkrementační – v nové populaci je nahrazen potomkem pouze jediný jedinec původní populace.
- S překrytím generací – v nové populaci je nahrazena potomky část jedinců původní populace.

Selekční tlak Hypotetická síla, kterou působí prostředí nebo člověk na určitou populaci tím, že z ní odstraňuje nositele určitých znaků. Selekční tlak vyvíjí na populaci především operátor selekce, který upřednostňuje jedince s vysokou hodnotou fitness. Míra selekčního tlaku pak určuje úroveň zohlednění fitness funkce při výběru jedince. Čím vyšší selekční tlak bude, tím více bude selekční operátor upřednostňovat jedince s vysokou hodnotou fitness. Vyšší selekční tlak sice urychluje konvergenci, je zde však riziko nalezení pouze lokálního, nikoliv globálního optima.

Elitismus Princip, který zajišťuje, že se nejlepší jedinec z předchozí populace vždy dostane do populace nové. Tento princip může zajistit rychlejší konvergenci.

2.2 Obecný postup a komponenty algoritmu

Na začátku výpočtu je inicializována počáteční populace obsahující předem stanovený počet kandidátů. Zpravidla je vytvářena náhodně, případně pomocí vhodné heuristiky. V každém cyklu algoritmu, který se označuje jako generace, jsou všechna kandidátní řešení ohodnocena fitness funkcí. U této funkce platí, že vyšší hodnota označuje lepší řešení reprezentované jedincem. Každá nová populace je vytvářena tak, že se nejprve vyberou vhodní jedinci z předchozí populace. Tito jedinci tvoří množinu rodičů. Poté, aplikací genetických operátorů nad množinou rodičů, vznikne množina potomků. Z množiny potomků a rodičů je následně vytvořena vhodným výběrem kandidátů nová populace. Tento výběr provádí selekční algoritmus, který své rozhodování provádí na základě hodnot fitness funkce. Pokud je tento algoritmus navržen dobře, pak průměrná hodnota fitness celé populace poroste. Díky fitness funkci tedy vzniká selekční tlak, který by měl zajistit hledání lepších řešení. Genetické operátory nejsou prováděny deterministicky, přičemž jejich stochastičnost je parametrizovatelná uživatelem. Algoritmus končí, jakmile je nalezen dostatečně kvalitní jedinec, anebo je dosaženo určitého počtu generací. Pseudokód evolučního algoritmu je zobrazen v algoritmu 1.

Algoritmus 1 Pseudokód evolučního algoritmu

```
1:  procedure Evoluční Algoritmus
2:      INICIALIZACE populace náhodnými kandidáty;
3:      VYHODNOCENÍ každého kandidáta;
4:
5:      repeat
6:          SELEKCE rodičů;
7:          REKOMBINACE párů rodičů;
8:          MUTACE potomků;
9:          VYHODNOCENÍ nových kandidátů;
10:         SELEKCE kandidátů pro další generaci;
11:     until UKONČUJÍCÍ PODMÍNKA je splněna
12: end procedure
```

Nejdůležitější komponenty evolučního algoritmu jsou:

- Způsob reprezentace – definice zakódování chromozomů; obvykle je zapotřebí využití abstrakce; určuje i způsob mapování genotypů na fenotypy.
- Vyhodnocovací funkce (fitness funkce) – přímo určuje požadavky, na které by se populace měla adaptovat; představuje základ pro selekci.
- Populace – množina jedinců definované velikosti; velikost se za běhu algoritmu nemění, což vede k vytvoření omezených zdrojů, a tedy i soutěžení jedinců mezi sebou.
- Mechanismus výběru rodičů – výběr rodičů pro rekombinaci; používá hodnoty fitness; pokud je mechanismus příliš *greedy*[1], je zde velká pravděpodobnost uváznutí

[1] Pro *greedy* mechanismus výběru platí, že vždy vybírá nejlepší dosud nalezené řešení.

v lokálním optimu.

- Operátory variace (rekombinace a mutace) – vytváření nových jedinců; noví jedinci představují nová kandidátní řešení daného problému.
- Nahrazovací strategie – způsob nahrazování jedinců v populaci; typicky používá hodnoty fitness; snaha zachovat určitou diverzitu v populaci pro kvalitnější výsledky křížení.

Všechny tyto komponenty jsou pro vytvoření kompletního evolučního algoritmu nezbytné. Navíc je nutné specifikovat způsob inicializace a ukončení celého algoritmu.

2.3 Genetické algoritmy

Genetické algoritmy [7, 16] jsou jednou z variant evolučních algoritmů. Jedná se o nejvíce známý typ evolučních algoritmů. Jejich autorem je John Holland, který je vymyslel jako prostředek ke studiu adaptivního chování ve své knize [15]. Genetické algoritmy byly ale považovány za metody pro optimalizaci funkcí. Nejspíše díky práci jeho žáka Davida Goldberga [9], a také díky úspěchům v oblasti řešení optimalizačních problémů. Společně s tezí De Jonga [4] pak vznikla definice toho, co je považováno za klasický genetický algoritmus - běžně nazývaný jako „kanonický“ nebo „jednoduchý genetický algoritmus“.

2.3.1 Jednoduchý genetický algoritmus

Jednoduchý genetický algoritmus používá binární reprezentaci jedinců, selekci ruletou, nízkou pravděpodobnost mutace a důraz je kladen na použití křížení inspirovaného genetikou jako prostředek pro vytváření potomků.

Genetické algoritmy mají mnohdy tento neměnný pracovní postup: je dána populace o velikosti μ jedinců, algoritmus výběru rodičů naplní dočasnou populaci velikosti μ s tím, že je povolena redundance. Následně je dočasná populace promíchána za účelem vytvoření náhodných párů jedinců, na které je aplikováno křížení s pravděpodobností p_c a potomci tohoto křížení nahrazují své rodiče. Nová dočasná populace podstoupí mutaci jedinců, kde každý z l bitů jedince může být modifikován mutací s pravděpodobností p_m. Výsledná dočasná populace představuje populaci pro novou generaci. Stará populace je tedy nahrazena kompletně. V rámci tohoto procesu může nastat i to, že se zde budou nacházet jedinci, kteří přežili křížení i mutaci, aniž by byli jakkoliv modifikováni. Pravděpodobnost výskytu takovýchto jedinců je však velmi nízká (závisí na parametrech μ, p_c, p_m).

Jednoduchý genetický algoritmus ale má své vady. Mechanismy, jako je elitismus a použití ne-generačních modelů pro nahrazovací strategie, byly přidány až později za účelem zrychlení konvergence. Také selekce ruletou nebo velmi často používaným turnajem neposkytuje dobrý vzorek populace, a proto byl vytvořen algoritmus známý jako *stochastic universal sampling* (SUS). Zkoumání závislostí mezi binární reprezentací a jednobodovým křížením pak vedlo k vývoji alternativ jako je například uniformní křížení. A nakonec je zde i problém, jak správně zvolit vhodnou fixní pravděpodobnost mutací. Tento problém byl nakonec vyřešen tak, že se tato pravděpodobnost zakóduje jako dodatečné geny v jedincích a je umožněn jejich vývoj.

Přesto je jednoduchý genetický algoritmus stále velmi používaný, a to nejen pro účely výuky a porovnávání výkonu s novými algoritmy, ale je vhodný i pro přímočaré problémy, jenž mají dobrou binární reprezentaci.

2.3.2 Kódování

Binární kódování zmíněné v souvislosti s genetickými algoritmy používá geny, které mohou nabývat pouze dvou hodnot (0 nebo 1). Pro vyjádření více možností je pak zapotřebí použít delší binární řetězec, jenž pokryje všechny možnosti. Problematika tohoto kódování spočívá v tom, že u genetických algoritmů platí následující předpoklad: malá změna chromozomu povede na malou změnu fenotypu a naopak. Tento předpoklad však u binárního kódování neplatí, protože změnou nejvíce významného bitu dojde k velké změně hodnoty. Chromozomy, které si jsou velmi podobné (liší se například pouze v jednom bitu), mohou představovat velmi odlišné fenotypy. Tato nevýhoda je však řešitelná pomocí tzv. Grayova kódu, jehož hlavní myšlenkou je, že každé dvě sousední hodnoty jsou zakódované řetězci dané délky tak, aby se lišily pouze v jednom bitu.

Přestože bylo doposud zmíněno pouze kódování pomocí abecedy obsahující dva znaky (0 a 1), nemusí být použito vždy. Tento typ kódování je sice poměrně jednoduchý, ale pro některé problémy může být více výhodné použít abecedu s více než dvěma znaky, nebo přímo čísla s pohyblivou řádovou čárkou. Reprezentace používající reálná čísla jsou velmi výhodná především u úloh, které vyžadují velkou přesnost. U binárního kódování by tyto úlohy vedly na vytváření příliš dlouhých chromozomů, takže by docházelo ke zbytečnému zvýšení velikosti prohledávaného prostoru.

Existují i tzv. permutační kódování, jenž se využívají při řešení kombinatorických a plánovacích úloh. Jedinec je pak reprezentován permutací několika čísel, čímž je určeno pořadí jednotlivých objektů v daném problému.

2.3.3 Fitness funkce

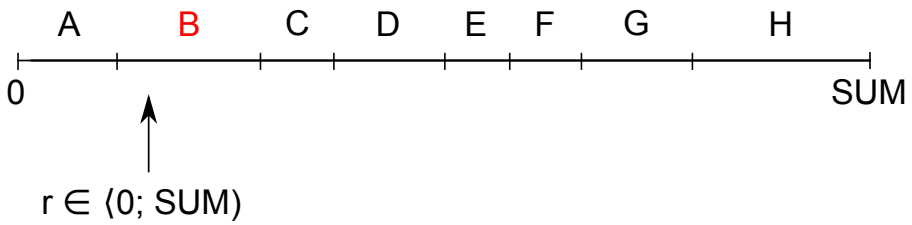
Fitness funkce je vždy úzce svázána s problémem řešeným pomocí genetického algoritmu. Jelikož slouží k vyjádření kvality řešení problému, které reprezentuje jedinec, musí být možné na základě této funkce vytvořit lineární uspořádání. Funkce tedy umožňuje porovnání jedinců. Z toho důvodu, že je nutné vyhodnotit každého jedince v každé generaci, bývá velké množství strojového času spotřebováno právě výpočtem této funkce. Při optimalizaci implementace genetických algoritmů je tedy vhodné zaměřit se na fitness funkci. Existují i některé alternativní způsoby volání fitness funkce, kdy ve výsledku nedochází k volání nad všemi jedinci, případně kombinacemi jedinců. Taková řešení ale typicky snižují přesnost ohodnocení.

2.3.4 Selekce

Ruletový mechanismus selekce známý také jako *fitness-proportionate selection* je jedním z nejčastěji používaných selekčních algoritmů. Principem tohoto algoritmu je náhodné vygenerování čísla v rozsahu $\langle 0; SUM)$, kde SUM značí sumu ohodnocení všech jedinců populace. Tento mechanismus je ilustrován na obrázku 2.1. Alternativně lze provádět součet normalizovaných hodnot fitness. Potom pravděpodobnost výběru každého jedince je rovna přímo jeho normalizované hodnotě fitness a náhodné číslo pro výběr potomka (na obrázku označeno jako r) je generováno v intervalu $\langle 0; 1)$.

Hlavní nevýhodou tohoto přístupu je, že postupně dochází ke snižování rozmanitosti v populaci, díky neustálému upřednostňování jedinců s vysokým ohodnocením. V případě, kdy bude nejlépe ohodnocený jedinec příliš dominantní[2], bude docházet k nevhodným výběrům. Ty způsobí, že bude křížen zmíněný jedinec sám se sebou, nebo bude součástí většiny

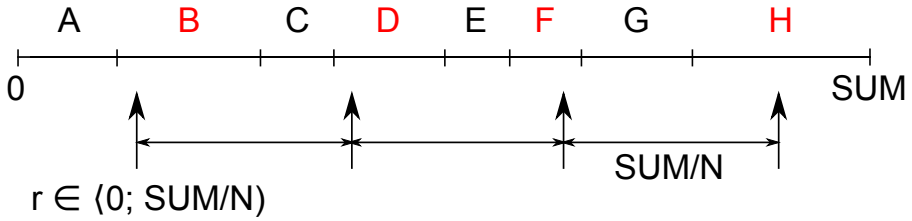
[2]Daný jedinec zabírá svým ohodnocením většinu nebo velkou část prostoru na ose výběru.



Obrázek 2.1: Ilustrace ruletového výběru.

procesů křížení. Tím může dojít k „degeneraci“ populace v takové míře, že už nebude možné dosáhnout globálního optima.

Problém ruletové selekce se snaží řešit již zmíněná metoda *stochastic universal sampling* (SUS). Jedná se o modifikaci původního algoritmu ruletového výběru, kdy namísto k-násobnému použití rulety pro výběr k jedinců ($k \geq 2$), spustíme algoritmus pouze jedenkrát s tím, že namísto použití jednoho ukazatele použijeme ukazatelů k. Jednotlivé ukazatele mají pevně definovaný rozestup. Princip tohoto algoritmu je ilustrován na obrázku 2.2.



Obrázek 2.2: Ilustrace metody SUS.

Tento způsob selekce již umožňuje udržet rozmanitost vybraných jedinců pro reprodukci na vyšší úrovni, ale neřeší hlavní problém většiny selekčních mechanismů. Tím je problém existence jedince s vysoce nadprůměrnou hodnotou fitness.

Potlačení vlivu nadprůměrných jedinců v populaci lze zajistit pomocí metody výběru dle pořadí. U této metody dochází k vzestupnému seřazení jedinců dle jejich ohodnocení, a výsledné pořadí je následně použito místo původní hodnoty fitness. Tím je zaručeno, že rozložení jedinců je rovnoměrné. Řešení tedy snižuje velké rozdíly případných vysoce nadprůměrných jedinců a zároveň zvyšuje rozdíly ke konci výpočtu, kdy jsou jednotlivá řešení již velmi dobrá a liší se méně. Tato výhoda se však může stát i nevýhodou v případě, kdy v populaci nejsou žádní výrazní jedinci, ale i přesto tento způsob výběru zveličí drobné rozdíly jedinců na násobky ohodnocení. Celkově pak dochází ke zpomalení evolučního procesu, neboť mají zdatnější jedinci silnější konkurenci, protože jsou zanedbány rozdíly ohodnocení jednotlivých jedinců. Díky pomalejší konvergencí je však vyšší pravděpodobnost nalezení globálního optima.

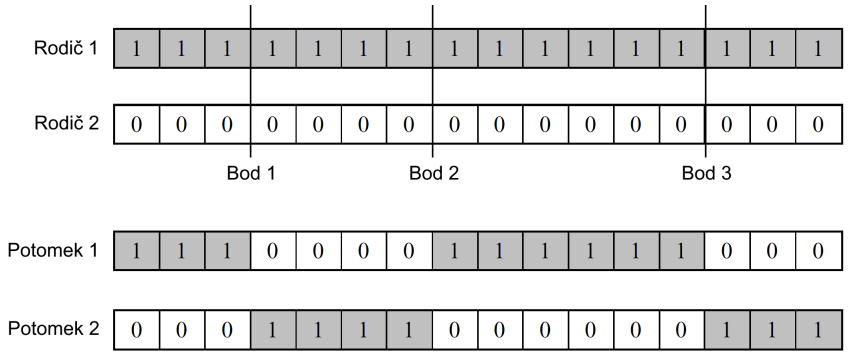
Turnajová metoda je další ze selekčních metod. Inspirována byla procesy živé přírody, kdy se jedinci spolu fyzicky utkávají o právo zúčastnit se reprodukce. Proces selekce je prostý: náhodně je zvolena skupina k jedinců ($k \geq 2$), ze kterých vyhrává ten, jenž má nejvyšší hodnotu fitness. Čím vyšší je hodnota k, tím vyšší je selekční tlak na populaci. Z tohoto důvodu bývá $k = 2$. Jelikož je turnajový způsob vlastně speciálním způsobem výběru dle pořadí, lze jej použít právě v situacích, kdy dochází k předčasné konvergenci.

Další výhodou je, že není u tohoto algoritmu nutné dodržovat nezápornost hodnot fitness funkce.

2.3.5 Křížení a mutace

Nejjednodušším typem křížení je tzv. jednobodové křížení, kdy jde náhodně vygenerován bod křížení, který rozděluje rodičovské chromozomy na dvě části. První potomek obdrží první část prvního rodiče a druhou část druhého rodiče. Druhý potomek pak naopak první část druhého rodiče a druhou část prvního rodiče.

Zobecněním tohoto křížení je tzv. k-bodové křížení, kdy je náhodně vygenerováno k bodů křížení, ale princip zůstává stejný. Princip tohoto křížení pro $k = 3$ ilustruje obrázek 2.3.



Obrázek 2.3: Ilustrace k-bodového křížení.

Zobecněním k-bodového křížení pak je tzv. uniformní křížení. Toto křížení pak stanovuje stejnou pravděpodobnost, že potomek zdědí daný gen od prvního nebo od druhého rodiče. Na rozdíl od předchozích dvou typů křížení zde nedochází k přenosu genetické informace v celistvé podobě a uniformní křížení má tedy spíše rozkladný vliv. Existují také různé modifikace tohoto křížení, kdy pravděpodobnost dědění genu není u obou rodičů stejná.

Zmíněné operátory křížení jsou poměrně univerzální vzhledem ke kódování, ale pokud je již využíváno zakódování chromozomů pomocí reálných čísel, existují i specifické operátory křížení pro takovéto chromozomy. Konkrétně se jedná o tyto operátory křížení:

- Aritmetický – nová alela je rovna aritmetickému průměru rodičovských alel.
- Geometrický – nová alela je rovna odmocnině součinu rodičovských alel.
- Rozšiřující – rozdíl mezi hodnotami rodičů je přičten k větší z nich a odečten od menší.

Stejně tak existují i speciální operátory mutace, kde na rozdíl od mutace pro binární vektory (negace daného bitu) může docházet k následujícím mutacím:

- Náhodná – nahrazení původní alely náhodně vygenerovanou hodnotou.
- Aditivní – přičtení náhodně vygenerované hodnoty $c \in \langle -\varepsilon, \varepsilon \rangle$ k původní alele.
- Multiplikativní – vynásobení původní alely náhodně vygenerovanou hodnotou $c \in \langle 1 - \varepsilon, 1 + \varepsilon \rangle$.

Hodnota ε je vždy zvolena dle potřeb řešeného problému.

2.3.6 Nahrazovací strategie

Genetické algoritmy používají tři základní typy nahrazovacích strategií. První z nich se nazývá generační. Principem této strategie je úplné nahrazení původní generace jejich potomky. Neexistuje tedy žádná možnost pro jedince ze starší populace, aby mohli přežít a ovlivňovat evoluční proces po delší dobu nežli jednu generaci. Hlavní riziko pak tkví v tom, že jedinec, či skupinka silných jedinců z původní generace může být ztracena, pokud se jim nepodaří projít procesem selekce nebo pokud bude jejich genetická informace příliš pozměněna genetickými operátory. Řešení tohoto problému je obsaženo ve zbývajících dvou strategiích. Tyto strategie využívají různý přístup k řešení zmíněného problému a to zejména co do počtu jedinců, kterým je umožněno ovlivňovat proces evoluce déle, nežli jednu generaci.

První přístup se nazývá elitismus a umožňuje ponechat vždy několik nejlepších jedinců z aktuální populace. Těmto jedincům se říká „elita“ a jsou bez jakýchkoliv úprav přeneseni do nové generace.

Druhý přístupem je nahrazovací strategie typu setrvalý stav (*steady-state reproduction*). Tato strategie naopak uchovává většinu populace v nezměněném stavu a nahrazuje pouze několik nejhůře hodnocených jedinců novými potomky. Počet těchto jedinců přitom nemusí být statický. Může být prováděno nahrazení pouze takovými jedinci, jejichž ohodnocení je vyšší než ohodnocení aktuálně nejslabšího jedince.

Kapitola 3

Neuronové sítě

Neuronové sítě [6] jsou jednou z metodologií softcomputingu, jehož koncept vymyslel L.A. Zadeh v devadesátých letech minulého století. Vývoj neuronových sítí však začal dříve, již ve čtyřicátých letech minulého století, kdy McCulloch a Pitts [20] zjistili, že k tomu, aby umělý neuron dokázal provádět logické funkce stačí, aby byl modelován jako jednoduché prahové zařízení. Na konci padesátých a začátku šedesátých let byl vytvořen panem Rosenblattem model perceptronu [29]. Ten je často považován jako základní matematický model neuronu.

Zájem o neuronové sítě poté opadl po publikaci knihy pana Minského a Paperta [24], která matematicky dokázala, že perceptron nelze použít pro řešení komplexních logických funkcí. Umělé neurony a neuronové sítě té doby totiž ještě nedokázaly řešit lineárně neseparovatelné problémy, jelikož umělé neurony používali lineární aktivační funkci.

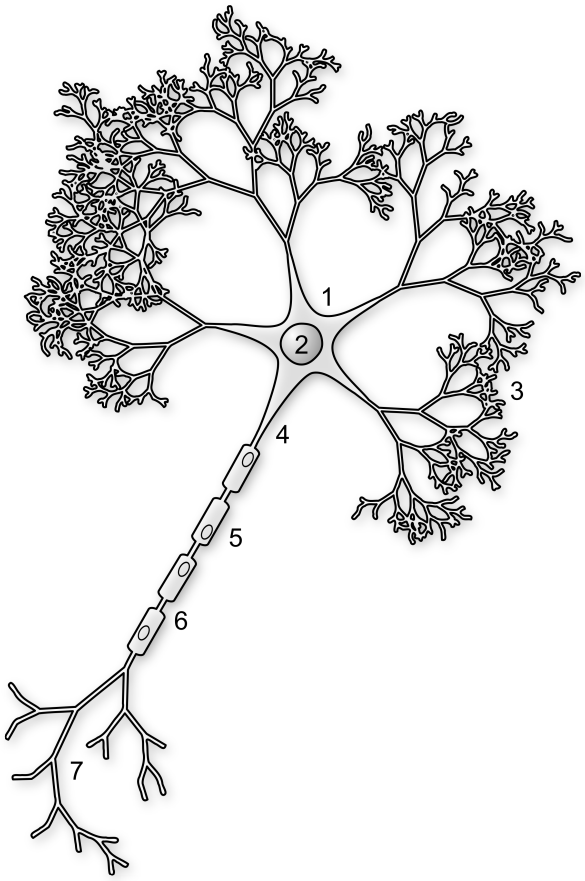
Zájem o neuronové sítě byl znovu obnoven až v druhé polovině osmdesátých let, kdy bylo dokázáno, že ačkoliv byl poznatek zmíněné knihy správný, byl nesprávně interpretován. Vznikl totiž model tzv. vícevrstvého perceptronu, který dokázal komplexní logické funkce vyřešit. Alternativní přístup pro řešení komplexních logických funkcí by také mohlo být použití vícevrstvých sítí, ale to autoři Minsky a Papert ve své knize nenavrhli.

Postupem času vzniklo mnoho typů neuronových sítí a algoritmů učení těchto sítí. Modely neuronových sítí se mohou velmi lišit a mohou mít různé aplikační oblasti dle svých charakteristik. Navíc se s postupem rozvoje neurobiologického vývoje čím dál více rozvíjí i modely neuronových sítí. A zatímco se většina inženýrů zabývá schopnostmi neuronových sítí řešit problémy, mnoho výzkumníků se věnuje modelování biologických neuronových systémů pomocí umělých neuronových systémů.

3.1 Biologický neuron

Základním stavebním prvkem nervové soustavy živých organismů je nervová buňka též nazývaná jako neuron [27]. Jedná se o samostatnou živou buňku specializující se na efektivní zpracování, uchování a přenos informací. Přestože byla zjištěna existence více druhů neuronů, jejich základní struktura je vždy stejná.
Struktura neuronu je zachycena na obrázku 3.1. Neuron se skládá z následujících částí:

1. Tělo neuronu – také se označuje jako *soma*.
2. Jádro neuronu – nachází se uvnitř těla neuronu.
3. Dendrity – vstupní rozhraní neuronu.



Obrázek 3.1: Biologický neuron [30].

4. Axon – výstupní rozhraní neuronu.
5. Myelinová pochva – nehomogenní ochranná vrstva axonu.
6. Ranvierovy zářezy – místa kde, je myelinová pochva zúžená.
7. Terminály axonu – zakončení axonu.

Stejně jako ostatní buňky živých organismů i neuron má polopropustnou membránu. Na této membráně pak vzniká tzv. membránový potenciál, který má zásadní význam pro přenos informací mezi neurony. Jeho hodnota v ustáleném stavu odpovídá $-70mV$.

Nervové buňky mají navíc schopnost podráždění elektrickým, chemickým nebo mechanickým podnětem. Podráždění může být dvojího charakteru, a sice lokální, které je vyvoláno podprahovým podnětem a dále se nešíří, a nebo dojde ke vzniku akčního potenciálu, který je reakcí na nadprahový podnět.

Při vzniku akčního potenciálu dojde k elektrické přepolarizaci membrány, takže potenciál v okolí podráždění nabude hodnoty asi $+35mV$. Tento vzruch však nezůstane pouze v místě podráždění, ale začne se šířit celým axonem. Po odeznění podnětu nastává v místě podráždění fáze hyperpolarizace, kdy je ve zmíněném místě dočasně snížena citlivost na další podráždění.

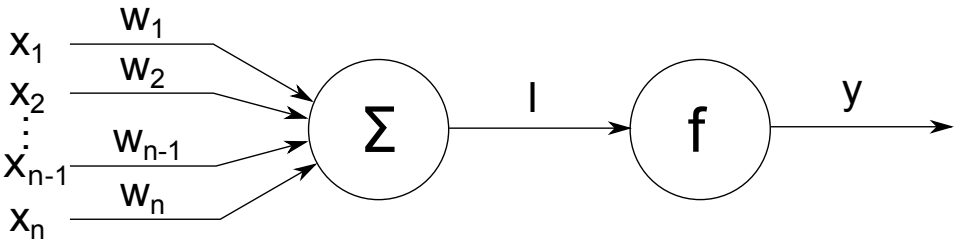
Pro přenos informací mezi neurony slouží tzv. chemické synapse. Ty se dle své funkce dělí na dva druhy:

- Excitační – budí a rozšiřují nervový vzruch v nervové soustavě.
- Inhibiční – tlumí nervový vzruch v nervové soustavě.

Míra vlivu synapse se může v čase řízeně nebo samovolně měnit. Konkrétně je slovem synapse označováno místo, kde jsou od sebe odděleny synaptickou štěrbinou dvě membrány patřící různým neuronům. Presynaptická membrána ukončuje axon předcházejícího neuronu a postsynaptická membrána je membrána dendritu nebo těla následujícího neuronu. Pro přenos informace se využívá tzv. neurotransmiteru, což je účinná látka, která se vylije z váčků v presynaptické membráně do synaptické štěrbiny a začne působit na postsynaptickou membránu. Jejím působením dojde ke vzniku elektrického proudu a díky tomu i k postsynaptické akční potenciálové vlně, čímž dojde k přenosu informace.

3.2 Umělý neuron

Umělý neuron [1] je abstrakcí biologického neuronu, přičemž hlavním cílem tohoto modelu je co nejvěrněji napodobit princip práce biologickému neuronu vlastní. Schematicky je tento model zobrazen na obrázku 3.2.



Obrázek 3.2: Umělý neuron.

Lze vidět, že neuron přijímá množinu vstupních signálů $x_1, x_2, ..., x_n$ tvořících vstupní vektor často označovaný písmenem X. Tyto vstupní signály jsou zpravidla výstupními signály jiných neuronů. Každý vstup je následně násoben odpovídající váhou spojení w_i, kde $1 \leq i \leq n$. Tato váha je analogií k míře vlivu synapse u biologického neuronu. Vážené vstupní signály následně vstupují do sumačního bloku, podobně jako u biologického neuronu vstupují do těla buňky, a provede se nad nimi algebraický součet, jehož výsledek vyjadřuje vnitřní potenciál neuronu. Formálně je vnitřní potenciál vypočítán dle následujícího vztahu

$$I = \sum_{i=1}^{n} x_i w_i \tag{3.1}$$

Výstupní signál neuronu je získán pomocí funkce f, do níž vstupuje vnitřní potenciál neuronu a nastavený práh neuronu θ

$$y = f(I - \theta) \tag{3.2}$$

Typicky nabývá funkce f následujících tvarů:

- Lineární funkce (k je konstanta)
$$y = k \cdot I \tag{3.3}$$
- Binární (prahová) funkce
$$y = \begin{cases} 1 & \text{pokud } I \geq \theta \\ 0 & \text{pokud } I < \theta \end{cases} \tag{3.4}$$
- Sigmoidální funkce
$$y = \frac{1}{1 + e^{-(I-\theta)}} \tag{3.5}$$

3.3 Umělé neuronové sítě

Neurony, které jsou propojeny mezi sebou a případně i s vnějším prostředím, vytvářejí neuronovou síť [1, 21]. Vstupní vektor X vstupuje do sítě aktivací vstupních neuronů. Váhy propojení neuronů jsou reprezentovány ve formě matice W, kde prvek w_{ij} popisuje váhu propojení mezi neurony i a j. Během pracovní fáze sítě je vstupní vektor transformován na výstupní průchodem sítí. Výpočetní silou sítě jsou tedy propojení a jejich váhy. Přitom propojení pouze spojují vstup jednoho neuronu s výstupem jiného a jejich síla je definována jejich váhou. Architektura sítě je reprezentována způsobem propojení jednotlivých neuronů. Dělí se na dva hlavní druhy:

- Plně propojená architektura – výstup každého neuronu je vstupem všech ostatních neuronů, ale i sebe samého; počet spojení je v této architektuře n^2, kde n je počet neuronů.
- Vrstvová (hierarchická) architektura – rozděluje neurony do skupin, tzv. vrstev; existují 3 typy vrstev: vstupní, výstupní a skrytá; vstupní a výstupní vrstva jsou v kontaktu s prostředím a interagují s ním.

Na obrázku 3.3 jsou ilustrovány následující typy architektur neuronových sítí:

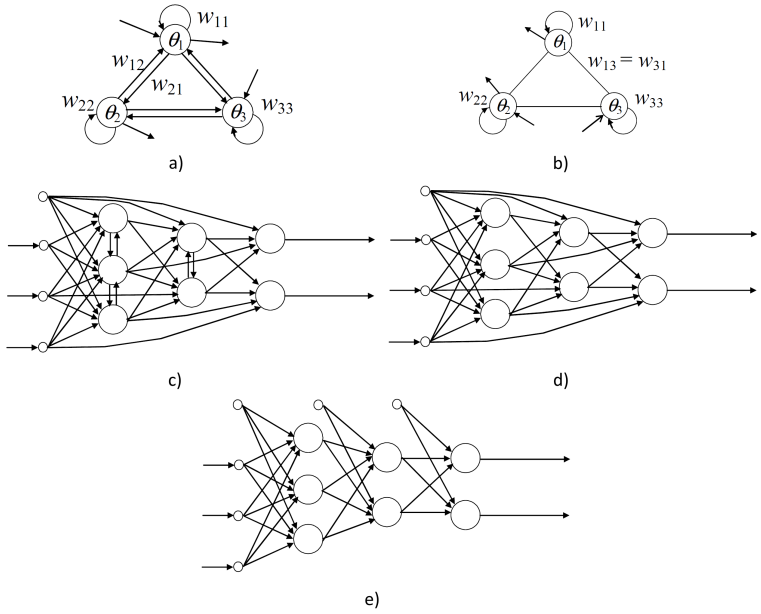
a) Plně propojená síť – každé spojení má svou definovanou váhu.

b) Plně propojená symetrická síť – plně propojená síť s platností vztahu $w_{ij} = w_{ji}$.

c) Vrstvová síť – neurony nemohou ovlivňovat svým výstupem neurony předchozích vrstev.

d) Acyklická síť – vrstvová síť, v níž neexistují spojení mezi neurony stejné vrstvy.

e) Dopředná síť – acyklická síť, v níž existují spojení pouze mezi neurony sousedních vrstev.



Obrázek 3.3: Typy neuronových sítí – (a) plně propojená síť, (b) plně propojená symetrická síť, (c) vrstvová síť, (d) acyklická síť, (e) dopředná síť [39].

3.3.1 Dopředná neuronová síť

Dopředné neuronové sítě nemají žádná zpětnovazební spojení. V těchto sítích, jak již bylo řečeno, přijímají neurony s-té vrstvy signály z prostředí (pokud $s = 1$) nebo od neuronů vrstvy $s - 1$ (pokud $s > 1$). Výstupy těchto neuronů pak zpracovávají neurony vrstvy $s + 1$ nebo prostředí (pokud je vrstva s poslední).

Tyto sítě mohou být jednovrstvé a vícevrstvé. Jednovrstvé sítě jsou sítě, které neobsahují žádnou skrytou vrstvu. To znamená, že tyto sítě obsahují pouze vstupní a výstupní vrstvu. Vstupní vrstva slouží k distribuci signálu každého vstupního neuronu všem neuronům výstupní vrstvy. Neurony výstupní vrstvy jsou výpočetní jednotky. Jejich výstupy jsou dány hodnotami aktivační funkce aplikované na vážený součet všech vstupních signálů. Aktivační funkce může být buď lineární nebo nelineární. Pokud je lineární, výstup sítě je vyhodnocován vztahem

$$Y = WX + \theta \tag{3.6}$$

kde W je vektor vah, X je vstupní vektor, Y je výstupní vektor a θ je prahová hodnota.

Použití nelineární aktivační funkce umožňuje zvýšit výpočetní sílu sítě. Výstup sítě při použití sigmoidální aktivační funkce je dán vztahem

$$Y = \frac{1}{1 - e^{-XW+\theta}} \tag{3.7}$$

Vícevrstvé neuronové sítě obsahují oproti jednovrstvým sítím navíc skryté vrstvy. Díky tomu mohou řešit komplexní problémy, protože skryté vrstvy umožňují zvýšit výpočetní sílu sítě. Výpočetní sílu lze také zvýšit vytvořením optimální struktury sítě nebo zvýšením počtu skrytých vrstev. Spolehlivost sítě lze mnohdy zvýšit optimální strukturou a zvýšením počtu neuronů ve skrytých vrstvách.

3.3.2 Rekurentní neuronová síť

Rekurentní neuronové sítě obsahují jak dopředné, tak zpětnovazební spojení mezi vrstvami neuronů, čímž vytváří komplikovaný dynamický systém. Díky schopnosti aproximovat spojitý nebo diskrétní nelineární dynamický systém pomocí systému nelineárních diferenciálních nebo diferenčních rovnic mohou rekurentní neuronové sítě nabídnout řešení pro problémy adaptivního řízení.

Z výpočetního hlediska může dynamická neurální struktura obsahující stavovou zpětnou vazbu nabídnout větší výpočetní výhodu, nežli neurální struktura čistě dopředná. Pro podobné problémy může být malý zpětnovazební systém ekvivalentní velkému (až nekonečnému) dopřednému systému.

Vícevrstvá rekurentní neuronová síť je kombinací dopředné a zpětnovazební neuronové sítě, kde je zpětná vazba reprezentována rekurentními spoji. Tato síť se skládá ze vstupní vrstvy, řady skrytých vrstev a vrstvy výstupní. V rámci sousedících skrytých vrstev jsou povoleny jak dopředné tak zpětnovazební spojení mezi neurony. Tyto sítě jsou schopny aproximovat nelineární dynamický systém.

3.4 Tradiční způsoby učení

Schopnost učit se [6] je pro neuronové sítě velmi důležitá. Algoritmy provádějící učení mají za úkol nalézt vhodné váhy spojení a případně i další parametry sítě. Jinými slovy je učení optimalizační proces, při kterém měníme parametry sítě tak, abychom dosáhli takových výstupů sítě, jenž jsou co možná nejblíže výstupům požadovaným.

Neuronové sítě jsou typicky trénovány/učeny v tzv. epochách. Epochou rozumíme úplný běh algoritmu učení, kdy je na vstup sítě přiložen každý vstupní vektor z trénovací množiny alespoň jedenkrát. Po dokončení procesu učení by síť měla být schopna generalizace a při použití nového neznámého vstupu by síť měla vytvořit správný výstup.

Algoritmy pro učení jsou rozděleny na 4 typy: učení s učitelem, učení bez učitele, posilované učení a evoluční učení. Učení s učitelem se často používá pro rozpoznávání vzorů, řízení, modelování a identifikaci, zpracování signálů a optimalizaci. Posilované učení je obvykle používáno pro problémy řízení. Algoritmy učení bez učitele jsou používány především pro rozpoznávání vzorů, shlukování, kvantizaci vektorů, kódování signálů a analýzu dat. Evoluční učení je poměrně univerzální, neboť provádí přizpůsobení architektury i parametrů sítě pomocí evolučního algoritmu. Také může být použito i pro optimalizaci řídících parametrů algoritmů učení s nebo bez učitele.

3.4.1 Učení s učitelem

Učení s učitelem je založeno na přímém porovnání výstupu sítě s požadovaným výstupem. Parametry sítě jsou upravovány na základě kombinace množiny trénovacích vzorů a jim odpovídajícím chybám. Chyby jsou vyjádřeny rozdílem požadovaného a skutečného výstupu. Učení s učitelem je tedy systém s uzavřenou zpětnovazební smyčkou, kde chyba představuje onu zpětnou vazbu.

Pro řízení procesu učení je zapotřebí stanovit podmínku, která určí, kdy dojde k ukončení tohoto procesu. U učení s učitelem se používá střední kvadratická chyba definovaná tímto vztahem

$$E = \frac{1}{N} \sum_{p=1}^{N} \|y_p - \hat{y}_p\|^2 \tag{3.8}$$

kde N je počet vzorů, y_p je výstupní část vzoru p a $\hat{y}_p$ je výstup sítě pro daný vzor p. Chyba E je pak počítána po každé epoše. Proces učení končí, jakmile je E dostatečně nízké.

K tomu, aby se E snižovalo k nule je často používána optimalizační metoda s názvem gradientní sestup. Metoda gradientní sestup vždy konverguje do lokálního minima v rámci blízkého okolí původního řešení, zde původních parametrů sítě. Tuto metodu používají například algoritmy LMS (*Least Mean Squares*) [8] a BP (*BackPropagation*) [2, 14], které jsou sice poměrně staré, ale přesto velmi používané algoritmy učení s učitelem.

3.4.2 Učení bez učitele

Učení bez učitele nepoužívá/nemá informace o požadovaných výstupech. Pokouší se autoasociovat vstupy tak, aby docházelo k podstatné redukci dimenzionality nebo množství dat. Učení bez učitele je založeno výhradně na korelacích vstupních dat a používá se k vyhledávání významných vzorů nebo vlastností vstupních dat bez pomoci učitele.

K ukončení procesu učení je opět zapotřebí stanovit podmínku zastavení. Bez této podmínky by proces učení pokračoval i v případě, že by byl síti předložen vstup, který již nepatří do množiny trénovacích vzorů. Síť by se snažila adaptovat na konstantně se měnící prostředí. Obecně však platí, že učení bez učitele se dostává do stabilního stavu velmi pomalu.

Nejpoužívanější přístupy učení bez učitele jsou: Hebbovo učení, soutěživé učení a Kohonenova SOM (*Self-Organizing Map*).

Hebbovo učení je čistě lokální a provádí se vždy pouze nad dvěma neurony a synapsí mezi nimi. Synaptická váha se mění proporcionálně ke korelaci mezi presynaptickými a postsynaptickými signály. Toto učení je aplikováno především na neuronové sítě používané pro PCA (*Principal Component Analysis*) nebo asociativní paměti.

V soutěživém učení spolu neurony výstupní vrstvy neuronové sítě soutěží. V jednoduchém soutěživém učení obsahuje neuronová síť jednu vrstvu výstupních neuronů používající pravidlo „vítěz bere vše“. Výstup neuronové sítě tedy udává pouze vítězný neuron výstupní vrstvy. Na tomto principu pak pracuje i Kohonenova SOM.

3.4.3 Posilované učení

Posilované učení je speciální forma učení s učitelem, kde není znám přesný požadovaný výstup. Je založeno pouze na informaci o tom, zda se aktuální výstup blíží nebo neblíží odhadu.

Posilované učení je učící procedura, která „odměňuje“ neuronovou síť za dobré výstupy a „trestá“ za výstupy špatné. Je používáno v případech, kdy není znám pro zadaný vstup správný výstup, ale přesto je zapotřebí určitý výstup vytvořit.

Vyhodnocení, zda je výstup sítě dobrý nebo špatný, závisí na konkrétním problému a prostředí. Pro řídící systém by pak platilo, že pokud pracuje kontrolér správně po zpracování vstupu, tak je jeho výstup považován za dobrý, jinak za špatný. Vyhodnocení výstupu je tedy binární a je nazýváno „vnější posílení“ (*external reinforcement*). Posilované učení je jistým druhem učení s učitelem s vnějším posílením jakožto chybovým signálem. Posilované učení provádí učení způsobem pokus-omyl a je vhodné i pro učení za běhu systému.

3.4.4 Evoluční učení

Evolučního učení je výhodné z toho důvodu, že dokáže lépe zvládnout prohledávací problémy v rozsáhlém, komplexním, multimodálním a nediferencovatelném prostoru. Je nezá-

vislé na informaci o gradientu chybové nebo fitness funkce, takže je vhodné pokud je tato informace nedostupná nebo velmi těžko zjistitelná.

Může být použito pro hledání optimálních řídících parametrů v učení s učitelem i bez učitele tím, že jsou optimalizovány jejich příslušné účelové funkce. Také může být použito jako nezávislá trénovací metoda pro parametry sítě pomocí optimalizace chybové funkce.

Výhoda těchto algoritmů je, že jsou velmi málo citlivé na počáteční podmínky, a díky tomu mají široké využití. Vždy provádí prohledávání s cílem nalezení globálně optimálního řešení, zatímco učení bez učitele i s učitelem jsou algoritmy, které obvykle dokážou najít pouze lokální optimum v prostoru blízkém počátečního řešení.

3.5 Aplikace neuronových sítí

Neuronové sítě mají díky své značné univerzálnosti a obecnosti mnoho potenciálních aplikačních domén [27]. V této sekci bude uvedeno pouze několik z mnoha možných aplikací neuronových sítí a bude nastíněno jak se v dané doméně neuronové sítě používají.

3.5.1 Komprese dat

Komprese dat [27, 38] se používá k redukci objemů dat a využívá se v mnoha různých doménách. Mezi nejdůležitější patří obrazové informace, které by byly bez použití komprese velmi objemné pro uchování i přenos. Pro kompresi dat existuje velké množství algoritmů, kde rozlišujeme dva základní typy: bezztrátová a ztrátová komprese. U běžných souborů je striktně vyžadováno, aby komprese i dekomprese byla bezztrátová. U některých dat, jako jsou například zvuková či obrazová data, však lze použít i kompresi ztrátovou. Ztráta užitečné informace pak musí být dostatečně malá, aby nedošlo k podstatnému zhoršení kvality výsledného celku.

Dále lze kompresi použít v rámci předzpracování dat v klasifikačních úlohách. Komprese má za úkol odstranit irelevantní a redundantní informace, čímž získáme kvalitnější vstupní data pro klasifikátor. I na tento typ komprese lze použít neuronovou síť, jelikož je schopna nalézt v datech i složité korelace a dosáhnout tím vysokého kompresního poměru.

V problematice komprese dat lze neuronové sítě využít různými způsoby. Jeden z přístupů spočívá ve vytvoření sítě s jednou skrytou vrstvou. Pokud jsou komprimovány m-dimenzionální vektory na n-dimenzionální ($n < m$), pak vstupní i výstupní vrstva obsahuje m neuronů a skrytá vrstva n neuronů. Po vytvoření takovéto sítě je provedeno její učení. V případě provádění komprese je použit výstup skryté vrstvy jako výstup sítě (odstranění výstupní vrstvy sítě). Pro dekompresi je použita skrytá vrstva jakožto vrstva vstupní (odstranění vstupní vrstvy sítě).

3.5.2 Predikce

Predikce zpravidla provádí stanovení jedné budoucí hodnoty dané proměnné. Někdy je vyžadováno budoucích hodnot více, nebo naopak pouze znalost trendu (rostoucí/klesající). Predikce trendu pak představuje vlastně klasifikaci do dvou tříd a úloha má jiný charakter, nežli predikce budoucí hodnoty. Obecně se predikce používá v meteorologii (počasí, stav vody v tocích), finančnictví (kurzy měn a akcií), energetice (spotřeba elektrické energie), zemědělství (velikost úrody plodin) a dalších oborech.

Povaha predikce závisí na datech, která jsou k dispozici:

- jednoduchá časová řada v ekvidistantních intervalech,

- jednoduchá řada s doplňujícími informacemi (průběh derivace, intervenční proměnné),
- několik časových řad veličin, které spolu souvisí,
- speciální výběr parametrů.

Důvod, proč se pro tuto aplikační doménu hodí neuronové sítě je ten, že tyto sítě mají schopnost učit se na příkladech a zachytit nelineární závislosti. Nevýhodou je, že lze jen velmi obtížně stanovit velikost chyby a interval spolehlivosti predikce. Většinou jsou tyto odhady prováděny pomocí heuristických postupů.

3.5.3 Klasifikace

Jednou z oblastí aplikace neuronových sítí je i klasifikace [22]. Klasifikace je problém identifikace, do jaké z existujících tříd patří klasifikovaný objekt. Řešením tohoto problému je použití klasifikátoru, jenž na základě učení z trénovací množiny objektů, u kterých je třída známá, dokáže tento problém rozhodnout.

Objekty jsou často převedeny na množinu kvantitativních rysů či atributů popisujících daný objekt. Tyto rysy mohou být:

- Kategorické:
  - Nominální – výčet nečíselných hodnot; do počtu neomezený; bez jakéhokoliv uspořádání.
  - Binární – dvouprvkový výčet nečíselných hodnot:
    - Symetrické – obě možné hodnoty mají stejnou váhu/důležitost.
    - Asymetrické – jedna z hodnot má vyšší váhu/důležitost.
  - Ordinální – výčet nečíselných hodnot; do počtu neomezený; s definovaným lineárním uspořádáním $\leq$.
- Numerické:
  - Celočíselné – omezené pouze na spočetnou množinu celých čísel.
  - Reálné – čísla s pohyblivou řádovou čárkou.

Pokud tento převod klasifikátor neprovádí, porovnávání objektů může být provedeno i jako přímé porovnávání, nebo pomocí funkce vzdálenosti.

Klasifikace je, stejně jako regrese a shlukování, příkladem obecnějšího problému s názvem rozpoznávání vzorů. Klasifikace se dále dělí na následující typy:

- Unární (*one-class*) klasifikace – provádí *identifikaci* objektů dané třídy mezi objekty všech tříd, díky učení se z trénovací množiny, která obsahuje pouze objekty dané třídy; tato úloha je obtížnější oproti klasifikacím s dvěma a více třídami, kde trénovací množina obsahuje objekty všech tříd [26].
- Binární klasifikace – provádí klasifikaci objektů dané množiny do jedné ze dvou tříd.
- Multinomiální (*multiclass*) klasifikace – provádí klasifikaci objektů dané množiny do jedné z více než dvou tříd.
- Pravděpodobnostní klasifikace – provádí predikci míry příslušnosti daného vstupu do jednotlivých tříd; jedná se tedy o klasifikaci s určitou mírou nejistoty [12].

- *Multi-label* klasifikace – provádí klasifikaci objektu do více tříd současně; formálně se jedná o problém hledání modelu mapující vstupy x na binární vektory y, namísto skalárních výstupů [36].

Binární klasifikace

Binární klasifikace [28] je speciálním typem klasifikace multinomiální, kdy máme k dispozici pouze dvě známé třídy, do kterých je klasifikace prováděna. Typickými úkoly binární klasifikace jsou například klinické testy, kdy je zjišťováno, zda daný pacient trpí určitou chorobou či nikoliv, nebo testy kvality v továrnách, kdy se zjišťuje, zda výrobek nemá vady a pracuje tak jak má.

U problémů řešených touto klasifikací je velmi důležité, jestli je váha použitých tříd symetrická nebo asymetrická. Ve velkém množství praktických případů platí nesymetričnost použitých tříd, a proto je přesnost klasifikátorů ovlivněna chybou typu I jinak, nežli chybou typu II. Příkladem mohou být klinické testy, kdy chyba typu I (*false positive*) je zvažována mírněji, nežli chyba typu II (*false negative*). Chyba typu I totiž představuje chybné označení zdravého pacienta za nemocného, kdežto chyba typu II představuje chybné označení nemocného pacienta za zdravého.

Ve spojitosti se zmíněnými typy chyb jsou pro binární klasifikátory zadefinovány různé metriky vyhodnocování výkonu klasifikátoru. Po provedení klasifikace jsou známé 4 základní hodnoty:

- *True Positives* (TP) – správně klasifikované kladné výsledky.
- *True Negatives* (TN) – správně klasifikované zamítavé výsledky.
- *False Positives* (FP) – nesprávně klasifikované kladné výsledky.
- *False Negatives* (FN) – nesprávně klasifikované zamítavé výsledky.

Tyto data pak tvoří tabulku o velikosti 2x2 a lze z ní vypočítat 8 metrik. Čtyři jsou různé a každá z nich má svoji komplementární metriku, kterou lze získat odečtením příslušné metriky od hodnoty 1. Hlavními metrikami jsou:

- *True Positive Rate* (citlivost) $TPR = \frac{TP}{TP+FN}$.
- *True Negative Rate* (specifičnost) $TNR = \frac{TN}{FP+TN}$.
- *Positive Prediction Value* (přesnost) $PPV = \frac{TP}{TP+FP}$.
- *Negative Prediction Value* $NPV = \frac{TN}{TN+FN}$.

Při testování jsou nejdůležitější TPR a TNR metriky a při zhodnocení získávání informací metriky PPV a TPR.

Kapitola 4

Evoluční návrh neuronových sítí

Neuronové sítě jsou velmi univerzální, a to především z hlediska aproximace funkcí. Dokáží řešit i velmi komplexní problémy, avšak s růstem komplexnosti problému roste také velikost potřebné neuronové sítě. S růstem velikosti sítě pak roste i obtížnost učení dané sítě, jelikož obsahuje mnohem více spojení, jimž je zapotřebí přiřadit vhodnou váhu. Standardní algoritmy provádějící učení jsou velmi zdlouhavé a mohou snadno uváznout v lokálním optimu i u průměrně obtížných problémů, jako je vyvážení inverzního kyvadla s jednou tyčí. To je samozřejmě nežádoucí, a proto jsou hledány alternativní cesty k vytváření a učení těchto sítí [35].

Efektivním řešením je použití automatizované metody, která díky použití evolučního algoritmu vhodně vytvoří topologii a nastaví váhy. Aplikací evolučního algoritmu lze provést evoluci vah, parametrů a topologie neuronové sítě a řešit tak vysoce komplexní problémy. Systémy schopné provádět takovouto evoluci se nazývají TWEANNs (*Topology and Weight Evolving Aritficial Neural Networks*) [35].

4.1 NeuroEvolution of Augmenting Topologies (NEAT)

NEAT [35] je neuroevoluční algoritmus vytvořený pro evoluční návrh neuronových sítí, použitých pro řešení složitých řídících a sekvenčních rozhodovacích úkolů. Tento algoritmus patří do třídy algoritmů TWEANNs. NEAT se však od ostatních algoritmů třídy TWEANNs odlišuje tím, že vhodně řeší 3 základní problémy související s jejich činností:

1. Konkurující si konvence:
   - Možnost vyjádřit stejné řešení problému více jak jedním způsobem.
   - Křížení takovýchto řešení obvykle produkuje potomky obsahující pouze částečné řešení.
2. Ochrana inovace:
   - Inovace sítě vzniká, když mutace způsobí přidání nové struktury do sítě, což obvykle vede ke snížení fitness hodnoty jedince.
   - Inovace se mohou pozitivně projevit až po čase, kdy se k těmto inovacím vyvinou vhodná propojení včetně vah; tento čas ale obvykle díky snížené fitness hodnotě nedostávají.

3. Počáteční populace:

    - Počáteční populace bývá náhodná, takže se zde objevují i nerealizovatelné neuronové sítě a je zde problém minimalizace, protože se z rozsáhlých sítí typicky nedaří vytvářet menší, ale stále popisující řešení daného problému.

Přestože je algoritmus NEAT původně vystavěn pro řešení složitých řídících a rozhodovacích problémů, lze nalézt jeho aplikaci například i v počítačové grafice. Konkrétně v částicových systémech, kde dokáže v kombinaci s IEC (*Interactive Evolutionary Computation*) vytvářet částicové efekty dle uživatelských preferencí. Výhoda tohoto přístupu je, že vytvářené částicové efekty nemusí uživatel matematicky, fyzikálně ani programově popsat. Využitím IEC uživatel volí, který z nabízených jedinců aktuální populace nejvíce odpovídá jeho požadavkům a představám o finálním produktu. V procesu tedy roli fitness funkce přebírá člověk. NEAT se zde používá pro evoluci neuronových sítí sloužících pro výpočty změn pozic částic mezi jednotlivými snímky animace [13].

4.1.1 Řešení problémů TWEANNs

V následujících sekcích bude popsáno řešení výše popsaných problémů TWEANNs v rámci algoritmu NEAT.

Konkurující si konvence

K řešení problému konkurujících si konvencí používá algoritmus NEAT tzv. historické značky. Tyto značky popisují historický vznik každého genu v populaci. Využívá se předpokladu, že dva geny, jenž mají stejný historický původ, musí reprezentovat stejnou strukturu, jelikož byly odvozeny ze stejného genu v určitém bodě v minulosti.

K vytváření těchto historických značek není zapotřebí složitých výpočtů, neboť stačí přiřazovat nově vytvářeným genům vždy inkrementovanou hodnotu *globálního čísla inovace*. Tato čísla pak představují chronologickou posloupnost výskytu jednotlivých genů v populaci. Konkrétní řešení pak spočívá v použití těchto historických značek při křížení. Více viz v sekci 4.1.4, která se zabývá křížením.

Alternativním způsobem řešení tohoto problému je analýza topologie a provedení křížení takovým způsobem, aby k uvedenému problému nedocházelo. Tato varianta je však mnohem více výpočetně náročná.

Ochrana inovace

K ochraně inovace používá algoritmus NEAT speciaci. Speciace je evoluční proces, při kterém se reprodukčně izolované populace vyvíjejí tak, aby vytvořily různé druhy. Umožňuje tudíž zajistit, aby jedinci soupeřili pouze v rámci skupiny podobných jedinců namísto s celou populací. Hlavní myšlenkou je rozdělit populaci na skupiny podobných topologií. Tento úkol by však znamenal provádět porovnávání topologií. Tomu se však lze vyhnout pomocí historických značek.

Počet přebývajících a disjunktních genů (viz 4.1.4) mezi párem chromozomů je přirozeným měřítkem jejich podobnosti. Čím více jsou chromozomy disjunktní, tím méně evoluční historie spolu sdílejí a jsou méně kompatibilní. Kompatibilitu topologií v algoritmu NEAT lze vypočítat pomocí následujícího vztahu

$$\delta = \frac{c_1 E}{N} + \frac{c_2 D}{N} + c_3 \cdot \overline{W} \tag{4.1}$$

Koeficienty c_1, c_2 a c_3 umožňují upravovat důležitost každého ze tří použitých činitelů, E vyjadřuje počet přebývajících genů, D je počet disjunktních genů, $\overline{W}$ je průměrný rozdíl vah odpovídajících si genů a N odpovídá počtu genů ve větším z porovnávaných chromozomů.

Kompatibilita chromozomů δ pak umožňuje provést speciaci díky stanovení prahu kompatibility δ_t. Chromozomy jsou mezi sebou porovnávány, a pokud je jejich kompatibilita menší než stanovený práh, pak jsou dané chromozomy umístěny do stejné skupiny. Každý chromozom je umístěn do první skupiny, jenž splní danou podmínku, takže nedojde k tomu, že by byl součástí více skupin současně.

Počáteční populace

Algoritmus NEAT nepoužívá standardní generování počáteční populace, jako velké množství algoritmů TWEANNs. Tím je vygenerování náhodné počáteční populace obsahující různé sítě, které mohou být jednak nerealizovatelné, nebo příliš složité. Snižování složitosti takovýchto sítí je velmi obtížné, protože čím více spojení síť obsahuje, tím více dimenzí musí být prohledáno, aby byla síť optimalizována.

NEAT provádí hledání řešení v rámci prostorů s minimem dimenzí díky jednotné počáteční populaci (všichni jedinci počáteční populace jsou topologicky stejní) a každý jedinec počáteční populace obsahuje právě jeden neuron skryté vrstvy (propojení vstupní vrstvy se skrytou a skryté vrstvy s výstupní je úplné). Nová struktura sítě je přidávána inkrementálně pomocí strukturálních mutací a přežije pouze taková, jenž se projeví jako užitečná. To znamená, že každá vzniklá struktura odůvodněná a funkční, neboť je ohodnocena fitness funkcí.

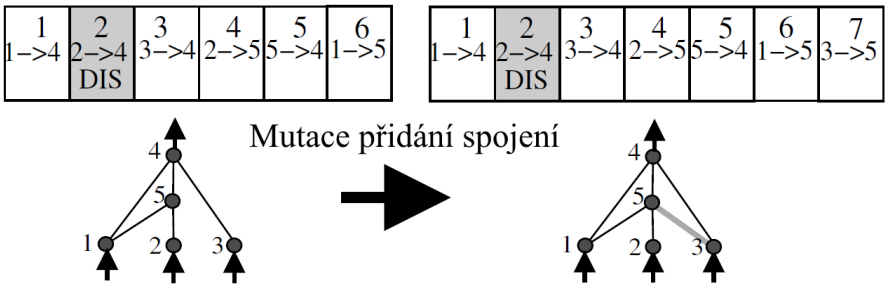
Díky této minimální počáteční populaci je minimalizován i prohledávaný prostor a algoritmus NEAT tedy prohledává méně dimenzí, než ostatní TWEANNs algoritmy, či algoritmy používající fixní topologie. Také je zvýšena pravděpodobnost nalezení řešení s jednodušší topologií, protože prohledávání začíná od nejjednodušších sítí.

4.1.2 Kódování

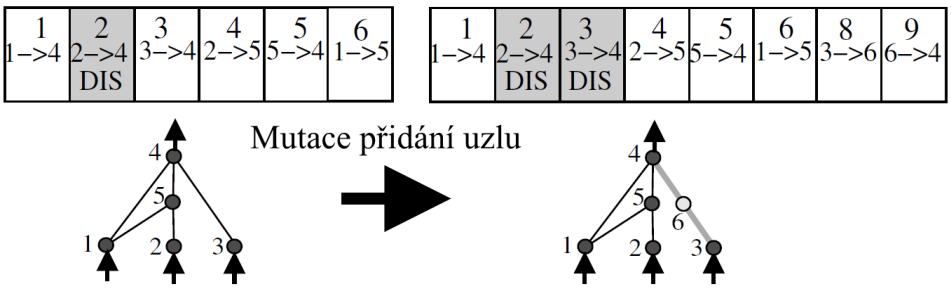
Zakódování neuronové sítě v algoritmu NEAT je navrženo tak, aby bylo možné provádět křížení co nejjednodušším způsobem. Chromozomy jsou lineární reprezentací síťových propojení. Každý chromozom obsahuje seznam genů popisujících spojení mezi dvěma neurony. Všechny takovéto geny specifikují vstupní uzel, výstupní uzel, váhu spojení, aktivnost genu a *číslo inovace*, neboli historickou značku.

Mutace pak může měnit jak váhy spojení, tak strukturu sítě. Strukturální mutace mohou nastat ve dvou formách. Obě však znamenají zvýšení velikosti chromozomu přidáním genu nebo genů. První formou strukturální mutace je přidání spojení, kdy vznikne nový gen popisující spojení dvou dříve nespojených uzlů. Princip této mutace je znázorněn na obrázku 4.1. Druhá forma strukturální mutace přidává nový uzel. Toho je docíleno rozdělením stávajícího spojení a přidáním nového uzlu na místo, kde se spojení nacházelo. Původní spojení je deaktivováno a namísto něj jsou přidána dvě nová spojení. Díky tomu je nový uzel okamžitě integrován do sítě a může dojít k optimalizaci spojení s ním souvisejících. Provedení mutace přidávající do neuronové sítě nový uzel je znázorněno na obrázku 4.2.

Chromozomy v algoritmu NEAT nemají konstantní velikost. S přibývajícím počtem strukturálních mutací dochází k jejich zvětšování. Výsledkem je tedy populace tvořená chromozomy různých délek, někdy s úplně odlišnými spojeními na stejných pozicích chromozomu.



Obrázek 4.1: Ilustrace principu mutace přidání spojení [35].



Obrázek 4.2: Ilustrace principu mutace přidání uzlu [35].

4.1.3 Ohodnocování

Při ohodnocování dochází k tzv. explicitnímu sdílení hodnoty fitness. To znamená, že jedinci musí sdílet fitness jejich skupiny. Tím je zajištěno, že se žádná skupina nemůže stát příliš početnou, přestože bude nabízet velmi kvalitní řešení daného problému. Díky tomu je velmi nepravděpodobné, že by mohla nějaká skupina převzít celou populaci, což je stěžejní pro současnou evoluci více skupin. Současná evoluce více skupin slouží k zachování diverzity v populaci.

Fitness hodnota jedince je vypočtena pomocí následujícího vztahu

$$f_{ij} = \frac{f_i}{N_j} \tag{4.2}$$

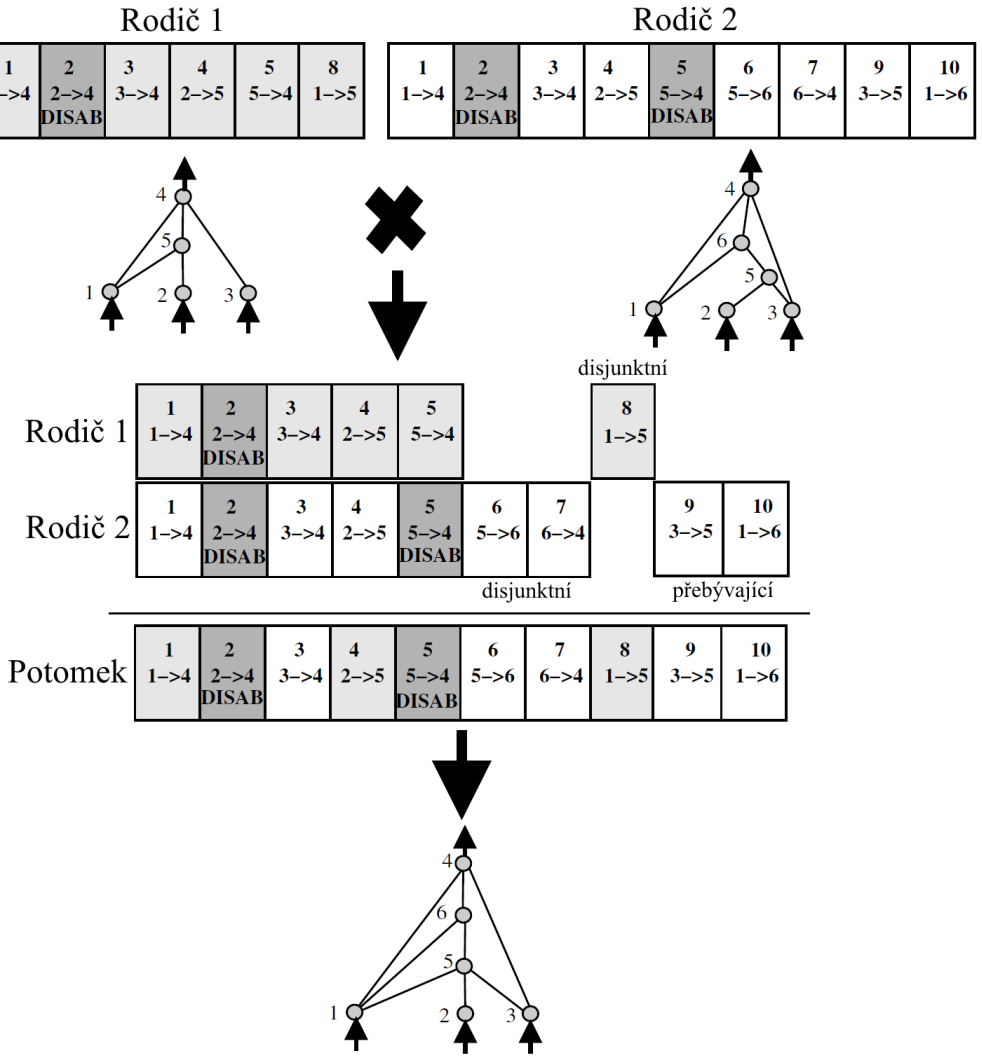
V uvedeném vztahu je upravená hodnota fitness jedince i skupiny j označena f_{ij}. Dílčí hodnoty potřebné na výpočet jsou neupravená fitness hodnota jedince i označena jako f_i a počet jedinců skupiny j označen jako N_j. Tím je zajištěno, že bude skupina růst nebo se zmenšovat v závislosti na tom, zda je průměrná hodnota upravené hodnoty fitness jedinců vyšší či nižší, než průměrná hodnota populace

$$N'_j = \frac{\sum_{i=1}^{N_j} f_{ij}}{\overline{f}} \tag{4.3}$$

Hodnota N'_j a N_j označuje nový a starý počet jedinců skupiny j. Aritmetický průměr upravené hodnoty fitness celé populace je označen jako $\overline{f}$.

4.1.4 Křížení

Při křížení jsou geny v chromozomech seřazeny dle historický značek. Geny, které mají stejnou historickou značku a nachází se v obou chromozomech, se nazývají odpovídající si. Ostatní geny se pak dělí na geny disjunktní a přebývající v závislosti na tom, zda se vyskytují uvnitř či vně rozsahu historických značek druhého rodiče. Při vytváření potomka je z odpovídajících si genů vybíráno náhodně (jako v uniformní vyhledávání s pravděpodobnostmi 0,5), zatímco u přebývajících nebo disjunktních genů se použijí vždy ty, které obsahuje rodič s vyšší hodnotou fitness. Pokud mají oba rodiče stejnou hodnotu fitness, pak jsou použity disjunktní i přebývající geny obou rodičů. Celý proces křížení je ilustrován na obrázku 4.3.



Obrázek 4.3: Ilustrace principu křížení [35].

4.2 Další přístupy evolučního návrhu neuronových sítí

Algoritmus NEAT není jediný, který provádí evoluční návrh neuronových sítí. Alternativním algoritmem je například jeho upravená varianta HyperNEAT, která se soustředí především na vytváření velmi rozsáhlých neuronových sítí. Další alternativou je také kartézské genetické programování, které lze s výhodou jednoduchosti celého konceptu použít i pro účely evoluce neuronových sítí. Algoritmus ESP je další z alternativ pro vývoj topologie i váhy propojení současně, přičemž tento algoritmus je velmi výkonný, a přestože je NEAT rychlejší, je to stále zajímavá alternativa.

V následujících sekcích budou výše uvedené algoritmy stručně popsány a zhodnoceny.

4.2.1 HyperNEAT

Algoritmus HyperNEAT (*Hypercube-based NeuroEvolution of Augmenting Topologies*) [34] je založen na myšlence, že reálná neuronová síť (například lidský mozek) se oproti umělým neuronovým sítím liší především ve složitosti. Nejvíce pak v komplexnosti propojení, kde reálná síť dosahuje až 10^{14} různých spojů mezi neurony. K těmto hodnotám se umělé neuronové sítě obvykle ani nepřibližují. Klíčem k dosažení takto vysoké složitosti je znovupoužitelnost genů při sestavování fenotypu. Konkrétně těch, jenž popisují propojení neuronů.

HyperNEAT je tedy navržen k evoluci rozsáhlých neuronových sítí. K tomu, aby mohl reprezentovat tyto sítě kompaktním způsobem používá tzv. substrát a spojovací síť CPPN (*Compositional Pattern Producing Network*).

Substrát je tvořen pravidelnou mřížkou rovnoměrně rozmístěných uzlů (neuronů). Jednotlivé uzly jsou na něm „adresovány“ souřadnicemi v intervalu $\langle -1; 1\rangle$ ve všech jeho dimenzích. Substrát tvoří kartézský prostor se středem v bodě $[0; 0; 0]$, případně $[0; 0]$ pro substrát v rovině.

CPPN je neuronová síť umožňující reprezentovat vzory propojení jako funkce kartézského prostoru. Tvoří tedy nepřímé kódování pro prostorové vzory. Vstupem CPPN jsou dva uzly substrátu a výstupem je váha propojení mezi nimi. Chromozom popisující síť CPPN tedy představuje geny, které jsou opakovaně použity pro popis propojení uzlů výsledné sítě. CPPN reprezentuje propojení konceptuálně, tzn. nezávisle na množství neuronů v substrátu (rozlišení substrátu).

Algoritmus pak pracuje tak, že sítě CPPN podléhají evoluci pomocí algoritmu NEAT za účelem zakódování propojení rozsáhlých neuronových sítí. Toho dosahují díky objevení pravidelností ve spojích mezi neurony. Nalezené prostorové vzory vykazují důležité geometrické motivy jako je symetrie, nedokonalá symetrie, opakování a opakovaní s variací.

Důležitá výhoda metody HyperNEAT je možnost provedení evoluce CPPN se substrátem nepříliš velkého rozlišení, a poté škálování tohoto řešení dle potřeby (zvyšováním rozlišení substrátu). Typicky totiž není zapotřebí provádět dodatečnou evoluci CPPN, protože řešení reprezentované CPPN je škálovatelné.

4.2.2 Kartézské genetické programování (CGP)

Kartézské genetické programování [23] je kartézské ve smyslu používání mřížky uzlů adresovaných v kartézském souřadném systému. Program CGP se označuje P a je definován jako n-tice $\{G, n_i, n_o, n_n, F, n_f, n_r, n_c, l\}$, kde G reprezentuje genotyp, což je množina celých čísel reprezentujících indexované vstupy programu n_i, vstupy a funkce uzlů n_n a výstupy programu n_o. Množina F reprezentuje n_f funkcí uzlů. Počet uzlů v řádku a sloupci pak definují n_r a n_c. Poslední parametr (l) definuje vzájemnou propojitelnost uzlů grafu. Určuje

kolik předchozích sloupců uzlů může mít svůj výstup připojen na vstup uzlu v aktuálním sloupci. Výjimku tvoří vstupy programu, jež mohou být připojeny na vstup jakéhokoliv uzlu. Uzly ve stejném sloupci nemohou být propojeny mezi sebou. Nejvíce volně propojená síť vzniká při parametrech $n_r = 1$ a $l = n_c$. Obvykle je konektivita pouze dopředná, ale lze dosáhnout i sekvenčního zpracování umožněním připojení vstupů programu na výstup skrze zpožďovací člen.

Délka genotypu G, jenž je mapován na konkrétní program, je fixní a je rovna $n_r \cdot n_c \cdot (n_n + 1) + n_o$. Tato délka ale omezuje pouze maximální velikost výsledného programu. Skutečná velikost programu může být menší, díky tomu, že některé uzly jsou neaktivní a nemají vliv na výstup programu.

Při vytváření genotypů nebo při jejich mutaci musí být dodržena definovaná omezení, aby výsledný genotyp reprezentoval validní program. Příkladem takovýchto omezení může být například: $0 \leq f < n_f$. Toto omezení vyjadřuje, že hodnota označující funkci bloku musí být v intervalu definovaných funkcí číslovaných od nuly. Díky tomuto, a dalším omezením, lze bezpečně provádět mutace i křížení bez problémů s validitou výsledných programů.

CGP se často používá bez křížení a algoritmus pak vypadá takto:

1. Vygenerování náhodné počáteční populace (při dodržení definovaných omezení).
2. Vyhodnocení fitness hodnot všech genotypů v populaci.
3. Povýšení nejlepšího genotypu do nové populace.
4. Zaplnění zbylých míst v populaci pomocí mutací nejlepšího genotypu.
5. Návrat na druhý krok, dokud není splněno ukončovací kritérium.

Pokud ve třetím kroku nevznikl žádný nový genotyp s vyšší fitness hodnotou, pak je vybráno náhodně z těch genotypů, jenž mají stejnou a zároveň nejvyšší hodnotu fitness v populaci. Takovému výběru se říká neutrální výběr.

Pro evoluci neuronových sítí pomocí CGP se může používat jak přímé, tak i nepřímé kódování genotypů. Pří použití přímého kódování FCGPANN (*Feed forward CGP evolved ANN*) [18] každý uzel grafu reprezentuje jeden neuron. Funkce daného uzlu pak představuje aktivační funkci neuronu a každý vstup uzlu je popsán třemi hodnotami: odkaz na vstupní neuron, váha spojení mezi vstupním a popisovaným neuronem a hodnota udávající aktivitu tohoto spojení. Zmíněné kódování se používá pro zakódování dopředných neuronových sítí. Existuje také obdobné kódování pro zakódování rekurentních neuronových sítí RCGPANN (*Recurrent CGP evolved ANN*) [18].

Příkladem nepřímého zakódování je například *Developmental Network* [17]. Toto kódování využívá evolučních principů k nalezení výpočetních funkcí analogických subkomponentám biologických neuronů. Použitý model umožňuje vznik, růst i zánik neuronů, dendritů a axonálních zakončení v závislosti na zpracování informací v rámci zadaného výpočetního problému.

4.2.3 SANE a ESP

Algoritmus SANE (*Symbiotic, Adaptive Neuro-Evolution*) [10] se oproti ostatním neuroevolučním algoritmům liší tím, že provádí evoluci populace neuronů. Nikoliv celých sítí, jak tomu běžně bývá. Tyto neurony jsou následně kombinovány tak, aby vytvořily skryté vrstvy dopředných neuronových sítí. Výsledné sítě jsou poté vyhodnocovány na zadaném problému.

Algoritmus se skládá ze 4 základních kroků:

1. Inicializace – Specifikuje se počet neuronů u, jenž budou tvořit skrytou vrstvu výsledných neuronových sítí. Také je provedena inicializace chromozomů reprezentujících neurony.

2. Vyhodnocení – Náhodně se vybere množina obsahující u neuronů pro vyplnění skryté vrstvy dopředné sítě. Síť je následně podrobena testování na zadaném problému a ohodnocena fitness hodnotou. Tato hodnota je poté přičtena ke kumulativní fitness hodnotě každého z použitých neuronů. Tento proces se opakuje, dokud každý neuron není vyhodnocen ve vytvářené síti přibližně desetkrát.

3. Rekombinace – Vypočítá se průměrná fitness hodnota každého neuronu (kumulativní fitness dělená počtem vyhodnocení neuronu). Na základě této hodnoty jsou neurony seřazeny a následně rekombinovány. Rekombinují se neurony horního kvartilu vždy s neuronem zařazeným výše. Rekombinace se provádí pomocí jednobodového křížení a mutace. Potomci poté nahrazují spodní polovinu jedinců v populaci.

4. Opakování – Opakovaně se provádí cyklus vyhodnocení–rekombinace, dokud není nalezena síť, řešící zadaný problém dostatečně dobře.

V algoritmu SANE neurony soutěží v závislosti na tom, jak dobře (v průměru) fungují neuronové sítě, jichž jsou součástí. Vysoká průměrná hodnota fitness značí, že neuron přispívá k vytváření kvalitních sítí a vhodně spolupracuje s ostatními neurony.

V algoritmu ESP (*Enforced Sub-populations*) [10] vycházejícího z algoritmu SANE, je populace opět tvořena neurony namísto celými sítěmi. Tyto neurony pak tvoří podmnožiny o velikosti u, ze kterých jsou celé sítě vytvářeny. Na rozdíl od SANE však ESP vytváří pro každý z u neuronů oddělenou populaci tak, aby mohl být daný neuron rekombinován pouze se členy jeho subpopulace. ESP urychluje algoritmus SANE ze dvou důvodu:

1. Subpopulace, jenž se v SANE postupně vytváří, jsou v ESP vytvořeny přímo, což je dáno algoritmem. Díky tomu se mohou neurony lépe a rychleji specializovat, protože nejsou rekombinovány s neurony jiné specializace.

2. Neurony jsou vždy vyhodnocovány na základě toho, jak dobře plní svoji roli. V algoritmu SANE mohou sítě obsahovat více neuronů se stejnou specializací a naopak mohou být některé specializace vybranými neurony nezastoupené. Proto může docházet k nekonzistentnímu vyhodnocování fitness hodnot.

Hlavní přínos ESP je ale možnost evoluce rekurentních sítí. Díky tomu, že SANE vytváří výsledné sítě náhodným výběrem neuronů z jedné populace, nemohou neurony spoléhat na to, že budou použity v kombinaci s podobnými neurony v jakýchkoliv dvou vytvářených sítích. Proto se může neuron chovat velmi odlišně při ohodnocování různých sítí. Fitness hodnoty pak jsou velmi nestálé. Naproti tomu architektura používající subpopulace (ESP) vytváří pro neurony více konzistentní podmínky pro vyhodnocování. Zpětnovazební spojení neuronu pak bude vždy připojeno k neuronu z určité subpopulace a váha této vazby se může vhodně upravit díky dodatečné specializaci připojené subpopulace. Rekurentní sítě lze tedy vytvářet právě díky stálosti spojení zpětných vazeb k určité subpopulaci neuronů.

Kapitola 5

Předzpracování

Složité klasifikační problémy řešené neuronovou sítí s vysokou přesností mají společný problém, kterým je vysoká složitost použitých sítí a s ní související vysoké výpočetní nároky. Pokud takový klasifikační problém navíc vyžaduje zpracování velkého množství vstupních dat, znamená to zpravidla dlouhou dobu potřebnou pro zpracování těchto dat. Hlavní myšlenkou navrhovaného řešení je genetický návrh klasifikátoru ve formě neuronové sítě podstatně méně komplikované, nežli konvenční neuronové sítě řešící stejný klasifikační problém. Tohoto zjednodušení je dosaženo tím, že vytvářená síť nemusí zpracovávat celou datovou sadu zadaného problému. Neuronová síť tedy provádí předzpracování vstupní datové sady, při kterém musí být zajištěna maximální možná přesnost. Zbývající neoklasifikované vzorky jsou poté vstupem sítě představující konvenční řešení. Síť je sice podstatně složitější, ale dokáže provést zpracování všech zbývajících vzorků datové sady s vysokou přesností. Cílem je u složitých klasifikačních problémů dosáhnout úspory času, a tudíž i energie spotřebované pro zpracování celé vstupní datové sady.

Genetickým návrhem neuronových sítí se již zabývala kapitola 4. V tomto konkrétním řešení bude použit algoritmus NEAT, který byl v téže kapitole také popsán. Následující text se tedy bude zabývat myšlenkou předzpracování a jejím využitím.

5.1 Princip předzpracování

Hlavní myšlenkou předzpracování je snížení počtu vzorků klasifikovaných složitou konvenční sítí pomocí sítě s jednodušší strukturou, tzn. s nižším počtem neuronů a spojení mezi nimi. Pro správnou funkčnost tohoto procesu je zapotřebí, aby byly splněny následující dvě podmínky:

1. Síť provádějící předzpracování musí být méně výpočetně náročná, nežli síť konvenční.
2. Síť provádějící předzpracování musí provádět minimální množství chybných klasifikací.

Cílem tohoto procesu je snížení množství strojového času potřebného pro zpracování celé datové sady.

Princip činnosti sítě provádějící předzpracování je postaven na skutečnosti, že má tato síť možnost určit si, zda daný vstupní vzorek bude či nebude klasifikovat. Tím je dosaženo toho, že bude síť klasifikovat pouze část sady vstupních vzorků, a důsledkem je, že se taková síť dokáže naučit přesně klasifikovat určitou podmnožinu vzorků vstupní sady s podstatně nižší výpočetní složitostí.

Realizaci možnosti zvolit si, které vzorky budou a které nebudou zpracovány, lze zajistit přidáním dodatečného výstupu klasifikace představujícího odpověď „nevím“. Touto klasifikační třídou budou označeny všechny vzorky, u nichž síť neprovedla klasifikaci a budou muset být zpracovány složitou konvenční sítí. Zavedení této klasifikační třídy ale představuje nutnost upravit i fitness funkci, neboť je nutné správně zpracovávat i tuto novou klasifikační třídu při ohodnocování jedinců.

Je zřejmé, že klasifikační třída představovaná odpovědí „nevím“ nebude nikdy považována za správnou. Nelze ji však považovat ani za špatnou, protože při genetickém návrhu závisí vlastnosti nalezených řešení právě na ohodnocení fitness funkcí. Proto je nutné do fitness funkce přidat koeficient určující velikost chyby při odpovědi „nevím“ a vhodně stanovit jeho hodnotu.

Dalším důležitým faktem je, že síť nesmí označit všechny vzorky jako „nevím“, přestože bude tato odpověď výhodnější, nežli chybná klasifikace. Je proto nutné vyvážit, z pohledu ohodnocení jedince, 3 možné výsledky klasifikace:

- správná klasifikace (nulová chyba),
- špatná klasifikace (plná chyba),
- odmítnutí klasifikace (podíl z plné chyby).

Další možnost, jak ovlivnit množství neklasifikovaných vzorků, je zavedení pevně stanoveného omezení počtu takovýchto vzorků. Po překročení zadaného limitu je i přes zvolení možnosti neklasifikovat započítána stejná chyba, jako při klasifikaci špatné. Tímto mechanismem by mělo dojít k omezení počtu jedinců populace nevhodně využívajících možnost neklasifikovat, neboť budou fitness funkcí hůře ohodnoceni.

Pokud by nastal opačný problém, kdy by síť nepoužívala možnost neklasifikovat a její strukturální složitost by příliš rostla, bylo by možné omezit množství vytvářených neuronů skryté vrstvy maximálním počtem. Zmíněné neurony jsou totiž vytvářeny dynamicky pomocí mutace a té lze v případě potřeby přiřadit nulovou pravděpodobnost.

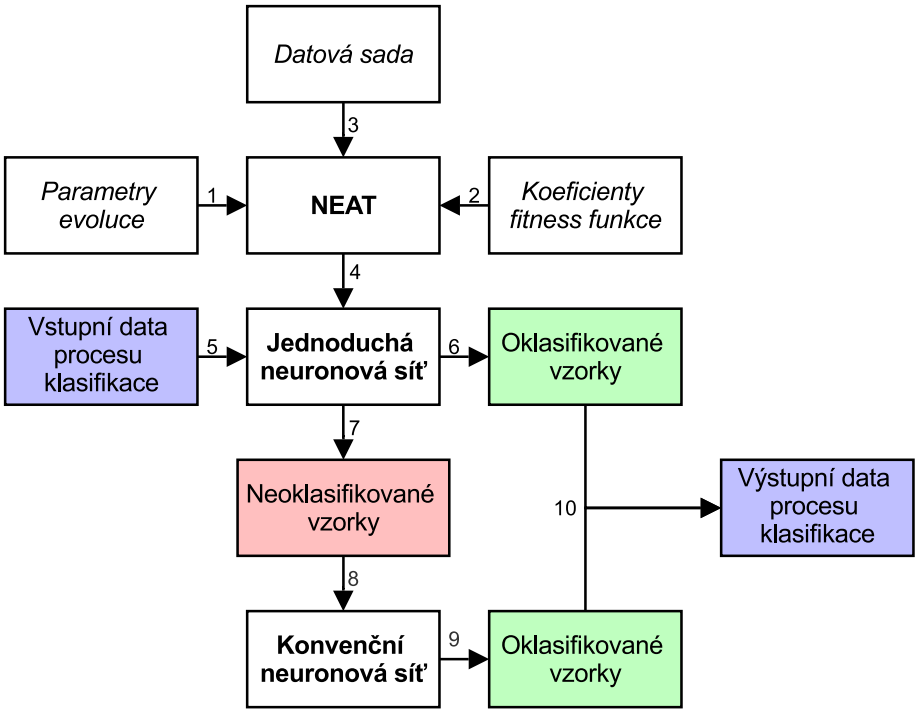
Další zajímavá možnost, jak ovlivnit evoluci sítí provádějících předzpracování, je stanovení limitu počtu správně klasifikovaných vzorků, než mohou sítě použít možnost neklasifikovat vzorky. Tím by mohla být zajištěna alespoň část již fungující struktury sítě, která by se poté zdokonalovala již s možností neklasifikovat vstupní vzorek. Také by tento limit mohl zamezit jevu, kdy síť nejprve využije všechny své možnosti neklasifikovat, a až poté začne provádět snahu vzorky klasifikovat.

5.2 Použité řešení

Práce výsledného řešení je zachycena schématem 5.1. Na tomto schématu lze vidět, že vstupem algoritmu NEAT je vhodné nastavení koeficientů fitness funkce (2), jež je pro správnou funkci evoluce takřka klíčová. Dále parametry evoluce (1) určující především limity vytvářených sítí, a v neposlední řadě vstupní datová sada (3) umožňující vytvořit síť pro odpovídající klasifikační problém.

Výstupem algoritmu NEAT je jednoduchá neuronová síť (4) provádějící předzpracování vstupních dat procesu klasifikace (5). Výstupem této sítě jsou dvě disjunktní množiny vzorků. První z nich obsahuje vzorky, u nichž byla provedena klasifikace (6), a budou tedy přímou součástí výstupních dat procesu klasifikace (10). Druhá množina vzorků obsahuje vzorky, u nichž klasifikace provedena nebyla (7), a je nutné provést jejich opětovné zpracování, tentokrát ale značně složitější konvenční neuronovou sítí (8).

Výstupem konvenční neuronové sítě pak jsou zbývající oklasifikované vzorky (9), jenž při sjednocení s množinou oklasifikovaných vzorků z jednoduché neuronové sítě (6) tvoří kompletní výstupní data procesu klasifikace (10).



Obrázek 5.1: Schéma procesu klasifikace.

Tato práce se dále bude soustředit na část provádějící předzpracování, neboť zbývající část řetězce zpracování přestavuje pouze použití již existující konvenční sítě.

5.3 Significance driven computation (SDC)

Myšlenka použití mechanismu předzpracování je postavena na metodě návrhu SDC (*Significance Driven Computation*) [25]. Metoda návrhu SDC je sice zamýšlena pro DSP (*Digital Signal Processing*) systémy, ale základní princip je poměrně obecný. Základní myšlenka tohoto přístupu je stanovení kompromisu mezi požadovaným příkonem, kvalitou a robustností výsledného řešení.

Prvním krokem návrhu (algoritmická úroveň) je stanovení významných a nevýznamných výpočtů. Všechny výpočty ovlivňující kvalitu výsledku by měly být označeny za významné a ostatní za nevýznamné. Ve významných výpočtech by mělo být dosaženo maximální možné přesnosti. U použitého mechanismu předzpracování odpovídá tomuto principu možnost neuronové sítě neklasifikovat určitý podíl vzorků. Zmíněné neklasifikované vzorky (označené odpovědí „nevím“) pak představují nevýznamné výpočty a ostatní vzorky představují výpočty významné. V metodě SDC jsou navíc brány v potaz tzv. efektivní a neefektivní takty. To se ovšem tohoto řešení netýká.

Ve druhém kroku (architektonická úroveň) provádí metoda SDC nastavení variabilní latence pro různé délky použitých cest ve vytvářeném DSP systému, takže ani tato část metody se tohoto řešení netýká.

Poslední krok představuje vyhodnocení kvality výstupu a zpětnou vazbu. Tou je provedení úprav na obou úrovních tak, aby se výsledek více přiblížil požadovanému. Část výsledného řešení odpovídající tomuto principu je vlastní genetický algoritmus, který vyhodnocuje pomocí fitness funkce jednotlivá řešení a provádí úpravy nejlepších z nich buď přímo pomocí mutací, nebo nepřímo pomocí křížení.

Nejdůležitějším převzatým principem však zůstává již popsané rozdělení vzorků na významné a nevýznamné, které umožňuje podstatně zjednodušit vytvářenou síť a dosáhnout tak požadované úspory času, a tedy i energie.

Kapitola 6

Implementace

Výsledná implementace je postavena na publikované implementaci algoritmu NEAT s názvem AccNEAT [5]. Tato implementace je alternativou (*fork*) k původnímu projektu autora algoritmu NEAT [33] a primárně se zabývá snížením množství výpočetního času potřebného k řešení složitých problémů. Čas bývá zredukován nejméně o jeden řád a pro složité problémy až o řády tři. Této akcelerace je dosaženo díky následujícím vlastnostem:

- Využívání paralelního hardware:
  - více-jádrové procesory,
  - grafické karty.
- Použití dodatečných genetických operátorů:
  - Mutace odstraňující uzel.
- Použití vyhledávacích strategií zabraňujících explosivnímu růstu sítí představujících řešení problému:
  - Technika fázového hledání.
- Optimalizace vylepšující původní implementaci NEAT beze změny algoritmu:
  - Použití datových struktur redukujících jev *CPU cache miss*.
  - Použití vyhledávacích algoritmů se složitostí $O(logN)$.

AccNEAT tedy představuje mírně vylepšenou, ale především optimalizovanou implementaci algoritmu NEAT. Hlavním vylepšením algoritmu je speciální vyhledávací strategie s názvem fázové hledání a s ním související mutace odstraňující dynamicky vytvořené neurony skryté vrstvy.

Název fázového hledání vychází z jeho základního principu. Tím je střídání dvou fází:

1. Fáze zvyšující komplexnost sítí (*complexifying*) – Používají se mutace přidávající neuron do skryté vrstvy a k reprodukci se využívá mutace i křížení. Tato fáze odpovídá původní strategii v algoritmu NEAT.
2. Fáze snižující komplexnost sítí (*pruning*) – Používají se mutace odstraňující neuron skryté vrstvy a k reprodukci se využívá pouze mutace.

Před začátkem hledání je zapotřebí vypočítat prahovou hodnotu pro přejití z první fáze do fáze druhé. Tato hodnota vychází z průměrné komplexnosti sítí v populaci, ke které je přičten definovaný koeficient.

Hledání začíná první fází. Ta je prováděna tak dlouho, dokud není překročena zmíněná prahová hodnota. Do toho okamžiku tedy dochází k postupnému zvyšování komplexnosti sítí v populaci. Jakmile je prahová hodnota překročena, přechází hledání do druhé fáze, kde je naopak komplexnost sítí postupně snižována. V této fázi hledání setrvává do doby, kdy průměrná komplexnost sítí po několik (počet je zadefinován konstantou) po sobě jdoucích generací neklesne. Poté se aktualizuje prahová hodnota a hledání se vrací do první fáze [11].

Fázové hledání má oproti původnímu řešení NEAT následující výhody [11]:

- Druhá fáze odstraňuje veškerou nebo většinu funkčně redundantní struktury v populaci. To zajišťuje posun hledání zpět do prostoru problému s méně dimenzemi, ve kterém je hledání rychlejší.
- Každá prahová hodnota pro přechod do druhé fáze je nastavena relativně vůči základní komplexnosti sítí, která byla dosažena v předchozím průchodu druhou fází. Díky tomu je komplexnost populace vždy mezi zmíněnou základní komplexností a novou prahovou hodnotou. Také je tím zajištěna oscilace komplexnosti sítí v populaci, což je požadovaný efekt k udržení malého poměru spekulativní struktury v populaci. Je tím tedy zabráněno nekontrolovatelnému zvyšování komplexnosti sítí za úroveň použitelnosti.

Kromě těchto odlišností je AccNEAT implementací původního algoritmu NEAT popsaného v sekci 4.1. Provádění zadaných klasifikačních experimentů ale vyžadovalo dodatečná rozšíření stávající implementace. V rámci práce byl tedy implementován algoritmus rozšiřující AccNEAT. Dodatečné funkce jsou detailněji popsány v následující podkapitole 6.1.

6.1 Rozšíření AccNEAT

Pro provádění požadovaných experimentů bylo zapotřebí základní implementaci AccNEAT rozšířit o další funkcionalitu, která byla autorem této práce implementována a otestována.

Nejdříve byla implementace AccNEAT rozšířena o rozhraní umožňující provádět experimenty, jenž mají za úkol vytvářet klasifikátory buněk rakoviny prsu (viz oddíl 7.1). Dále bylo přidáno rozhraní i pro experimenty vytvářející klasifikátor MNIST (viz oddíl 7.2). Obě tato rozšíření spočívala především v práci s datovými sadami zmíněných klasifikačních problémů, pro které jsou klasifikátory vytvářeny. Pro vytvoření experimentu je důležité načíst data vstupní datové sady do předdefinovaného formátu ve formě vektorů a přiřadit jim požadovaný výstup opět ve formě vektoru. Při tomto procesu byla zachována původní struktura souborů všech datových sad, aby bylo možné v případě nové verze datové sady pouze nahradit datový soubor bez úpravy kódu.

Poté byla původní implementace rozšířena o nové možnosti vyhodnocování výstupní vrstvy neuronové sítě. Původní vyhodnocování bylo postaveno na aritmeticky počítaných chybách pro experimenty nevyžadující více jak jeden neuron výstupní vrstvy. Pro účely klasifikace je však výhodnější reprezentovat každou klasifikační třídu vlastním neuronem výstupní vrstvy. Výstup neuronové sítě pro klasifikační problém pak odpovídá kódu *1 z N*, kde se v N bitovém vektoru (pro N klasifikačních tříd) nachází hodnota 1 pouze na pozici odpovídající neuronu, jenž reprezentuje správnou klasifikační třídu. Proto bylo zapotřebí

rozšířit možnosti vyhodnocování tak, aby výstupní vrstva byla vyhodnocována formou soutěžení, kdy neuron s nejvyšší hodnotou na výstupu určuje klasifikační třídu. U tohoto typu vyhodnocování je chyba sítě určena tím, zda byl výstup sítě správný, nebo nikoliv. Pokud byl výstup správný, jinými slovy zvítězil neuron výstupní vrstvy reprezentující správnou klasifikační třídu, pak je chyba sítě nulová bez ohledu na výstupy ostatních neuronů. Pokud byl výstup sítě nesprávný, pak je započítána plná chyba. A pokud je zvolen jako výstup neuron, představující možnost „nevím“, pak je započítána pouze část plné chyby. Přesněji je toto řešení popsáno v algoritmu 2, kde je navíc rozšířeno o další specifika využívaná v této implementaci.

Algoritmus 2 Pseudokód soutěživého vyhodnocení

```
1:  procedure COMPETITIVE EVALUATION(expected[N], actual[N])
2:                  ▷ Všechny proměnné ve tvaru nazev_promenne jsou externí konstanty.
3:      error ← 0
4:      expectedMax ← idx_max(expected)                          ▷ index max hodnoty
5:      actualMax ← idx_max(actual)
6:      actualSndMax ← idx_snd_max(actual)                       ▷ index druhé max hodnoty
7:      distanceMax ← distance_max(actual)                       ▷ rozdíl prvních dvou max hodnot
8:
9:      if distanceMax = 0 then                                  ▷ zvoleno více klasifikačních tříd současně
10:         error ← N                                                              ▷ plná chyba
11:     else if distanceMax < min_distance then                  ▷ rozdíl je menší než minimální
12:         error ← N · dunno_coef                                  ▷ chyba pro výstup „nevím“
13:     else if actualMax ≠ expectedMax then                          ▷ nesprávná klasifikace
14:         if actualMax ≠ dunno_idx or dunno_activated ≠ true or dunno_count ≥
    dunno_limit then                                  ▷ výstup není nebo nemůže být „nevím“
15:             if actualSndMax = expectedMax then          ▷ druhá max hodnota je správná
16:                 error ← N · snd_max_coef                              ▷ mírně snížená chyba
17:             else
18:                 error ← N                                                  ▷ plná chyba
19:             end if
20:         else
21:             error ← N · dunno_coef                          ▷ chyba pro výstup „nevím“
22:         end if
23:     end if                                          ▷ správný výstup má nulovou chybu
24: end procedure
```

Specifika, která pseudokód obsahuje navíc jsou:

- Pokud má více neuronů shodnou a zároveň nejvyšší hodnotu, pak je klasifikace označena za nesprávnou.
- Pokud je rozdíl mezi dvěma nejvyššími hodnotami nižší než parametrem (`-p`) zadaná hodnota, pak klasifikační výstup odpovídá možnosti „nevím“.
- Pokud je klasifikace nesprávná, ale druhá nejvyšší hodnota představuje správnou klasifikaci, pak je započítána mírně snížená chyba (např. o 10%).

První dvě z těchto rozšíření původní myšlenky soutěžení mají za úkol donutit evoluci hledat řešení, která od sebe více odlišují jednotlivé klasifikační třídy. Poslední pak slouží k mírnému

zvýhodnění sítí potenciálně blízkých správné klasifikaci.

Další implementovanou možností vyhodnocování výstupní vrstvy neuronové sítě je alternativní soutěživé vyhodnocení. To pracuje na stejném principu, jako výše popsané soutěživé vyhodnocování. Rozdíl je jen v tom, že je plná chyba počítána jako aritmetický rozdíl skutečných výstupů od požadovaných výstupů jednotlivých neuronů výstupní vrstvy. Pro špatnou klasifikaci pak plná chyba sítě dosahuje maximálně hodnoty N, což je odlišné oproti normálnímu soutěžení, kde je plná chyba vždy právě N. Obvykle je tedy plná chyba podstatně nižší než N, protože reflektuje reálnou vzdálenost skutečného výstupního vektoru od požadovaného.

Následně byla implementace rozšířena o výpočet a výpis statistik nejlepšího nalezeného jedince populace. Mezi statistiky patří především fitness hodnota jedince, počet správně a nesprávně klasifikovaných vzorků, počet neklasifikovaných vzorků a počet spojení a uzlů neuronové sítě daného jedince. Pro nejlepšího jedince v rámci celého experimentu je navíc ukládán průběh procesu klasifikace do `csv` souboru. Ten je poté možné použít pro statistické vyhodnocení klasifikačního procesu nalezeného řešení.

Posledním rozšířením AccNEAT je doplnění množiny vstupních parametrů pro experimenty a implementace jim odpovídající funkcionality:

- `-o` – Vypnutí OpenMP paralelismu.
- `-e` – Typ vyhodnocování výstupní vrstvy neuronové sítě.
- `-d` – Procentuální podíl klasifikovaných vzorků, které mohou být označeny odpovědí „nevím“ (mohou být neklasifikovány).
- `-a` – Procentuální podíl klasifikovaných vzorků, které musí být správně klasifikovány, než muže síť začít používat výstup „nevím“.
- `-h` – Počet neuronů skryté vrstvy v počáteční populaci.
- `-l` – Limit počtu neuronů skryté vrstvy.
- `-m` – Minimální význačnost vstupního bodu experimentu MNIST.
- `-p` – Minimální vzdálenost hodnot dvou neuronů výstupní vrstvy s nejvyšší a druhou nejvyšší hodnotou na výstupu.

Možnost vypnout OpenMP paralelismus byla přidána z důvodu provádění experimentů na clusteru, kde bylo paralelismu dosaženo vícenásobným spuštěním implementovaného řešení současně. Bez použití tohoto přepínače si OpenMP rozdělí proces na tolik vláken, kolik je na daném systému k dispozici logických procesorů.

Parametr `-e` sloužící pro zvolení způsobu vyhodnocování výstupní vrstvy neuronové sítě bylo nutné přidat z důvodu vytvoření nových metod vyhodnocování. Cílem bylo zachovat možnost provádět i původní experimenty, které jsou součástí původní implementace AccNEAT. Ty pro svou správnou funkci totiž předpokládají použití původního aritmetického vyhodnocování.

Důležitým parametrem nastavující maximální procentuální podíl neklasifikovaných vzorků je parametr `-d`. Nastavením zmíněného podílu na nenulovou hodnotu umožňuje vytvářeným sítím označit některé vzorky jako neklasifikované, a provádět tím činnost předzpracování. Bez možnosti tohoto nastavení by nemohlo být dosaženo principu předzpracování a s ním souvisejícího zjednodušení vytvářených sítí. Parametr `-d` bude ve všech experimentech nastavován na poměrně vysoké hodnoty, přestože je výchozí hodnota rovna 0. Vyšší

podíl totiž méně omezuje sítě v používání možnosti neklasifikovat. To je u sítí provádějících předzpracování výhodou za předpokladu vhodně nastavených koeficientů v algoritmu vyhodnocování. Koeficienty musí zajistit upřednostňování sítí, jenž nevyužijí lokálního optima v podobě neklasifikovat žádný vzorek, ale zároveň takových sítí, jenž dělají co nejmenší počet chybných klasifikací.

V principu opačný parametr k parametru `-d` je parametr `-a`. Jeho úkolem je nastavení procentuálního podílu vzorků, které musí jedinci vytvoření evolucí klasifikovat správně, než je jim umožněno používat možnost některé vzorky neklasifikovat. Myšlenkou tohoto nastavení je zabránění okamžitému použití výstupu „nevím“ pro všechny povolené vzorky (limitováno podílem parametru `-d`), namísto hledání správných klasifikací. Záměrem je vytvořit v daných jedincích struktury provádějící správné klasifikace, jenž přežijí změnu evolučních podmínek po zavedení možnosti neklasifikovat. Tím by mělo být dosaženo vyššího počtu klasifikovaných vzorků, při zachování nízkého množství chyb díky možnosti neklasifikovat zvolené vzorky.

Přestože je v algoritmu NEAT zadefinováno, že počáteční populace používá sítě s jedním skrytým neuronem, parametr `-h` umožňuje nastavit množství těchto neuronů na libovolnou hodnotu. Úvaha stojící za touto možností je, že pokud víme, že je zadaný problém obtížný a 1 skrytý neuron nemůže stačit, je možné počet těchto neuronů již v počáteční populaci uměle navýšit. Problémem těchto uměle přidaných neuronů je však velmi rychlý nárůst počtu spojení v síti, neboť pro tyto neurony platí, že jsou připojeny na všechny vstupní a všechny výstupní neurony sítě.

Doplňkovým parametrem pro parametr `-h` je parametr `-l` limitující maximální počet vytvořených neuronů skryté vrstvy pomocí operátoru mutace. Úkolem implementovaného řešení je hledat výrazně jednodušší sítě než konvenční, takže byla přidána i tato možnost omezit velikost vytvářených sítí. Omezení je provedeno zákazem používání operátoru mutace přidávající neuron do skryté vrstvy, pokud je dosaženo limitního počtu těchto neuronů.

Parametr `-m` provádí omezení načítání vstupních pixelů problému MNIST a stejně jako předchozí parametr slouží ke snižování strukturální složitosti vytvářených sítí. Více je tento parametr popsán v sekci 7.2.1.

Posledním doplněným parametrem je parametr `-p`. Smysl tohoto parametru již byl krátce popsán ve specifikách, které pseudokód algoritmu 2 obsahoval navíc. Hlavním cílem tohoto parametru je zvýšení rozdílů mezi jednotlivými klasifikačními třídami. Důvodem pro tento požadavek je vynucení méně zanedbatelných rozdílů mezi výstupy jednotlivých soutěžících neuronů, neboť by takováto nejistá vítězství mohli vést k chybným klasifikacím. Pokud je tento parametr aktivní, dochází k automatickému rozhodnutí o neklasifikování všech vzorků, u nichž se vyskytne zmíněné nejisté vítězství, kdy mezi neuronem s nejvyšší hodnotou výstupu a neuronem s druhou nejvyšší hodnotou výstupu má rozdíl nižší hodnotu, než hodnotu zadanou parametrem `-p`.

Kapitola 7

Experimenty

Experimenty byly provedeny nad dvěma známými klasifikačními problémy. První z nich je klasifikace buněk rakoviny prsu a druhý je klasifikace ručně psaných číslic z databáze MNIST. Celkově bylo provedeno téměř 2000 experimentů, jejichž provedení bylo z větší části zautomatizované pomocí skriptů napsaných v jazyce Python.

V rámci prováděných experimentů byly vhodně nastavovány parametry k dosažení požadovaných výsledků. Explicitně nenastavené parametry odpovídají výchozím hodnotám jsou shrnutých v tabulce 7.1.

Parametr	Výchozí hodnota	Stručný popis
`-n`	1000	Velikost populace.
`-x`	10000	Maximální počet generací.
`-s`	phased	Strategie vyhledávání.
`-e`	competitive	Typ vyhodnocování výstupní vrstvy neuronové sítě.
`-d`	0	Procentuální podíl klasifikovaných vzorků, které mohou být označeny odpovědí „nevím“.
`-a`	0	Procentuální podíl klasifikovaných vzorků, které musí být správně klasifikovány, než muže síť začít používat výstup „nevím“.
`-h`	1	Počet skrytých uzlů v počáteční populaci.
`-l`	UINT_MAX	Limit počtu skrytých uzlů.
`-m`	0	Minimální význačnost vstupního bodu experimentu MNIST.
`-p`	0	Minimální vzdálenost hodnot dvou neuronů výstupní vrstvy s nejvyšší a druhou nejvyšší hodnotou na výstupu.

Tabulka 7.1: Tabulka výchozích hodnot vstupních parametrů.

Jelikož mohou vytvářené klasifikátory používat i odpověď „nevím“, nelze používat standardní metriku přesnosti klasifikátoru, která by vyjadřovala procentuální podíl správně klasifikovaných vzorků s tím, že by platil následující vztah

$$\text{Presnost}\,[\%] = \frac{C}{N} \cdot 100 = \left(1 - \frac{W}{N}\right) \cdot 100 \tag{7.1}$$

kde C je počet správně klasifikovaných vzorků, W je počet špatně klasifikovaných vzorků a N je celkový počet vzorků. U vytvářených klasifikátorů tento vztah nemusí platit, neboť přestává platit rovnice

$$N = C + W \tag{7.2}$$

pokud platí, že D není rovno nule, kde D je počet vzorků klasifikovaných odpovědí „nevím“. Nechť je přesnost klasifikátorů umožňující tento typ odpovědi počítána dle vztahu

$$\text{Presnost}\,[\%] = \left(1 - \frac{W}{N}\right) \cdot 100 \tag{7.3}$$

Tímto způsobem výpočtu přesnosti je však ztracena informace o hodnotách C a D. Proto je zavedena druhá metrika s názvem *úplnost*, jejíž výpočet probíhá následujícím způsobem

$$\text{Uplnost}\,[\%] = \frac{C + W}{N} \cdot 100 = \frac{N - D}{N} \cdot 100 \tag{7.4}$$

Ta bude vyjadřovat procentuální podíl zpracovaných vzorků, neboli vzorků, u kterých nebyla zvolena odpověď „nevím“.

7.1 Klasifikátor buněk rakoviny prsu

Rakovina prsu, neboli *mammakarcinom*, je maligní (zhoubné) nádorové onemocnění prsu, postihující především ženy. Největší incidence rakoviny prsu je zaznamenávána ve vyspělých zemích. V roce 2011 bylo u žen České republiky diagnostikováno 6 620 případů, což v přepočtu na 100 tisíc žen představovalo 123,9 případů a jedná se o druhé nejčetnější maligní nádorové onemocnění u žen [40].

7.1.1 Problém

Cílem klasifikátoru buněk rakoviny prsu je správně klasifikovat maligní a benigní (nezhoubnou) nádorovou buňku na základě dostupných parametrů buňky. Danou problematikou se zabývá několik publikovaných datových sad. Pro provádění experimentů s tímto zaměřením byly vybrány dvě disjunktní datové sady. Pro obě datové sady bylo hledáno řešení zvlášť, neboť každá obsahuje odlišný způsob popisu klasifikovaných buněk.

První datová sada[1] použitá pro experimentování obsahuje 699 vzorků, z toho 241 maligních a 458 benigních. Tato datová sada obsahuje i vzorky s chybějícími hodnotami. Každý vzorek je popsán 11 celočíselnými atributy, z nichž první je identifikační číslo a poslední je klasifikační třída. Proto zbývá 9 atributů, dle kterých jsou klasifikované buňky roztřiďovány do zmíněných dvou klasifikačních tříd.

Druhá datová sada[2] obsahuje 569 vzorků, z toho 212 maligních a 357 benigních. Na rozdíl od první datové sady nemá žádný vzorek chybějící hodnoty, a také je podstatně více atributů popisujících vlastnosti buněk. Každý vzorek je totiž popsán 32 reálnými čísly, kde první je identifikační číslo a druhý je klasifikační třída. Zbývajících 30 atributů slouží pro klasifikaci.

Klasifikátor tohoto typu musí být velmi přesný, neboť jeho chyba může ohrozit zdraví člověka. Cílem těchto experimentů je tedy nalézt klasifikátor s minimální chybovostí pro

[1] K dispozici na adrese https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Original).

[2] K dispozici na adrese https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic).

danou datovou sadu. Z pohledu závažnosti pak také platí, že je vhodnější chyba typu I (*false positive*).

V následujících podkapitolách bude experimentálně ověřena schopnost implementace vytvořit požadovaný klasifikátor nejprve pro první a poté i pro druhou datovou sadu. Také bude zhodnocena přesnost a typy chyb nejlepších nalezených klasifikátorů.

7.1.2 Hledání řešení pro 1. datovou sadu

Před zahájením hledání výsledného řešení bylo provedeno 10 experimentů pro nalezení vhodné hodnoty parametru `-p` (minimální vzdálenost). Tyto experimenty měli všechny vstupní parametry kromě parametru `-p` fixní. Konkrétně byl nastaven parametr `-d` (podíl neklasifikovaných vzorků) na hodnotu 100 a parametr `-x` (počet generací) na hodnotu 2000. Ostatním parametrům byly ponechány výchozí hodnoty (viz tabulka 7.1). Výsledky experimentů lze vidět v tabulce 7.2. Zvýrazněný řádek tabulky vždy představuje nejlepší výsledek experimentu.

Exp	`-p`	**Výsledek** $[C/W/D]$	**Počet uzlů**	**Počet spojení**
1	0	690/9/0	18	94
2	0,00001	687/12/0	23	83
3	0,00005	688/11/0	25	102
4	0,0001	689/10/0	19	82
5	0,0005	688/11/0	17	54
6	0,001	689/10/0	19	56
7	0,005	687/8/4	29	120
8	0,01	689/10/0	15	56
9	0,05	689/10/0	24	84
10	0,1	687/10/2	21	117

Tabulka 7.2: Výsledky experimentů pro hledání vhodné hodnoty parametru `-p` (minimální vzdálenost).

Jako vhodná hodnota pro parametr `-p` byla použita hodnota 0, neboť má nejvyšší počet správně klasifikovaných vzorků a zároveň platí, že má tato síť o téměř 38% nižší počet uzlů, nežli síť z experimentu číslo 7. Ta má sice o jeden vzorek nižší počet špatně klasifikovaných vzorků, ale jedná se o nejsložitější síť ze všech vytvořených, protože má nejvyšší počet uzlů i spojení.

Po nastavení parametru `-p` na hodnotu 0 bylo provedeno 5 experimentů. Tyto experimenty měly nastaveny 5-krát vyšší počet maximálně prováděných generací (10000), čímž bylo zajištěno vyšší množství času pro hledání řešení. Parametr `-d` byl opět nastaven na hodnotu 100 a parametr `-r` (inicializace RNG) byl v každém experimentu nastaven na explicitně zadanou náhodnou hodnotu. Ostatní parametry zůstaly na svých výchozích hodnotách. Výsledky těchto experimentů nepřinesly žádné zlepšení a zároveň žádná výsledná síť nepoužívala odpověď „nevím“. Proto byl v algoritmu upraven koeficient pro tuto odpověď, aby byla pro sítě výhodnější, tzn. představovala nižší chybu. Následně bylo znovu provedeno těchto 5 experimentů s podstatně lepšími výsledky zaznamenanými v tabulce 7.3.

Přesnost výsledného řešení mírně převyšuje 99 procent a úplnost 98 procent. Neuronová síť tvořící toto řešení obsahuje 20 neuronů a 97 spojení, což není velké množství, neboť 10 neuronů tvoří vstupní vrstvu sítě (9 vstupních atributů + bias) a 3 neurony tvoří vrstvu

Exp	Výsledek $[C/W/D]$	Počet uzlů	Počet spojení	Přesnost	Úplnost
1	691/8/0	27	127	98,80%	100%
2	689/10/0	18	73	98,51%	100%
3	691/8/0	18	81	98,80%	100%
4	680/6/13	20	97	99,10%	98,14%
5	691/8/0	18	62	98,80%	100%

Tabulka 7.3: Výsledky experimentů pro 1. datovou sadu.

výstupní (2 klasifikační třídy + možnost „nevím“). Zbývajících 7 neuronů pak tvoří skrytou vrstvu. Síť je tedy poměrně jednoduchá a zároveň dokáže zpracovat téměř kompletní datovou sadu s vysokou přesností. Další důležitá informace je, že v rámci chyb, které síť při klasifikaci provedla byla pouze jedna chyba typu II (*false negative*). Ostatní chyby (5 chyb) byly chyby typu I, což jsou v lékařském oboru chyby méně nebezpečné.

V tabulce 7.4 lze vidět srovnání přesnosti nalezeného řešení (NEAT) a již existujících řešení pro první datovou sadu. Konkrétně se jedná o klasifikátory typu *Naive Bayes* (NB), *Multilayer Perceptron* (MLP), *Support Vector Machine* (SVM), *K-Nearest Neighbor* (KNN) a *decision tree* (J48) [31].

Klasifikátor	NEAT	SVM	NB	MLP	J48	KNN
Přesnost	99,10%	97,00%	95,99%	95,28%	95,14%	94,56%

Tabulka 7.4: Porovnání výsledného řešení s existujícími řešeními pro první datovou sadu [31].

7.1.3 Hledání řešení pro 2. datovou sadu

Experimenty nad druhou datovou sadou byly prováděny spolu s experimenty nad první datovou sadou, takže je jejich povaha velmi podobná. Nejprve byla experimentálně zjišťována vhodná hodnota parametru `-p` (minimální vzdálenost). Parametry pro tyto experimenty jsou stejné jako u první sady, tedy proměnný parametr `-p` a ostatní parametry fixní. Kromě parametru `-d` (podíl neklasifikovaných vzorků) nastaveného na hodnotu 100 a parametru `-x` (počet generací) nastaveného na hodnotu 2000 zůstaly všechny ostatní parametry na svých výchozích hodnotách. Výsledky lze vidět v tabulce 7.5.

Pozn.: Oproti předchozím tabulkám se v této tabulce navíc vyskytuje sloupec indikující počet aktivních uzlů. Tato hodnota se totiž u těchto sítí liší vůči hodnotě sloupce *Počet uzlů*. Indikuje počet uzlů, které nejsou izolované. Jedná se o uzly, které figurují alespoň v jednom spojení v rámci sítě.

Charakter výsledků je velmi podobný výsledkům první datové sady. Velmi dobrý výsledek poskytla výchozí hodnota parametru `-p` (0), ale také hodnota 0,05, která navíc způsobila použití výstupu „nevím“. Jelikož zde není tak velký rozdíl v počtu uzlů těchto sítí, jako u první datové sady, budou obě vhodné hodnoty experimentálně ověřeny. Pozn.: Přestože nabízí zajímavé výsledky i hodnoty v experimentech číslo 4 a 10, nejedná se o lepší výsledek, nežli výsledek pro výchozí hodnotou parametru, jenž není pro evoluci omezující, takže nejsou zmíněné hodnoty pro další experimenty použity.

Všechny následující experimenty v této sekci budou mít téměř shodné nastavení parametrů. Parametr `-d` bude nastaven na hodnotu 100, parametr `-x` bude nastaven na výchozí

Exp	p	Výsledek $[C/W/D]$	Počet uzlů	Počet spojení	Aktivní uzly
1	0	564/5/0	37	78	35
2	0,00001	563/6/0	44	108	41
3	0,00005	562/7/0	39	60	37
4	0,0001	564/5/0	39	76	38
5	0,0005	563/6/0	43	104	42
6	0,001	563/6/0	49	141	49
7	0,005	563/6/0	46	137	45
8	0,01	563/6/0	38	66	35
9	0,05	564/4/1	42	98	38
10	0,1	564/5/0	38	75	36

Tabulka 7.5: Výsledky experimentů pro hledání vhodné hodnoty parametru `-p` (minimální vzdálenost).

hodnotu 10000, parametr `-r` (inicializace RNG) bude nastaven na náhodnou hodnotu a parametr `-p` bude nastaven na danou hodnotu, pro kterou budou konkrétní experimenty prováděny. Ostatní parametry zůstanou na výchozích hodnotách.

Pro první vhodnou hodnotu parametru `-p` (0) byly nejprve provedeny experimenty s původním nastavením koeficientů. Opět, jako u první datové sady, nedošlo k žádnému zlepšení výsledků a bylo zapotřebí použít algoritmus s upraveným koeficientem pro odpověď „nevím“, který tuto odpověď zvýhodňuje. Výsledky po této úpravě jsou zobrazeny v tabulce 7.6.

Exp	Výsledek $[C/W/D]$	Počet uzlů	Počet spojení	Aktivní uzly	Přesnost	Úplnost
1	566/3/0	39	107	39	99,47%	100%
2	565/4/0	40	78	38	99,30%	100%
3	564/5/0	42	103	41	99,12%	100%
4	564/5/0	38	96	38	99,12%	100%
5	564/5/0	41	103	40	99,12%	100%

Tabulka 7.6: Výsledky experimentů pro 2. datovou sadu s hodnotou parametru `-p` = 0.

U této datové sady lze pozorovat vyšší přesnost výsledných řešení. K tomuto jevu dochází nejspíše díky vyššímu množství informací pro každý vstupní vzorek, neboť je množství atributů vzorků v této sadě více jak trojnásobné vůči sadě první. Pro parametr `-p` nastavený na výchozí hodnotu (0) je tedy maximální dosažená přesnost téměř 99,5% při zachování úplnosti na 100%. Výsledek je tedy lepší, než u první datové sady. Navíc je z pohledu počtu neuronů skryté vrstvy výsledná síť jednodušší, jelikož zde obsahuje skrytá vrstva pouze 5 neuronů. Vstupní vrstva pak obsahuje 31 neuronů a zbývající 3 neurony tvoří vrstvu výstupní.

Pro druhou vhodnou hodnotu parametru `-p` $(0,05)$ bylo následně také provedeno 5 experimentů. Výsledky těchto experimentů lze vidět v tabulce 7.7.

Z této tabulky lze vyčíst, že pro druhou z vhodných hodnot nebylo dosaženo zlepšení. Přesnost zůstala u nejlepšího výsledku stejná, ale klesl počet správně klasifikovaných vzorků a tím tedy i úplnost. Zvýšená hodnota tohoto parametru tedy splnila očekávání, neboť způ-

Exp	Výsledek $[C/W/D]$	Počet uzlů	Počet spojení	Aktivní uzly	Přesnost	Úplnost
1	564/3/2	40	111	40	99,47%	99,65%
2	558/3/8	35	73	34	99,47%	98,59%
3	564/5/0	39	100	39	99,12%	100%
4	564/4/1	37	75	33	99,30%	99,82%
5	562/3/4	36	74	36	99,47%	99,30%

Tabulka 7.7: Výsledky experimentů pro 2. datovou sadu s hodnotou parametru `-p` $= 0,05$.

sobila častější výsledek s minimální dosaženou chybovostí, ale také vedla k vyšším počtům použití odpovědi „nevím“.

Pro tuto datovou sadu je tudíž nejlepším řešením výše popsaný klasifikátor pro parametr `-p` s hodnotou 0 (první řádek v tabulce 7.6). Tento klasifikátor chyboval pouze 3 krát, avšak všechny tyto chyby byly chyby typu II (*false negative*).

V tabulce 7.8 lze vidět srovnání přesnosti nalezeného řešení (NEAT) a již existujících řešení pro druhou datovou sadu. Porovnání je provedeno vůči stejným klasifikátorům jako u první sady (viz konec podkapitoly 7.1.2).

Klasifikátor	NEAT	SVM	MLP	KNN	J48	NB
Přesnost	99,47%	97,72%	96,66%	95,96%	93,15%	92,97%

Tabulka 7.8: Porovnání výsledného řešení s existujícími řešeními pro druhou datovou sadu [31].

Obecně se na výsledek ani jednoho z vytvořených klasifikátorů (i pro první datovou sadu) nedá stoprocentně spolehnout, neboť medicínské aplikace jsou velmi kritické co se přesnosti týče. V tomto konkrétním případě je navíc přesto nutné provádět biopsii nádoru a laboratorní zkoumání získaných buněk, protože je nutné zjištění atributů buněk pro tyto klasifikátory. V případě léčby chemoterapií je navíc velmi kritická i chyba typu I, protože je tato léčba velmi agresivní a neměla by být podávána pacientovi, který ji nepotřebuje.

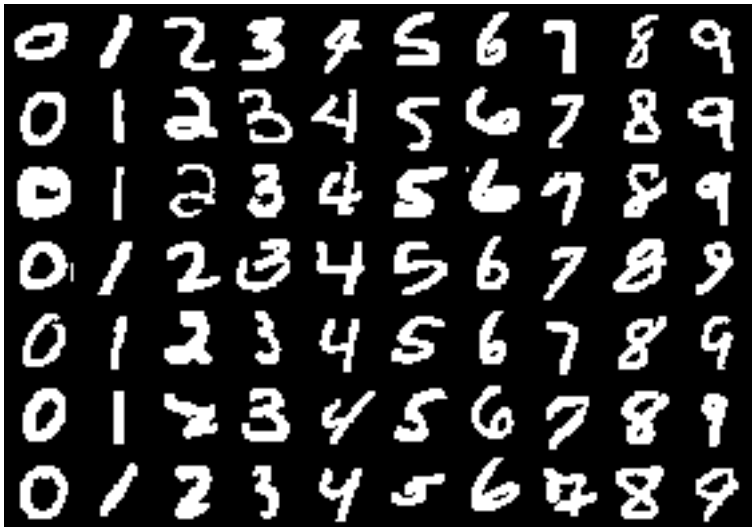
Přesnost vytvořených klasifikátorů je ale velmi vysoká, a to i v porovnání s již existujícími řešeními, které se zabývají stejnou problematikou. Schopnost algoritmu NEAT vytvářet klasifikátory pro různé datové sady tedy byla ověřena s pozitivními výsledky nalezení velmi kvalitních řešení zadaného klasifikačního problému.

7.2 Klasifikátor MNIST

MNIST[3] je databáze ručně psaných číslic obsahující celkem 70000 vzorků od 500 různých autorů. Jedná se o podmnožinu rozsáhlejší databáze NIST. Na rozdíl od NIST jsou vzorky databáze MNIST předzpracované. Původním černobílým obrázkům databáze NIST byla normalizována velikost tak, aby se vešly do čtverce o hraně 20 pixelů se zachováním poměru stran. Výsledné obrázky obsahují stupně šedi jakožto výsledek antialiasingové techniky použité v normalizačním algoritmu. Poté byly obrázky umístěny na střed čtverce o hraně 28 pixelů pomocí výpočtu těžiště pixelů a transformace obrázku do většího čtverce tak, aby se těžiště překrývalo se středem čtverce. Vzorky MNIST jsou tedy velikostně normalizované

[3] K dispozici na adrese http://yann.lecun.com/exdb/mnist/.

a umístěné na střed obrázku ve stupních šedi o pevné velikosti 28x28 pixelů. Na obrázku 7.1 lze několik vzorků MNIST vidět.



Obrázek 7.1: Ukázka několika vzorků z databáze MNIST.

7.2.1 Problém

Cílem klasifikátoru MNIST je schopnost klasifikovat výše popsané vstupní vzorky do deseti klasifikačních tříd odpovídajících celým číslům v intervalu $\langle 0; 9\rangle$. Tento problém je poměrně náročný, neboť je každý vzorek reprezentován 28^2 vstupy, kdy každému z těchto vstupů musí odpovídat jeden vstupní neuron. Počáteční populace tedy obsahuje sítě, které tvoří 797 uzlů a 796 spojení, přičemž tyto sítě obsahují pouze jeden skrytý neuron. S každým dalším skrytým neuronem přidaným pomocí parametru `-h` do sítí počáteční populace vzroste počet spojení o 796 spojení. Takto složité sítě jsou velmi výpočetně náročné. Kvůli této vysoké náročnosti byl zaveden parametr `-l` limitující maximální počet vytvořených skrytých neuronů. Typicky však jeho použití není nutné za předpokladu, že má počáteční populace pouze 1 skrytý neuron a je použita strategie fázového hledání.

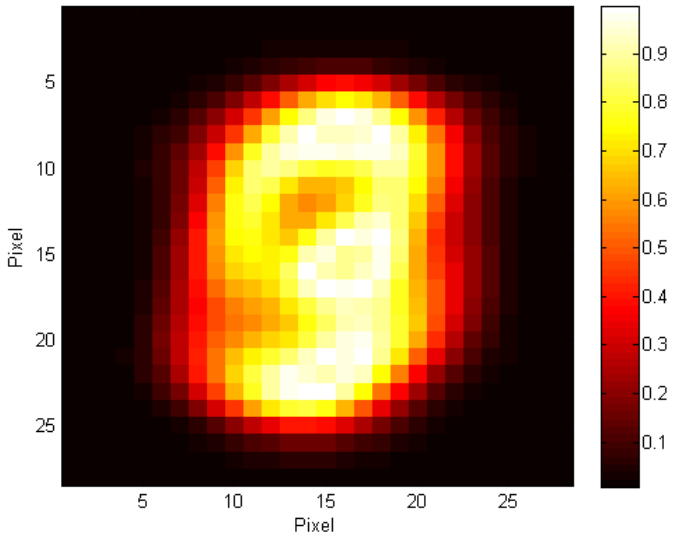
Aby byly zajištěny snížené výpočetní nároky vytvářených sítí, byl zaveden parametr `-m` omezující množství použitých pixelů ze vstupních obrázků. Inspirací k tomuto omezení byla práce pana Venkataramani et al [37], ve které je mimo jiné zkoumána i závislost proměnlivosti výstupů jednotlivých neuronů na jimi zpracovávaných pixelech. Tato závislost zobrazila velké rozdíly mezi pixely blíže středu a pixely poblíž okrajů, z čehož vyplývá i rozdíl v jejich důležitosti pro klasifikaci.

Omezení vstupních pixelů, které nebudou brány v potaz, je v této práci dáno častostí změn daných pixelů, tedy jak moc jsou pro klasifikaci zajímavé. Hodnota pro každý pixel se spočítá dle vzorce:

$$m_{[x,y]} = 1 - \frac{|128 - \bar{v}_{[x,y]}|}{128} \tag{7.5}$$

kde $\bar{v}_{[x,y]}$ je průměr hodnoty pixelu přes všechny klasifikované vzorky a hodnota 128 odpovídá středu intervalu hodnot, kterých mohou jednotlivé pixely nabývat. Hodnota $m_{[x,y]}$ se pohybuje v intervalu $\langle 0; 1\rangle$ a platí, že čím vyšší hodnota je, tím více je pixel $[x, y]$ zají-

mavý pro klasifikaci. Po výpočtu této hodnoty, pro všechny pixely vznikne teplotní mapa zobrazená na obrázku 7.2.

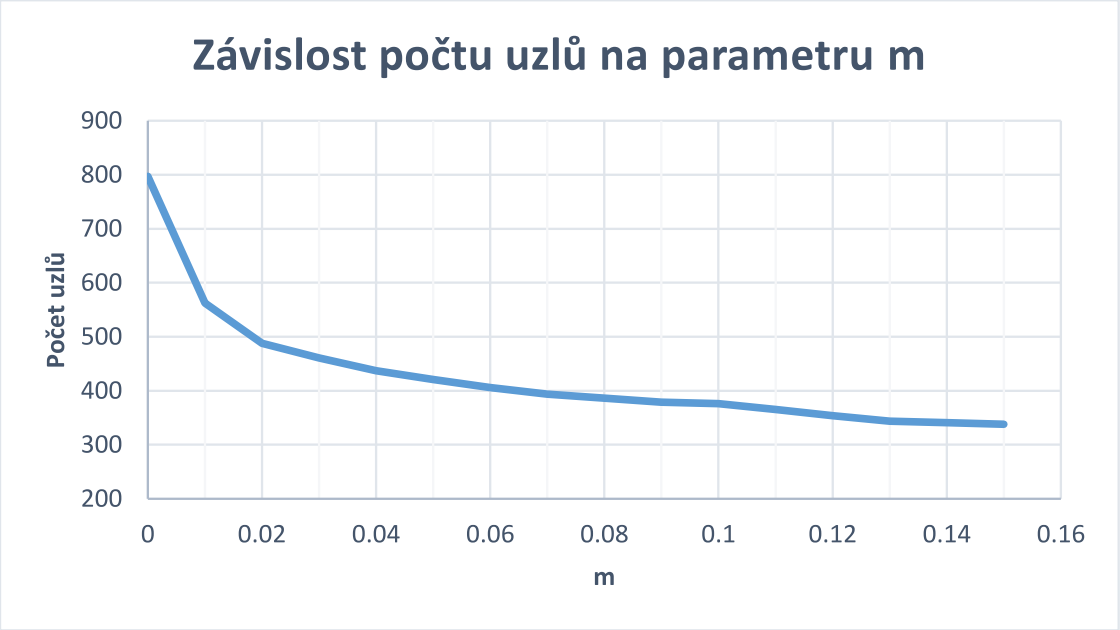


Obrázek 7.2: Teplotní mapa změn hodnot pixelů v rámci použité sady vzorků. Tmavší barva = méně zajímavý pixel pro klasifikaci.

Na teplotní mapě lze pozorovat, že okrajové pixely nejsou příliš často nositeli důležitých informací a je možné je bez ztráty velkého množství informace při zpracování zanedbat. Množství zanedbaných pixelů se nastavuje nepřímo pomocí parametru `-m`, kterým je určena minimální důležitost pixelu pro klasifikaci, neboli minimální hodnota pixelu v teplotní mapě. Hodnoty parametru jsou nastavovány z intervalu $\langle 0; 1 \rangle$ odpovídajícímu intervalu možných hodnot v rámci teplotní mapy. Graf 7.3 pak zobrazuje závislost počtu uzlů sítí počáteční populace na hodnotě parametru `-m`.

Na grafu lze pozorovat, že zvyšující se hodnota parametru `-m` vede ke snižování počtu uzlů v počátečních populacích, neboť se snižuje počet neuronů vstupní vrstvy. Závislost má od hodnoty 0,02 lineární charakter. Hodnoty 0 a 0,01 představují větší skoky nežli lineární závislost, protože velké množství vstupních pixelů okolo krajů má nízké hodnoty v teplotní mapě, na které lze tento jev pozorovat (viz 7.2).

Jelikož je problém MNIST poměrně náročný, bylo zapotřebí nejdříve stanovit vhodné parametry evoluce. Nejprve bylo nutné nalézt vhodné koeficienty používané ve fitness funkci. Tyto koeficienty určují, jak velká část plné chyby bude při nesprávné klasifikaci započítána. Jedná se o koeficienty *dunno_coef* a *snd_max_coef* zmíněné již v algoritmu 2. Po nalezení vhodných hodnot pro tyto koeficienty bylo zapotřebí určit vhodné hodnoty i pro vstupní parametry. Těmi jsou již dříve popsané parametry `-m`, `-a`, `-d` a `-p` (minimální význačnost, podíl správných klasifikací, podíl neklasifikovaných vzorků a minimální vzdálenost). Všechna tato nastavení mají velký vliv na proces evoluce, a tím je značně ovlivněna kvalita algoritmem nalezených řešení. Teprve po nalezení nejvhodnější kombinace těchto parametrů algoritmu bylo provedeno několik experimentů s cílem nalezení co nejkvalitnějšího řešení tohoto problému.



Obrázek 7.3: Graf závislosti počtu uzlů sítě na hodnotě parametru `-m` (minimální význačnost).

Experimenty tedy byly rozděleny do tří fází:

1. Hledání vhodných koeficientů fitness funkce.
2. Hledání vhodných vstupních parametrů.
3. Hledání co nejlepšího řešení.

Každé z těchto fází odpovídá jedna z následujících podkapitol, ve které jsou experimenty prováděné v dané fázi popsány a vyhodnoceny. Všechny experimenty jsou zaměřené především na maximalizaci přesnosti vytvářených klasifikátorů. Úplnost klasifikátoru vzhledem k povaze předzpracování není tolik kritická a jak již bylo řečeno, tento problém je náročný, takže řešení malou neuronovou sítí nemůže přinést příliš vysoké výsledky v obou metrikách současně.

7.2.2 Hledání koeficientů fitness funkce

Hledání vhodných koeficientů fitness funkce bylo provedeno experimentálně. Pro tyto experimenty byly nastaveny fixní vstupní parametry uvedené v tabulce 7.9.

Parametr	`-n`	`-d`	`-a`	`-m`	`-x`
Hodnota	40	60	42	0,1	20000

Tabulka 7.9: Hodnoty vstupních parametrů experimentů pro hledání koeficientů fitness funkce.

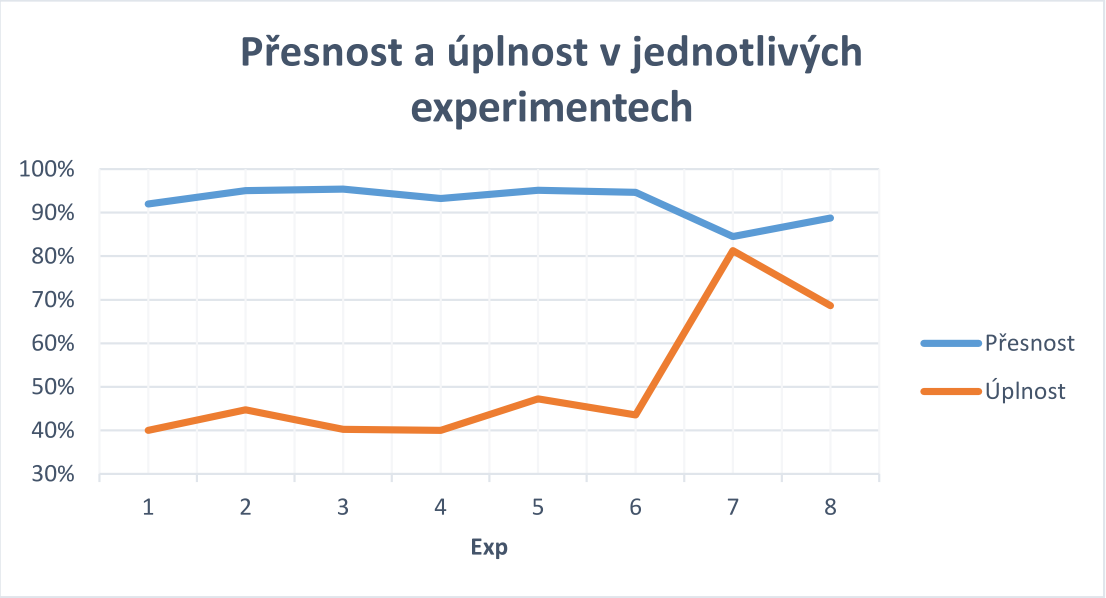
Parametrům, které nejsou v tabulce zmíněné, byla ponechána hodnota výchozího nastavení. Proměnné hodnoty ovlivňující výsledky experimentů jsou koeficienty fitness funkce:

dunno_coef a *snd_max_coef*. Pro tyto proměnné hledáme hodnoty poskytující nejlepší výsledky. Jak již bylo řečeno v úvodu do problematiky MNIST, cílem je nalézt nastavení, jenž poskytuje co nejvyšší přesnost.

Exp	***dunno_coef***	***snd_max_coef***	**Výsledek** $[C/W/D]$	**Přesnost**	**Úplnost**
1	0,20	0,90	3201/799/6000	92,01%	40,00%
2	0,30	0,80	3977/492/5531	95,08%	44,69%
3	0,30	0,90	3565/459/5976	95,41%	40,24%
4	0,35	0,85	3328/672/6000	93,28%	40,00%
5	0,40	0,80	4233/489/5278	95,11%	47,22%
6	0,40	0,90	3814/537/5649	94,63%	43,51%
7	0,50	0,75	6579/1549/1872	84,51%	81,28%
8	0,50	0,90	5738/1126/3136	88,74%	68,64%

Tabulka 7.10: Výsledky experimentů pro hledání vhodných koeficientů fitness funkce.

V tabulce 7.10 lze vidět výsledky popisovaných experimentů. Nejvyšší přesnost byla dosažena s nastavením: $dunno\_coef = 0{,}3$, $snd\_max\_coef = 0{,}9$. Síť s těmito výsledky obsahuje 206 aktivních uzlů a 282 spojení. Tato síť je zajímavá tím, že obsahuje pouze jeden skrytý neuron. Navíc díky parametru `-m` = 0,1(minimální význačnost) obsahuje 364 vstupních uzlů, ale protože je počet aktivních uzlů mnohem menší, znamená to, že i přes velké množství vynechaných vstupních pixelů se nepoužívají i další, přestože jsou jejich hodnoty přivedeny na vstup.



Obrázek 7.4: Přesnost a úplnost v jednotlivých experimentech.

V grafu 7.4 je vidět grafické znázornění metrik přesnosti a úplnosti v jednotlivých provedených experimentech. Lze pozorovat, že mezi těmito dvěma metrikami je záporná korelace. Hodnota této záporné korelace je -0,913, což znamená, že je poměrně silná. To znamená, že

metriky jsou protichůdné a zvýšení jedné z nich povede k poklesu druhé. I z tohoto důvodu je výhodné, že byla již na začátku zvolena jako prioritní metrika přesnost.

Dle výše uvedených výsledků byly zvoleny hodnoty koeficientů fitness funkce na 0,3 pro $dunno\_coef$ a 0,9 pro snd_max_coef. V následující podkapitole budou hledány nejvhodnější vstupní parametry implementace.

7.2.3 Hledání vstupních parametrů

Vstupní parametry implementace byly hledány systematickým prováděním experimentů s proměnnými vstupními parametry. V rámci této fáze bylo provedeno 1920 experimentů. Jelikož jeden experiment trval přibližně hodinu a půl při výpočtu bez paralelismu, bylo zapotřebí spustit tyto experimenty na výkonnějších výpočetních systémech. Experimenty byly provedeny na výpočetním clusteru a serverech edesignX. Pro tyto účely bylo nutné napsat skript v jazyce Python, který běh těchto experimentů zautomatizoval, a další skript na sběr dat z výstupních souborů.

Zmíněných 1920 experimentů mělo za úkol ověřit 960 unikátních kombinací vstupních parametrů. Každá kombinace tedy byla testována dvakrát a výsledek těchto dvou odpovídajících si experimentů byl zprůměrován. V tabulce 7.11 jsou zmíněny vstupní parametry, které byly v těchto experimentech nastavovány a jejich jednotlivé hodnoty.

Parametr	**Hodnoty**
`-m` (minimální význačnost)	0,08 | 0,10 | 0,12 | 0,14 | 0,16 | 0,18 | 0,20 | 0,25
`-a` (podíl správných klasifikací)	0 | 15 | 30 | 45
`-d` (podíl neklasifikovaných vzorků)	60 | 70 | 80 | 90 | 100
`-p` (minimální vzdálenost)	0,00001 | 0,00005 | 0,0001 | 0,0005 | 0,001

Tabulka 7.11: Hodnoty vstupních parametrů ověřovaných v experimentech.

Ostatní parametry byly nastaveny dle tabulky 7.12. Parametrům, které nebyly zmíněny ani v jedné z těchto tabulek, byla ponechána jejich výchozí hodnota.

Parametr	`-r`	`-n`	`-x`
Hodnota	náhodná hodnota	60	1000

Tabulka 7.12: Fixní parametry pro experimenty (inicializace RNG, počet jedinců, počet generací).

Výsledky všech experimentů zde nejsou pro jejich velké množství všechny uvedeny. Byly však zpracovány dle hodnoty poměru počtu správně klasifikovaných vzorků vůči počtu špatně klasifikovaných vzorků. Mimo to byly odfiltrovány sítě nedosahující ani 100 správných klasifikací. Výstupem pak jsou dvě vhodné kombinace vstupních parametrů, jenž jsou zapsány v tabulce 7.13.

Kombinace	`-m`	`-a`	`-d`	`-p`
1	0,20	0	90	0,0001
2	0,16	15	100	0,001

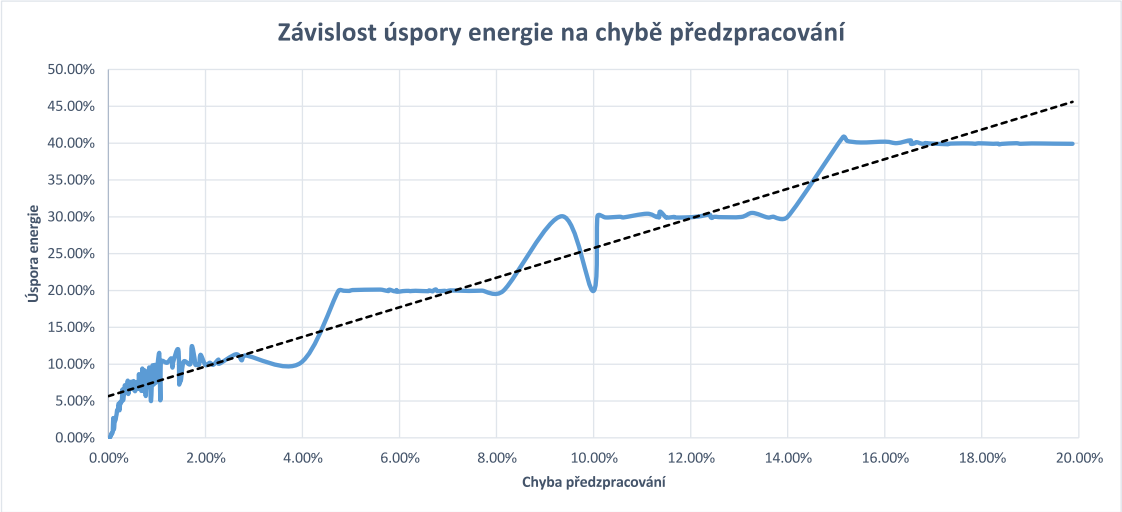
Tabulka 7.13: Výsledné nejlepší kombinace vstupních parametrů (minimální význačnost, podíl správných klasifikací, podíl neklasifikovaných vzorků, minimální vzdálenost).

První z nich je kombinací s nejnižším již zmíněným poměrem. Hodnota poměru u ní byla rovna číslu 52, což z této kombinace činí nejlepší variantu s odstupem téměř 12% od druhé nejlepší hodnoty.

Druhá kombinace v tabulce 7.13 je kombinace obsahující modus hodnot parametrů 5% nejlepších výsledků zmíněného poměru. Je to tedy kombinace tvořená nejčastějšími hodnotami parametrů v prvních 48 výsledcích seřazených dle poměru od nejvyššího k nejnižšímu.

Kromě těchto dvou výstupů ve formě vhodných kombinací vstupních parametrů pro hledání výsledného řešení byly v rámci fáze hledání vhodných vstupních parametrů zpracovány dvě analýzy výsledků experimentů. Důvod provedení analýz výstupů nad těmito výsledky je ten, že v této fázi bylo provedeno největší množství experimentů s velmi podobnými podmínkami. Nejzásadnější vliv na porovnatelnost výsledků má totiž parametr `-x` a ten byl pro tyto experimenty fixní.

První analýza je analýza závislosti úspory energie na chybě provedené předzpracováním. Cílem je tedy zjistit, jaké úspory energie lze dosáhnout při povolení určitého procenta chyb sítě provádějící předzpracování. Výsledná závislost je zobrazena na grafu 7.5.



Obrázek 7.5: Závislost úspory energie na chybě provedené ve fázi předzpracování.

Lze pozorovat, že výsledná závislost má skokové chování, ale lze ji aproximovat lineární spojnicí trendu. Trend je totiž z širšího hlediska neměnný, protože hodnoty rostou stále přibližně stejnou rychlostí. Hodnota úspory energie byla počítána dle následujícího vztahu:

$$Uspora\,[\%] = \left(1 - \frac{N \cdot preprocLinks + D \cdot procLinks}{N \cdot procLinks}\right) \cdot 100 \tag{7.6}$$

kde N je celkový počet klasifikovaných vzorků, D je počet vzorků klasifikovaných odpovědí „nevím“, $preprocLinks$ je počet spojení v síti provádějící předzpracování (sítě zde vytvářené) a $procLinks$ je počet spojení v síti představující konvenční řešení s vysokou přesností.

Jako konvenční síť byla pro tuto analýzu zvolena síť s 300 skrytými neurony, pro niž je odpovídající počet spojení roven hodnotě 238200 (hodnota proměnné $procLinks$) [3]. Hodnota proměnné $preprocLinks$ odpovídá počtu spojení dané konkrétní sítě provádějící předzpracování.

Důvodem pro výpočet úspory energie pouze pomocí informace zohledňující počet spojení v síti je ten, že jedno spojení v síti představuje při provádění sítě operaci násobení, což je

operace velmi náročná na výpočet i pro dnešní výpočetní techniku. Zejména když se jedná o násobení čísel s pohyblivou řádovou čárkou. A přestože může představovat aktivační funkce těla neuronu také potenciálně složitou operaci, těchto operací je v síti vzhledem k poměru počtu uzlů a spojení provedeno podstatně méně. Proto bylo tuto informaci možné zanedbat.

Na grafu 7.5 lze také pozorovat šum výsledné křivky u hodnot pro něž chyba předzpracování nepřevyšuje hodnotu 2%. Ten je způsoben tím, že tato oblast křivky obsahuje mírně nadpoloviční množství všech výsledných dat, ze kterých byl graf tvořen.

Také je nutné dodat, že tento graf vznikl z výsledků sítí, jenž měly pouze 1000 generací na evoluci, což je u takto rozsáhlého a složitého problému poměrně silně omezující faktor. Vyšší počet generací však přináší delší dobu potřebnou pro výpočet experimentu, a ten byl na zmíněných výpočetních systémech omezený. Snaha maximalizovat možnou úsporu bude v následující kapitole, kdy bude mít vyhledávání výsledného řešení problému MNIST podstatně větší množství času na výpočet.

Druhá analýza spočívala ve stanovení a ověření následující hypotézy pro každou kombinaci číslic: „Sítě provádějící předzpracování provádí často záměnné chyby číslice X za číslici Y.“. Nulová hypotéza H_0 tedy zní: „Daná kombinace číslic nebyla sítěmi zaměňována.“. A alternativní hypotéza H_1 je ve tvaru: „Daná kombinace číslic byla sítěmi zaměňována.“.

Platnost těchto hypotéz byla ověřována t-testem s $\alpha = 0{,}01$ u 5% nejlepších sítí vybraných dle již zmíněného poměru (C/W). Počet kombinací všech možných číslic od nuly do devíti je 45, takže zde nebudou vypsány hodnoty výsledků pro všechny, ale jen pro kombinace, u kterých byla hypotéza H_0 s danou hladinou významnosti zamítnuta. Výsledky lze vidět v tabulce 7.14.

Kombinace	0-1	1-3	1-4	1-5	1-6	1-7	1-8	1-9	2-3	4-8
N	48	48	48	48	48	48	48	48	48	48
T-test	15,00	10,77	6,49	6,35	10,33	10,30	12,65	13,26	9,32	2,61

Tabulka 7.14: Výsledky t-testu pro kombinace, kde došlo k zamítnutí nulové hypotézy.

Z výsledků vyplývá, že sítě provádějící předzpracování nejvíce zaměňují zmíněných 10 kombinací čísel. Tyto výsledky platí s 99% pravděpodobností. Mezi výsledky se však objevily i hodnoty, jenž i bez statistického zpracování znamenají, že sítě některé kombinace nikdy nezaměnily. Těmi kombinacemi jsou: 2-6, 3-4, 3-7, 4-5, 6-9 a 7-8.

7.2.4 Hledání řešení

Fáze hledání řešení představuje experimentální ověření schopnosti algoritmu NEAT nalézt vhodné řešení i pro tak složitý problém, jako je předzpracování pro databázi MNIST. Hledání řešení bylo provedeno pro obě vhodné kombinace vstupních parametrů nalezené v předchozí podkapitole. Kompletní parametry obou experimentů obsahuje tabulka 7.15.

Exp	`-m`	`-a`	`-d`	`-p`	`-n`	`-x`
1	0,20	0	90	0,0001	60	30000
2	0,16	15	100	0,001	60	30000

Tabulka 7.15: Parametry experimentů hledajících řešení problému předzpracování MNIST (minimální význačnost, podíl správných klasifikací, podíl neklasifikovaných vzorků, minimální vzdálenost).

Lze vidět, že oproti již provedeným experimentům jsou tyto experimenty podstatně delší, protože je maximální počet generací nastaven na hodnotu 30000. To zajišťuje algoritmu dostatek času pro nalezení dobrého řešení daného problému. Výsledky těchto experimentů jsou v tabulce 7.16.

Exp	**Výsledek** $[C/W/D]$	**Počet uzlů**	**Počet spojení**	**Aktivní uzly**	**Přesnost**	**Úplnost**	**Úspora energie**
1	947/14/9039	321	301	269	99,86%	9,61%	9,48%
2	1015/17/8968	337	171	139	99,83%	10,32%	10,25%

Tabulka 7.16: Výsledky experimentů hledajících řešení problému předzpracování MNIST.

Přestože byla na začátku experimentování s MNIST databází stanovena jako prioritní metrika přesnost, byl zde zvolen výsledek s nižší přesností, protože je rozdíl v přesnosti obou výsledku poměrně zanedbatelný, což se o ostatních metrikách říci nedá. Rozdíl v přesnosti výsledných sítí je totiž roven 0,03%, kdežto rozdíl v úplnosti a úspoře energie činí 0,71% a 0,77%.

Výsledné řešení tedy dokáže zpracovat 10,32% vzorků a dosahuje přesnosti 99,83%. Úspora energie pak činí 10,25%, což při chybě 0,17% převyšuje všechna dosud nalezená řešení. Síť výsledného řešení tvoří 171 spojení a 139 aktivních uzlů, přičemž skrytou vrstvu tvoří 5 uzlů. Velikost sítě je tedy vůči konvenčnímu řešení s 300 skrytými neurony téměř 8 krát nižší vzhledem k počtu uzlů, počet uzlů skryté vrstvy je 60 krát nižší a počet spojení v síti je asi 1393 krát nižší. „Daň“ za tuto značně nižší složitost je schopnost zpracovat zhruba desetinu vstupních vzorků, nikoliv celou datovou sadu.

Následuje statistická analýza nalezeného řešení, spočívající ve zkoumání výstupního souboru popisujícího klasifikační proces výše popsané sítě. V rámci této analýzy byla ověřována platnost hypotézy: „Rozdíl mezi první a druhou maximální hodnotou výstupních neuronů je signifikantně větší jak 0.“. Nulová hypotéza H_0 je pak ve tvaru: „Rozdíl prvních dvou maximálních hodnot je signifikantně roven 0.“. A alternativní hypotéza H_1 je: „Rozdíl prvních dvou maximálních hodnot je signifikantně větší jak 0.“.

Platnost těchto hypotéz byla ověřována t-testem s $\alpha = 0{,}01$. Výsledkem této analýzy je zamítnutí hypotézy H_0 s danou hladinou významnosti, a platí tedy alternativní hypotéza. Rozdíl mezi dvěma maximálními hodnotami je tudíž signifikantně větší jak 0, což znamená, že výsledná síť klasifikovala vzorky s dostatečně velkým rozdílem výstupů jednotlivých neuronů výstupní vrstvy. Na tomto výsledku se pravděpodobně projevil i fakt, že byl vstupní parametr `-p` (minimální vzdálenost) pro tuto síť nastaven na poměrně vysokou hodnotu, a sice 0,001.

Kapitola 8

Závěr

Tématem této práce byl genetický návrh neuronových sítí pro účely klasifikace. Teoretickou část práce proto z větší části představuje popis evolučních algoritmů umožňujících genetický návrh, a neuronových sítí, které jsou používány pro účely klasifikace. Výsledné řešení pak používá neuroevoluční algoritmus NEAT rozšířený především o možnost vyhodnocování výstupu klasifikátoru způsobem soutěžení a o různá nastavení tykající se možnosti klasifikace odpovědí „nevím“. Z toho vyplývá i použití principu předzpracování, čímž je vyjádřena snaha snížit výpočetní nároky vytvářených klasifikátorů, jenž by měli kooperovat s konvenčními neuronovými sítěmi. S implementovaným řešením byly provedeny experimenty k ověření správnosti neuroevolučního procesu i myšlenky předzpracování.

Experimenty byly prováděny nad dvěma klasifikačními problémy. První z nich je klasifikace buněk rakoviny prsu. Tento klasifikační problém je velmi rozšířený a bylo na něm možné ověřit schopnost algoritmu vytvářet neuronové sítě řešící zadaný problém, a také výsledné řešení porovnat s již existujícími řešeními. Druhým klasifikačním problémem je klasifikace ručně psaných číslic v databázi MNIST. Tento problém je mnohem komplexnější, a proto na něm byla ověřována především myšlenka předzpracování, jenž měla ve výsledku zajistit snížení nároků na výpočetní zdroje, potřebné pro zpracování klasifikované datové sady.

Klasifikace buněk rakoviny prsu byla prováděna nad dvěma různými datovými sadami s cílem najít co nejpřesnější řešení. Pro obě datové sady byl nalezen velmi přesný klasifikátor, který byl následně porovnán s již existujícími řešeními. Klasifikátor pro první datovou sadu dosahoval přesnosti 99,1%, což je výsledek převyšující svou přesností o více jak 2% nejpřesnější porovnávané řešení získané pomocí SVM. Pro druhou datovou sadu byl vytvořen klasifikátor s přesností téměř 99,5% a touto přesností překonal o téměř 2% nejlepší srovnávané řešení pro tuto datovou sadu opět získané pomocí SVM. Vyšší dosažená přesnost u druhé datové sady nalezeného řešení byla pravděpodobně způsobena vyšším počtem atributů známých u těchto buněk při klasifikaci. U obou datových sad byla nalezena velmi dobrá řešení daného problému a byla tím potvrzena schopnost implementace nalézt řešení pro zadaný klasifikační problém.

Problém MNIST byl z hlediska komplexnosti mnohem větší a jeho řešení je podstatně náročnější. Proto byl přidán dodatečný parametr umožňující zanedbat některé ze vstupních pixelů zpracovávaných obrázků a snížit tak velikost prohledávaného prostoru možných řešení. Tento krok se pro sítě provádějící předzpracování ukázal jako velmi vhodný, neboť umožnil hledání jednodušších a zároveň efektivnějších řešení. Z důvodu velkého počtu možných kombinací parametrů bylo provedeno systematické hledání vhodných parametrů, při kterém bylo vyhodnoceno 1920 experimentů. Výsledné řešení pak dokáže zpracovat přes

10% vstupní datové sady s velmi nízkou chybou 0,17%. Při použití s konvenční sítí obsahujících 300 skrytých neuronů pak dojde k více jak 10% úspoře energie.

V rámci zpracovaných experimentů tedy bylo dosaženo jak potvrzení schopnosti implementace hledat řešení pro zadaný problém, tak užitečnost principu předzpracování ve smyslu úspory energie při klasifikaci rozsáhlých a zároveň komplexních problémů.

Možnost pokračování v této práci by mohlo být navázání na proces hledání parametrů pro experiment MNIST statistickou cestou, pro kterou je zapotřebí provést pro každou vstupní kombinaci parametrů alespoň 30 experimentů. Alternativní možností by mohlo být zakomponování procesu hledání vhodných vstupních parametrů evoluce do algoritmu, čímž by došlo k zautomatizování celého procesu hledání řešení.

Literatura

[1] Aliev, R.; Aliev, R.: *Soft computing and its applications*. New Jersey: World Scientific, 2001, ISBN 98-102-4700-1.

[2] Altman, C.; Zapatrin, R. R.: Backpropagation Training in Adaptive Quantum Networks. *International Journal of Theoretical Physics*, ročník 49, č. 12, 2010: s. 2991–2997, ISSN 0020-7748.

[3] Cox, R.; Haskell, B.; LeCun, Y.; aj.: On the application of multimedia processing to telecommunications. *Proceedings of International Conference on Image Processing*, ročník 1997, č. 1, 1997: s. 5–8.

[4] De Jong, K. A.: *An Analysis of the Behavior of a Class of Genetic Adaptive Systems.* Dizertační práce, Ann Arbor, MI, USA, 1975, aAI7609381.

[5] Dougherty, S.: GitHub - sean-dougherty/accneat: fork of Kenneth Stanley's NEAT. 2014.
URL https://github.com/sean-dougherty/accneat

[6] Du, K.; Swamy, M.: *Neural networks in a softcomputing framework*. London: Springer, 2006, ISBN 978-184-6283-024.

[7] Eiben, A. E.; Smith, J.: *Introduction to evolutionary computing*. New York: Springer, 2003, ISBN 35-404-0184-9.

[8] Fan, H.; Song, Q.; Shrestha, S. B.: Online learning with kernel regularized least mean square algorithms. *Knowledge-Based Systems*, ročník 6, č. 59, 2014: s. 21–32, ISSN 09507051.

[9] Goldberg, D. E.: *Genetic Algorithms in Search, Optimization and Machine Learning.* Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., první vydání, 1989, ISBN 0201157675.

[10] Gomez, F. J.; Miikkulainen, R.: Solving Non-Markovian Control Tasks With Neuroevolution. In *Proceedings of the International Joint Conference on Artificial Intelligence*, San Francisco, CA: Kaufmann, 1999, s. 1356–1361.

[11] Green, C. D.: Phased Searching with NEAT. 2004.
URL http://sharpneat.sourceforge.net/phasedsearch.html

[12] Hastie, T.; Tibshirani, R.; Friedman, J.: *The Elements of Statistical Learning.* Springer Series in Statistics, New York, NY, USA: Springer New York Inc., 2001.

[13] Hastings, E.; Guha, R.; Stanley, K.: Interactive Evolution of Particle Systems for Computer Graphics and Animation. *IEEE Transactions on Evolutionary Computation*, ročník 13, č. 2, 2009: s. 418–432, ISSN 1941-0026.

[14] Hermans, M.; Dambre, J.; Bienstman, P.: Optoelectronic Systems Trained With Backpropagation Through Time. *IEEE Transactions on Neural Networks and Learning Systems*, ročník 26, č. 7, 2015: s. 1545–1550, ISSN 2162-237x.

[15] Holland, J. H.: *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. University of Michigan Press, 1975, ISBN 9780472084609.

[16] Hynek, J.: *Genetické algoritmy a genetické programování*. Praha: Grada, první vydání, 2008, ISBN 978-80-247-2695-3.

[17] Khan, G. M.; Miller, J. F.; Halliday, D. M.: Evolution of Cartesian Genetic Programs for Development of Learning Neural Architecture. *Evolutionary Computation*, ročník 19, č. 3, 2011: s. 469–523, ISSN 1063-6560.

[18] Khan, M. M.; Ahmad, A. M.; Khan, G. M.; aj.: Fast learning neural networks using Cartesian genetic programming. *Neurocomputing*, , č. 121, 2013: s. 274–289, ISSN 09252312.

[19] Kvasnička, V.: *Evolučné algoritmy*. Bratislava: Vydavateľstvo STU, první vydání, 2000, ISBN 80-227-1377-5.

[20] McCulloch, W. S.; Pitts, W.: A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, ročník 5, č. 4, 1943: s. 115–133, ISSN 0007-4985.

[21] Mehrotra, K.; Mohan, C. K.; Ranka, S.: *Elements of artificial neural networks*. Cambridge, Mass: MIT Press, 1997, ISBN 9780262133289.

[22] Michie, D.; Spiegelhalter, D.; Taylor, C.: *Machine learning, neural and statistical classification*. New York: Ellis Horwood, 1994, ISBN 01-310-6360-X.

[23] Miller, J. F.; Thomson, P.: Cartesian Genetic Programming. In *Genetic programming European conference, EuroGP 2000, Edinburgh, Scotland, UK, April 15-16, 2000*, Berlin: Springer, 2000, ISBN 9783540462392, s. 121–132.

[24] Minsky, M.; Papert, S.: *Perceptrons*. Cambridge, Mass: MIT Press, druhé vydání, 1969, ISBN 02-626-3022-2.

[25] Mohapatra, D.; Karakonstantis, G.; Roy, K.: Significance driven computation. In *Proceedings of the 14th ACM/IEEE international symposium on Low power electronics and design - ISLPED '09*, New York, New York, USA: ACM Press, 2009, ISBN 9781605586847, str. 195.

[26] Moya, M. M.; Hush, D. R.: Network constraints and multi-objective optimization for one-class classification. *Neural Networks*, ročník 9, č. 3, 1996: s. 463–474, ISSN 08936080.

[27] Šnorek, M.: *Neuronové sítě a neuropočítače*. Praha: Vydavatelství ČVUT, první vydání, 2002, ISBN 80-010-2549-7.

[28] Powers, D. M. W.: Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *International Journal of Machine Learning Technology*, ročník 2, č. 1, 2011: s. 37–63.

[29] Rosenblatt, F.: *Principles of neurodynamics*. Washington: Spartan Books, 1962.

[30] Rougier, N.: Biological neuron schema. 2001-2016.
URL https://commons.wikimedia.org/wiki/File:Neuron-figure-notext.svg

[31] Salama, G. I.; Abdelhalim, M. B.; elghany Zeid, M. A.: Experimental comparison of classifiers for breast cancer diagnosis. *2012 Seventh International Conference on Computer Engineering*, 2012: s. 180–185.

[32] Sekanina, L.: *Evoluční hardware*. Praha: Academia, první vydání, 2009, ISBN 978-80-200-1729-1.

[33] Stanley, K.: NNRG Software - NEAT C++. 2010.
URL http://nn.cs.utexas.edu/?neat-c

[34] Stanley, K. O.; D'Ambrosio, D. B.; Gauci, J.: A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks. *Artificial Life*, ročník 15, č. 2, 2009: s. 185–212, ISSN 1064-5462.

[35] Stanley, K. O.; Miikkulainen, R.: Evolving Neural Networks through Augmenting Topologies. *Evolutionary Computation*, ročník 10, č. 2, 2002: s. 99–127, ISSN 1063-6560.

[36] Tsoumakas, G.; Katakis, I.: Multi-Label Classification. *International Journal of Data Warehousing and Mining*, ročník 3, č. 3, 2007: s. 1–13, ISSN 1548-3924.

[37] Venkataramani, S.; Ranjan, A.; Roy, K.; aj.: AxNN. In *Proceedings of the 2014 international symposium on Low power electronics and design - ISLPED '14*, New York, New York, USA: ACM Press, 2014, ISBN 9781450329750, s. 27–32.

[38] Wang, Z.; Liu, M.; Cheng, Y.; aj.: Robustly Fitting and Forecasting Dynamical Data With Electromagnetically Coupled Artificial Neural Network. *IEEE Transactions on Neural Networks and Learning Systems*, ročník PP, č. 99, 2016: s. 1–14, ISSN 2162-237x.

[39] Zbořil, F. V.: Soft computing. 2015.

[40] ÚZIS-ČR: Aktuální informace č. 25/2014. 2014.
URL http://www.uzis.cz/rychle-informace/zhoubne-nadory-roce-2011

Přílohy

Seznam příloh

Příloha A

Obsah CD

- `bin\*` – spustitelný soubor aplikace s potřebnými soubory
- `src\*` – zdrojové soubory aplikace
- `thesis\*` – technická zpráva ve formátu pdf
- `thesis-latex\*` – zdrojové soubory technické zprávy ve formátu LaTeX
- `readme.txt` – návod na přeložení a spuštění aplikace