Použitie genetického algoritmu na adaptáciu neurónovej siete

Matúš Horváth

16. januára 2003

Zadanie

Skúška z prednášky „Evolučné algoritmy“ bude udelená na základe vypracovaného projektu z oblastí uvedených nižšie. Projekt je písomná práca v rozsahu približne 7–10 strán (page size A4, font size 10, line spacing single), obsahuje teoretický popis použitej metódy a získané numerické výsledky, ktoré sú ilustrované tabuľkami a grafmi. Integrálnou prílohou projektu bude aj fungujúci program, pomocou ktorého bola vypracovaná jeho aplikačná časť.

Téma

Implementujte neurónovú sieť s dopredným šírením signálu, ktorá obsahuje jednu vrstvu skrytých neurónov a so sigmoidálnou prechodovou funkciou. Použitím genetického algoritmu adaptujte túto neurónovú sieť (t. j. váhy a prahové koeficienty) siete tak, aby s čo najmenšou chybou produkovala rôzne Boolovské funkcie, pričom z numerických dôvodov namiesto hodnoty Boolovských premenných 0 a 1 budeme brať 0,1 a 0,9.

Obsah

Kapitola 1

Úvod

Problematika neurónových sietí tvorí významnú časť vednej disciplíny Umelá inteligencia. Neurónové siete ponúkajú možnosť pristupovať k modelovaniu zložitých problémov tak, ako je to človeku prirodzené – opisom vonkajšieho správania sa daného systému, na rozdiel od iných prístupov, ktoré vyžadujú algoritmické opísanie problému. Teória neurónových sietí je inšpirovaná podobnou štruktúrou nachádzajúcou sa v prírode – nervovou sústavou vyšších živočíchov a človeka [Rybár2002].

Jedným z často používaných typov neurónových sietí je viacvrstvová dopredná neurónová sieť. Takáto sieť je pri zvolení správnych koeficientov schopná na základe konečnej množiny bodov a k nim príslušných funkčných hodnôt aproximovať ľubovoľnú spojitú funkciu. Proces optimalizácie hodnôt spomínaných koeficientov (*učenie sa* alebo *trénovanie* neurónovej siete) sa najčastejšie realizuje metódou *spätného šírenia chýb* (angl. *error backpropagation*) [Návrat2002], ktorá vychádza z minimalizácie chybovej funkcie pomocou metódy *najprudšieho spádu*.

Cieľom tohoto projektu bolo preskúmať využitie odlišného prístupu k trénovaniu viacvrstvovej doprednej neurónovej siete. Namiesto metódy spätného šírenia chýb som použil na optimalizáciu hodnôt koeficientov *genetický algoritmus* [Kvasnička2000], teda ďalší prístup inšpirovaný prírodou. Princípom genetického algoritmu je napodobnenie darwinovského prirodzeného výberu na nájdenie čo najlepšieho riešenia optimalizačných problémov.

Aby som mohol zhodnotiť výhody a nevýhody použitia evolučného prístupu na trénovanie neurónovej siete, rozhodol som sa tento projekt zamerať na porovnanie obidvoch spomínaných metód – metódy spätného šírenia chýb a genetického algoritmu. Základné vedomosti o neurónových sieťach a o obidvoch metódach sú zhrnuté v nasledujúcej kapitole.

Pre potreby porovnania obidvoch prístupov som vytvoril program, ktorý umožňuje trénovať trojvrstvovú doprednú neurónovú sieť na aproximáciu rôznych funkcií pomocou obidvoch porovnávaných metód. V tretej kapitole je stručne opísaná architektúra tohoto programu a venujem sa v nej aj konkrétnej implementácií metód trénovania neurónovej siete. V prílohe tohoto dokumentu nachádza návod na použitie programu.

Pri porovnávaní metód trénovania som vykonal niekoľko simulácií, pri ktorých som sledoval rôzne parametre tréningu (napr. kvalitu aproximácie rôznych funkcií, priebeh chyby pri trénovaní alebo časovú náročnosť tréningu). Výsledky týchto simulácií sa nachádzajú vo štvrtej kapitole.

Kapitola 2

Trénovanie neurónových sietí

Obsahom tejto kapitoly je opis dvoch metód trénovania trojvrstvovej doprednej neurónovej siete. V prvej časti kapitoly sa čitateľ dozvie niečo o trojvrstvových doprednych neurónových sieťach, druhá časť kapitoly sa venuje metóde spätného šírenia chýb, v ďalšej časti sú vysvetlené všeobecné princípy genetického algoritmu a na záver kapitoly spomeniem, ako je možné využiť genetický algoritmus na adaptáciu neurónovej siete.

Cieľom tejto kapitoly nie je podrobne vysvetliť všetky spomínané metódy a algoritmy, ale poskytnúť informácie, ktoré budú slúžiť ako základ pre pochopenie zvyšnej časti dokumentu. Ďalšie, podrobnejšie informácie nájde čitateľ napríklad v publikáciách [Návrat2002], [Kvasnička1997] a [Kvasnička2000]).

2.1 Dvojvrstová dopredná neurónová sieť

Všeobecnú neurónovú sieť je možné zadefinovať ako orientovaný graf, v ktorom každý vrchol tvorí jeden neurón a každá hrana tvorí orientované spojenie medzi dvomi neurónmi [Kvasnička2000]. Každý neurón patrí do práve jednej z množín vstupných, skrytých, alebo výstupných neurónov. Vstupné neuróny sú susedné len s odchádzajúcimi a výstupné len s prichádzajúcimi hranami.

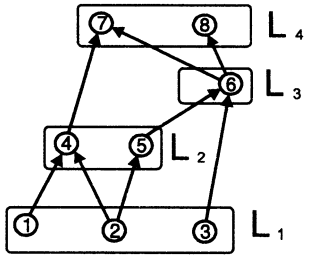
Dopredná neurónová sieť je špeciálnym prípadom neurónovej siete, ktorá neobsahuje orientovaný cyklus. Neuróny v takejto neurónovej sieti je možné rozdeliť do disjunktných vrstiev, pričom hrany môžu byť iba medzi neurónmi v dvoch rôznych vrstvách a všetky hrany medzi dvomi vrstvami neurónov sú orientované zhodne (obrázok 2.1). V trojvrstvovej doprednej neurónovej sieti sa nachádzajú tri vrstvy neurónov – vstupná, skrytá a výstupná (obrázok 2.2).

Všetky hrany aj neuróny v neurónovej sieti sú ohodnotené. Hodnotou vstupných neurónov sú zložky vstupného vektora siete. Hodnotou každého skrytého alebo výstupného neurónu je reálne číslo, ktoré sa vypočíta ako váhovaný súčet hodnôt neurónov, z ktorých do uvažovaného neurónu prichádzajú hrany:

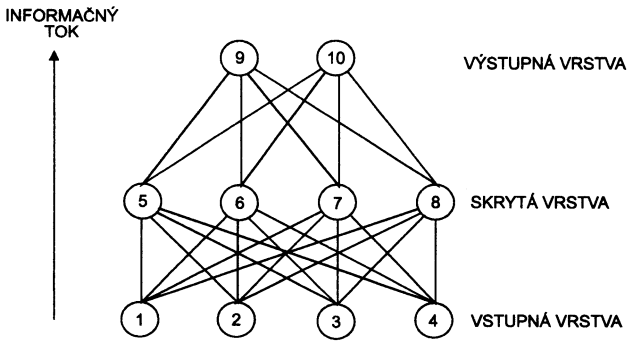
$$x_i = f(net_i) \tag{2.1a}$$

$$net_i = \sum_{j=1}^{J} w_{ij}\, x_j - \vartheta_i \tag{2.1b}$$

kde x_i je hodnota neurónu i, J je počet jeho predchodcov, w_{ij} je ohodnotenie hrany z neurónu j do neurónu i, ϑ_i je prah citlivosti neurónu i a f je aktivačná funkcia.



Obrázok 2.1: Rozdelenie neurónov v doprednej neurónovej sieti do štyroch disjunktných vrstiev L_1, L_2, L_3 a L_4 (prevzaté z [Kvasnička2000]).



Obrázok 2.2: Trojvrstvová dopredná neurónová sieť (prevzaté z [Kvasnička2000]).

Prah citlivosti neurónu je reálne číslo, ktoré je nutné napríklad pre zabezpečenie schopnosti viacvrstvovej doprednej neurónovej siete aproximovať ľubovoľnú spojitú funkciu. Bez prahu citlivosti by odozva neurónovej siete na vstupný vektor zložený zo samých núl bola daná podľa rovnice (2.1b) len aktivačnou funkciou f (pretože hodnota net_i by bola vždy 0).

V praxi sa pre zjednodušenie rovnice (2.1b) často pridáva do každej vrstvy neurónovej siete jeden neurón, ktorého hodnota je zafixovaná na -1. Hodnoty hrán smerujúcich od tohoto neurónu potom slúžia ako prahy citlivosti neurónov v ďalšej vrstve.

Aktivačná funkcia f je monotónne rastúca funkcia, ktorá zobrazuje reálne čísla na otvorený interval (A, B). Často sa používa funkcia

$$f(net) = \frac{B + Ae^{-\xi}}{1 + e^{-\xi}} \tag{2.2a}$$

Najčastejšie sa používajú hodnoty $A = 0$ a $B = 1$ alebo $A = -1$ a $B = 1$.

Dvojvrstvová dopredná neurónová sieť dostáva ako vstup vektor reálnych čísel, ktorého dimenzia je zhodná s počtom vstupných neurónov. Vstupné hodnoty sa podľa rovníc (2.1a) a (2.1b) najprv prešíria na skryté neuróny a potom na výstupné neuróny. Výstupom siete je vektor zložený z hodnôt všetkých výstupných neurónov. Podmienkou správneho fungovania siete je nastavenie vhodných ohodnotení hrán (alebo *váhových koeficientov*). Určovanie vhodných váhových koeficientov sa nazýva učenie alebo trénovanie neurónovej siete.

2.2 Metóda spätného šírenia chýb

Metóda spätného šírenia chýb [Návrat2002] slúži na optimalizáciu hodnôt váhových (a prahových) koeficientov doprednej neurónovej siete. Je založená na minimalizácii odchýlky medzi funkciou, ktorú má neurónová sieť aproximovať, a funkciou, ktorú v skutočnosti sieť realizuje. Takúto chybu je pre jeden zo vstupov siete možné vyjadriť nasledujúcou *chybovou funkciou*:

$$E_p = \frac{1}{2} \sum_{k=1}^{K} (d_{pk} - o_{pk})^2 \tag{2.3a}$$

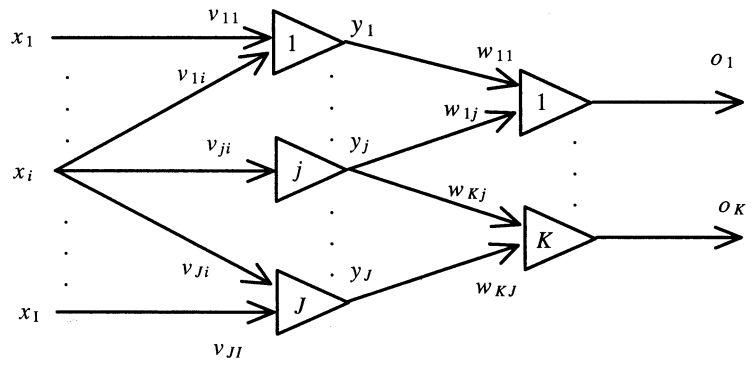
kde p je index trénovacieho vstupu, d_{pk} je požadovaný výstup siete pre tento vstup a o_{pk} je skutočný výstup siete.

Na hľadanie minima chybovej funkcie E_p sa používa metóda *najprudšieho spádu*, pri ktorej sa v každom kroku trénovania siete upravia váhové koeficienty spojení medzi neurónmi. Pre trojvrstvovú doprednú neurónovú sieť sa tieto koeficienty upravia podľa vzorcov

$$\Delta w_{kj} = -\alpha \frac{\partial E_p}{\partial w_{kj}} = \cdots = \alpha \, (d_{pk} - o_{pk}) \, f'_k \, y_j = \alpha \, \delta_{ok} \, y_j \tag{2.4a}$$

$$\Delta v_{ji} = -\alpha \frac{\partial E_p}{\partial v_{ji}} = \cdots = \alpha \left(\sum_{k=1}^{K} \delta_{ok} \, w_{kj} \right) f'_j \, x_i = \alpha \, \delta_{yj} \, x_i \tag{2.4b}$$

kde w_{kj} a v_{ji} sú váhy hrán medzi neurónmi v skrytej a výstupnej vrstve a medzi neurónmi vo vstupnej a skrytej vrstve (obrázok 2.3), α je rýchlosť učenia z intervalu $(0, 1)$, f'_k a f'_j sú derivácie aktivačných funkcií, y_j je hodnota j-teho skrytého neurónu a x_i je hodnota i-teho vstupného neurónu (a zároveň i-ta zložka vstupného vektora siete).



Obrázok 2.3: Označenie neurónov a spojení medzi nimi v trojvrstvovej doprednej neurónovej sieti (prevzaté z [Návrat2002]).

2.3 Genetický algoritmus

Jedným zo základných stochastických optimalizačných algoritmov je genetický algoritmus [Kvasnička2000]. Jeho hlavnou myšlienkou je napodobnenie darvinovskej evolúcie na vytvorenie optimálneho alebo takmer optimálneho riešenia zložitého optimalizačného problému. Genetický algoritmus je teda založený na vzájomnom porovnávaní rôznych riešení problému a preferovaní lepších riešení pred horšími.

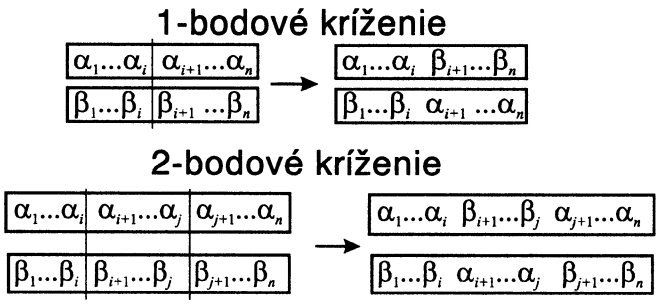
Na zdôraznenie analógie s prirodzeným výberom v prírode sa v terminológii genetického algoritmu často používajú termíny z oblasti biológie. Každé riešenie problému zakódované do reťazca binárnych čísel sa nazýva *chromozóm*. Množina chromozómov sa nazýva *populácia*. Každý chromozóm je ohodnotený funkciou *fitness*, ktorej hodnota je tým vyššia, čím je chromozóm – riešenie – bližší optimálnemu.

Na začiatku činnosti genetického algoritmu sa vygeneruje náhodná populácia chromozómov. V priebehu optimalizácie genetickým algoritmom táto populácia postupne vylepšuje pomocou troch základných operácií: *výberu rodičov*, *kríženia* a *mutácie*.

V každom kroku algoritmu sa vytvára nová populácia chromozómov. Najprv sa kvázináhodne vyberú dva chromozómy (rodičia) z pôvodnej populácie. Výber chromozómov závisí od ich fitness, čím je fitness chromozómu vyššia, tým má väčšiu šancu byť vybraný. Z dvoch rodičovských chromozómov sa potom s istou pravdepodobnosťou P_{repro} krížením vytvoria ďalšie dva, potomkovia. Kríženie je vlastne kombinácia dvoch binárnych reťazcov; často sa používa napríklad jednobodové kríženie, keď si chromozómy vymenia jeden úsek reťazca, alebo dvojbodové kríženie, keď si vymenia dva úseky (obrázok 2.4).

Po krížení nasleduje mutácia – náhodné úpravy potomkov, ktoré sa realizujú napríklad náhodným preklopením niektorých bitov chromozómov. Pravdepodobnosť mutácie P_{mut} býva väčšinou malá (rádovo 0,01). Následne sa potomkovia (v prípade, že bolo realizované kríženie), alebo pôvodní rodičia (ak sa kríženie nerealizovalo) pridajú do novej populácie. To sa opakuje, až kým sa počet chromozómov v novej populácii nezhoduje s počtom chromozómov v pôvodnej populácii.

Mnohonásobným opakovaním celého opísaného postupu sa v populácii začnú objavovať stále kvalitnejšie a kvalitnejšie chromozómy. Nastáva problém, kedy je vhodné genetický algoritmus zastaviť. Dobrou heuristikou je napríklad zastavenie algoritmu keď



Obrázok 2.4: Jednobodové a dvojbodové kríženie chromozómov. Vertikálna čiara určuje bod kríženia a chromozómy si vymenia svoje časti medzi týmito bodmi. (prevzaté z [Kvasnička2000]).

sa v populácii nachádza väčšina rovnakých chromozómov, pričom všetky ostatné chromozómy majú od nich nižšiu fitness.

2.4 Adaptácia neurónovej siete evolúciou

Ako som už spomenul, na trénovanie doprednej neurónovej siete sa najčastejšie používa metóda spätného šírenia chýb. Táto metóda dosahuje pri tréningu väčšinou dobré výsledky. Trénovanie doprednej neurónovej siete je však bežný optimalizačný problém, pri ktorom sa hľadajú také matice váh, ktoré zabezpečia čo najmenšiu hodnotu chybovej funkcie. Na vyriešenie takéhoto problému je možné využiť aj iné optimalizačné algoritmy.

Jednou z možností je použitie genetického algoritmu. V takom prípade sa každá konfigurácia váhových koeficientov považuje za jedno riešenie problému, teda za jeden chromozóm. Fitness chromozómu je možné definovať pomocou chybovej funkcie neurónovej siete (čím je chyba vyššia, tím je fitness nižšia). V priebehu optimalizácie genetickým algoritmom sa vykonáva výber, kríženie a mutácia nad populáciou neurónových sietí s rôznymi koeficientmi. Pri určovaní fitness niektorej siete je nutné nechať ju spracovať množinu vstupných vektorov so známymi správnymi výstupmi a na základe jej odpovede vypočítať chybu.

Je zjavné, že princípy takéhoto riešenia problému trénovania neurónovej siete sa veľmi blížia evolúcii nervového systému živých organizmov. V populácii živých organzmov majú tiež vyššiu šancu na rozmnožovanie tí jedinci, ktorých nervová sústava je schopná dávať „lepšie“ odpovede na podnety z okolia. Podobnosť s prírodou je o to vyššia, že podobne ako u živých organizmov sa vedomosti nadobudnuté pri učení ukladajú vo forme zmeny váh spojení medzi jednotlivými neurónmi.

Význam genetického algoritmu ako metódy trénovania doprednej neurónovej siete nie je len v samotnom natrénovaní siete. Výsledky trénovania často bývajú horšie a tréning väčšinou trvá dlhšie ako pri trénovaní pomocou algoritmu spätného šírenia chýb. Analógia medzi živou prírodou a trénovaním neurónovej siete pomocou genetického algoritmu je však dôležitým dôvodom, prečo sa genetický algoritmus na tento účel predsa využíva. Genetický algoritmus totiž umožňuje lepšie pochopiť procesy, ktoré v priebehu evolúcie života na Zemi prebiehali v prírode.

Kapitola 3

Vytvorený program

Účelom programu, ktorý som vytvoril v rámci tohoto projektu, je umožniť natrénovanie trojvrstvovej doprednej neurónovej siete pomocou dvoch metód – metódy spätného šírenia chýb a genetickým algoritmom. Pretože som chcel porovnať schopnosť siete aproximovať rôzne funkcie, jedným z cieľov bolo vytvoriť univerzálny program, ktorý bude umožňovať tréning sietí s rôznou štruktúrou (s rôznym počtom vstupných, skrytých a výstupných neurónov).

Základom programu sú tri triedy jazyka C++: trieda `BackpropNetwork`, trieda `GeneticNetwork` a trieda `DataSet`. Okrem nich sa v programe samozrejme nachádza aj niekoľko pomocných tried. Trieda `DataSet` má za úlohu načítať trénovacie dáta z externého súboru a zakódovať ich do binárnych reťazcov, ktoré sa potom dajú použiť ako vstupy neurónovej siete. Trieda `BackpropNetwork` realizuje tréning neurónovej siete pomocou spätného šírenia chýb, trieda `GeneticNetwork` realizuje to isté pomocou genetického algoritmu.

3.1 Implementácia spätného šírenia chýb

Moja implementácia metódy spätného šírenia chýb vychádza z algoritmu uverejneného v publikácii [Návrat2002]. Stručne zhrnuté, algoritmus v každom kroku trénovania „ukáže“ neurónovej sieti jeden z trénovacích vstupov a vypočíta podľa vzorcov (2.4b) a (2.4a) zmeny váhových koeficientov. Po prechode všetkými vstupmi v trénovacej množine je táto množina náhodne preusporiadaná a trénovanie pokračuje ďalším prechodom.

Na ukončenie tréningu som použil metódu *skorého zastavenia*. Táto metóda spočíva v tom, že sa všetky dostupné vstupné dáta neurónovej siete rozdelia na dve množiny A_{train} a A_{test} (rozhodol som sa rozdeliť ich v množstevnom pomere 3:1). Pri trénovaní sa potom používa len množina A_{train} a po každom prechode touto množinou sa vypočíta chyba aproximácie neurónovou sieťou pre obidve množiny A_{train} aj A_{test}. Kým chyba na množine A_{train} by mala v priebehu učenia stále klesať, chyba na množine A_{test} začne v istom okamihu stúpať (dochádza k preučeniu) a vtedy sa tréning zastaví.

Pri použití metódy spätného šírenia chýb závisí úspešnosť tréningu od rýchlosti učenia α. Pretože mojim cieľom nebolo podrobne skúmať túto metódu (cieľom bolo sústrediť sa na genetické programovanie), zvolil som túto konštantu po krátkej úvahe a niekoľko málo pokusoch na $\alpha = 0,05$. Podobne som zvolil aj rýchlosť zastavenia tréningu – tréning sa zastaví ak chyba na množine A_{test} stúpa počas viac ako 40 krokov.

3.2 Implementácia genetického algoritmu

Implementácia trénovania siete pomocou genetického algoritmu je založená na *stavoch siete*. Stavom siete rozumiem dve matice váhových koeficientov. Ako ohodnotenie stavu siete (fitness) som sa po niekoľkých neúspešných pokusoch rozhodol použiť funkciu uvedenú na strane 257 v [Kvasnička1997]:

$$\text{fitness} = \frac{1}{1-M}\left((1-\varepsilon)\,i + \varepsilon - M\right) \tag{3.1a}$$

kde M je počet chromozómov v populácii, ε je malé číslo (0,05) a i je poradie chromozómu v rade vzostupne usporiadanom podľa hodnoty funkcie

$$f() = E_{train} \tag{3.2a}$$

3.2.1 Priebeh tréningu

Tréning doprednej neurónovej siete genetickým algoritmom som implementoval nasledujúcim spôsobom: Pri tréningu sa vytvorí množina stavov siete (populácia) na začiatku inicializovaná náhodnými číslami z intervalu $(-1, 1)$. Počet prvkov v tejto populácii je kompromisom medzi výkonom a presnosťou tréningu, pretože som bol obmedzený dostupnými výpočtovými prostriedkami, zvolil som veľkosť populácie 100.

V každom kroku algoritmu sa vytvára pomocná populácia potomkov. Z populácie sa kvázináhodne vyberajú dvaja rodičia pomocou *algoritmu rulety* [Kvasnička2000], s pravdepodobnosťou P_{repro} dôjde k ich kríženiu a následne s pravdepodobnosťou P_{mut} dôjde k mutácii potomkov. Potomkovia (alebo rodičia, ak nedošlo ku kríženiu) sú potom vložení do pomocnej populácie. Nakoniec sa spojí pomocná a hlavná populácia: z hlavnej populácie sa vylúči polovica chromozómov s najnižšou fitness a doplní sa rovnakým počtom chromozómov z pomocnej populácie.

Ukončenie tréningu som pri implementácii genetického algoritmu podmienil tým, že sa v aktuálnej populácii nachádza najmenej 80% sietí s približne rovnakými koeficientmi. Je možné zadefinovať aj zložitejšie podmienky ukončenia, ale spomínaná podmienka sa v priebehu testovania programu dostatočne osvedčila.

3.2.2 Operácie kríženia a mutácie

Kríženie chromozómov som v programe realizoval formou jednobodového kríženia s bodom kríženia povoleným len na hranici reprezentácie dvoch váhových koeficientov. Ak by bod kríženia mohol byť lokalizovaný aj uprostred reprezentácie jedného z koeficientov siete (tak že polovica bitovej reprezentácie čísla by pochádzala z jedného a polovica z druhého chromozómu), vznikali by (ako som experimentálne overil) problémy s konvergenciou algoritmu. Pretože rozsah váh môže byť dosť veľký, často by sa stalo, že by výsledný váhový koeficient v bode kríženia bol veľmi odlišný od ostatných koeficientov a výsledná sieť by mala veľmi malú fitness.

Mutáciu chromozómov som vyriešil pomocou pripočítania náhodného čísla s logistickým rozdelením s parametrom $\sigma = 1$. Takéto rozdelenie dosť presne aproximuje normálne rozdelenie a výsledná náhodná premenná s pravdepodobnosťou 99% nadobúda hodnoty z intervalu $(-5, 5)$. Mutáciu pomocou náhodnej zmeny bitov som v tomto prípade nepoužil z podobných dôvodov, aké boli uvedené v predošlom odstavci.

Kapitola 4

Výsledky projektu

Pri experimentovaní s trénovaním neurónovej siete som sa najprv sústredil na nastavenie správnych koeficientov P_{repro} a P_{mut}, od ktorých závisí nielen úspešnosť natrénovania siete, ale aj to, či sa vôbec tréning zastaví. Po odladení hodnôt týchto koeficientov som niekoľkými experimentmi porovnal výkonnosť obidvoch metód trénovania siete – metódy spätného šírenia chýb aj genetického algoritmu.

4.1 Testovacie problémy

Pretože program, ktorý som vytvoril, umožňuje trénovanie siete na prakticky ľubovoľnú funkciu, okrem jednoduchých boolovských funkcií (ktoré požadovalo zadanie) som využil pri experimentovaní s trénovaním siete aj zložitejšie problémy:

Boolovská funkcia 1. Funkcia $f(A, B, C) = (A \oplus B) \oplus C$, kde $\oplus$ značí funkciu XOR. K dispozícii je všetkých 8 možných vstupných hodnôt. Neurónová sieť má 5 skrytých neurónov. Kvôli použitiu operátora $\oplus$ je tento problém relatívne ťažký na natrénovanie.

Boolovská funkcia 2. Funkcia $f(A, B, C) = (A \wedge B) \vee C$. K dispozícii je všetkých 8 možných vstupných hodnôt. Neurónová sieť má 5 skrytých neurónov. Tento problém je pravdepodobne zo všetkých najľahší.

Boolovská funkcia 3. Funkcia $f(A, B, C, D) = (A \wedge B) \vee (C \oplus D)$. K dispozícii je všetkých 16 možných vstupných hodnôt. Neurónová sieť má 5 skrytých neurónov. Ide o relatívne ľahký problém.

Násobenie. Vstup tvoria dve celé čísla od 0 po 7. Výstupom je výsledok násobenia týchto čísel. Vstupné hodnoty sú zakódované metódou „one-hot“, teda každé číslo je reprezentované ôsmimi bitmi, z ktorých práve jeden je vždy nastavený. Výstup je zakódovaný binárne. Sieť má 50 skrytých neurónov. Tento problém je relatívne zložitý kvôli zložitej závislosti medzi vstupmi a výstupmi aj kvôli zlej vypočítateľnosti optimalizovanej funkcie (výpočet trvá dlho kvôli veľkému počtu neurónov v sieti).

Parita. Vstup obsahuje čísla od 0 po 255, výstupom je parita binárnej reprezentácie čísla. Sieť obsahuje 10 skrytých neurónov. Vstup aj výstup sú kódované ako binárne čísla, ide teda o jednoduchý problém (spočítanie počtu núl na vstupe).

Tabuľka 4.1: Závislosť priebehu trénovania od hodnoty parametra P_{repro}. Parameter P_{mut} bol vo všetkých prípadoch rovný 0,01.

P_{repro}	$G(stop)$	$G(90\%)$	$G(80\%)$	E_{train}	N_{train}
0,50	5133	-	1459	10,680	86,5%
0,75	50355	7872	1374	2,483	96,9%
0,80	34197	2700	1447	4,011	94,8%
0,85	17785	3086	2307	2,495	96,4%
0,90	21846	3572	1575	2,110	95,8%
0,95	43093	4582	2360	4,103	95,3%

Tabuľka 4.2: Závislosť priebehu trénovania od hodnoty parametra P_{mut}. Parameter P_{repro} bol vo všetkých prípadoch rovný 0,85.

P_{mut}	$G(stop)$	$G(90\%)$	$G(80\%)$	E_{train}	N_{train}
0,005	11003	6563	2065	9,417	90,1%
0,010	17785	3086	2307	2,495	96,4%
0,050	11754	-	3073	9,107	87,5%
0,100	20214	-	3517	8,455	89,1%

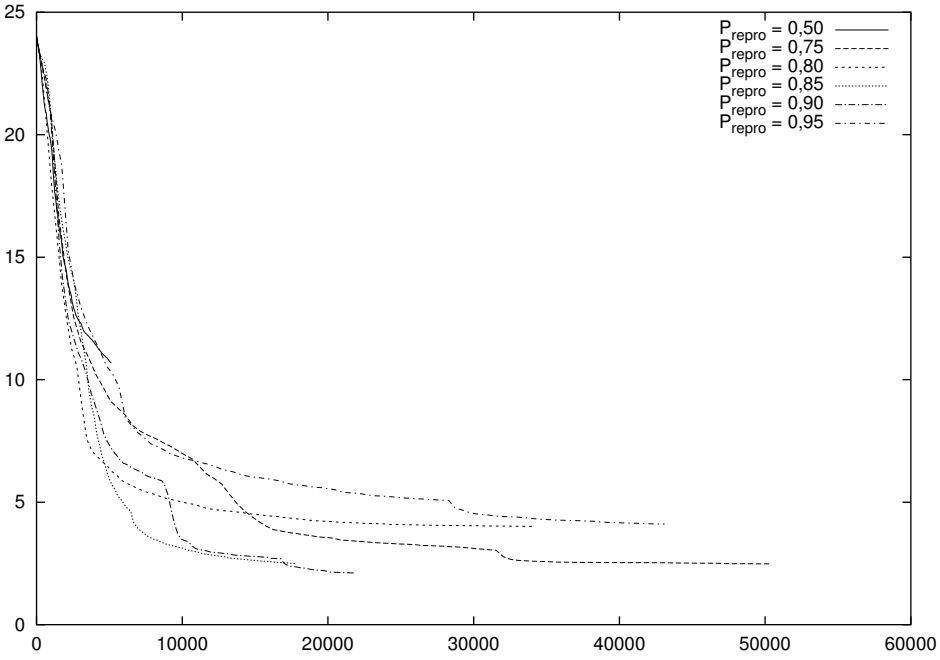
Splice. Najzložitejší z problémov. Ide o reálne dáta z oblasti génového inžinierstva. Súbor obsahuje 3175 vstupov, každý z nich obsahuje 60 čísel od 0 po 3. Výstupom je vždy jedno číslo od 0 po 3. Vstupy sú zakódované ako binárne čísla, výstup je zakódovaný „one-hot“. Neurónová sieť má 20 skrytých neurónov. Tento problém sa dá veľmi dobre natrénovať pomocou spätného šírenia chýb, ale tréning trvá kvôli vysokému počtu vstupov dlho.

4.2 Výber vhodných koeficientov

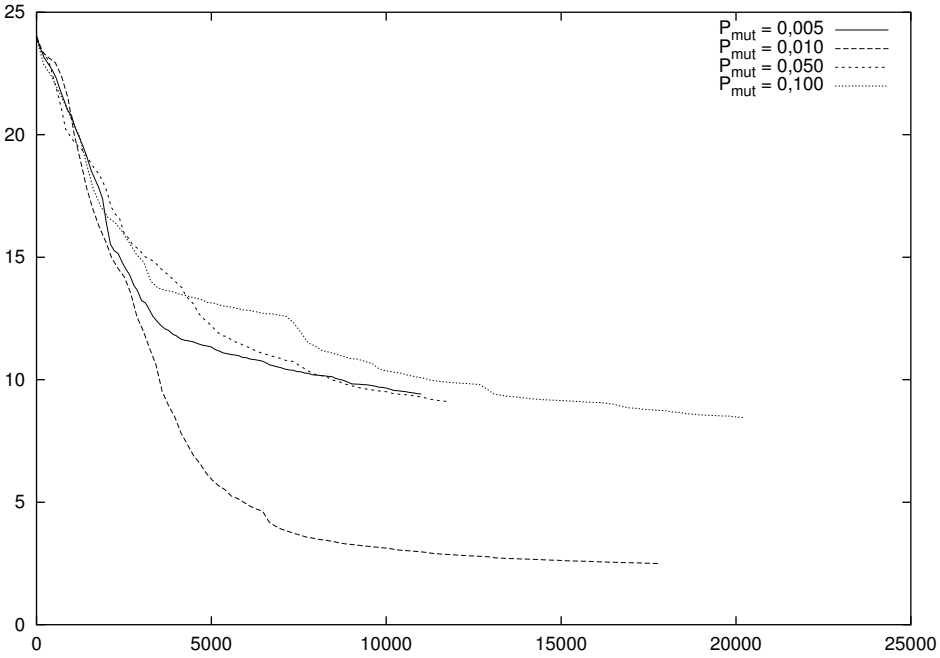
Pri výbere vhodných koeficientov pre učenie sa pomocou genetického algoritmu som niekoľko krát natrénoval neurónovú sieť na problém *Parita*. V tabuľke 4.1 sú zhrnuté údaje o tréningu získané pre rôzne hodnoty parametra P_{repro}, v tabuľke 4.2 zasa pre rôzne hodnoty P_{mut}.

Prvý stĺpec každej z tabuliek obsahuje hodnoty P_{repro} alebo P_{mut}. Druhý, tretí a štvrtý stĺpec obsahuje počet generácií potrebných na zastavenie programu, na dosiahnutie úspešnosti $N_{train} = 90\%$ a na dosiahnutie úspešnosti $N_{train} = 80\%$. Piaty stĺpec obsahuje výslednú chybu E_{train} a posledný stĺpec výslednú úspešnosť natrénovania N_{train}. Priebeh chyby E_{train} pri trénovaní siete genetickým algoritmom pre rôzne parametre P_{repro} a P_{mut} je znázornený aj na obrázkoch 4.1 a 4.2.

Z týchto obrázkov je vidieť, že genetický algoritmus veľmi rýchlo nájde takmer optimálne riešenie, ale potom sa riešenie len veľmi pomaly zlepšuje, takže dlho trvá, kým sa splní ukončovacia podmienka algoritmu. V prípade problému *Parita* trval tréning výnimočne dlho – chyba sa ku konci tréningu znižovala už len na treťom desatinnom mieste, pričom k zníženiu došlo približne raz za desať generácií.



Obrázok 4.1: Priebeh chyby E_{train} pri trénovaní genetickým algoritmom pre rôzne hodnoty P_{repro}. Hodnota P_{mut} je vo všetkých prípadoch 0,01. Na horizontálnej osi je počet generácií.



Obrázok 4.2: Priebeh chyby E_{train} pri trénovaní genetickým algoritmom pre rôzne hodnoty P_{mut}. Hodnota P_{repro} je vo všetkých prípadoch 0,85. Na horizontálnej osi je počet generácií.

Po vykonaní experimentov sa zdá byť najvýhodnejšia kombinácia parametrov genetického algoritmu $P_{repro} = 0,80$ a $P_{mut} = 0,01$. Hoci boli tieto hodnoty získané len trénovaním jedného problému, ukázalo sa, že zabezpečia takmer optimálne výsledky tréningu pre všetky problémy spomenuté v predošlej podkapitole.

4.3 Porovnanie metód

Na každý z problémov spomenutých v podkapitole 4.1 som natrénoval neurónovú sieť obidvomi metódami. Údaje, ktoré som o procese trénovania získal, sú zhrnuté v tabuľkách 4.3 a 4.4. Z nich je vidieť, že každá z porovnávaných metód má svoje výhody aj nevýhody, ktoré sa pri každom probléme prejavili inak. V dodatku B na konci dokumentu sa nachádzajú pre každý problém aj grafy priebehu chyby na testovacej a trénovacej množine pri obidvoch tréningoch.

4.3.1 Boolovská funkcia 1

Tento problém bol pre metódu spätného šírenia chýb príliš ťažký. Dokázala natrénovať sieť len na 50% na obidvoch množinách. Genetický algoritmus si s problémom poradil lepšie a porovnateľne rýchlo.

4.3.2 Boolovská funkcia 2

Druhá boolovská funkcia bola pre zmenu príliš ťažká pre genetický algoritmus. Napriek tomu, že bol schopný veľmi dobre natrénovať sieť, nedokázal skončiť. Pravdepodobne to bolo zapríčinené tým, že pre tento problém existujú dve rôzne optimálne konfigurácie váh, ktoré majú rovnakú fitness. Algoritmus potom nevytvoril populáciu skladajúcu sa z väčšiny rovnakých chromozómov, čo je podmienkou jeho ukončenia.

4.3.3 Boolovská funkcia 3

S touto funkciou si uspokojivo neporadila ani jedna metóda. Metóda spätného šírenia chýb nedokázala natrénovať sieť s dostatočnou úspešnosťou, genetický algoritmus zasa sieť pretrénoval tak, že na testovacej množine dosiahla úspešnosť 0%.

4.3.4 Násobenie

Tento problém je typickým príkladom, keď sa oplatí využiť genetický algoritmus. Hoci pomocou spätného šírenia chýb sa sieť natrénovala veľmi nekvalitne, genetický algoritmus dosiahol 100% a 31% úspešnosť na trénovacej a testovacej množine. Cenou za to bol veľmi dlhý čas tréningu – takmer hodina.

4.3.5 Parita

Problém *Parita* je veľmi zvláštny. Hoci ide o jednoduché spočítanie núl na vstupe, neurónová sieť sa pomocou metódy spätného šírenia chýb natrénovala len na približne 50%. Keď si uvedomíme, že výstupom siete je jeden bit, sieť vlastne dosiahla rovnakú úspešnosť ako generátor náhodných čísel.

Genetický algoritmus konvergoval veľmi pomaly, ale postupne dokázal nájsť optimálnu konfiguráciu váh. Problémom bolo aj zastavenie algoritmu, niekoľko krát sa stalo,

Tabuľka 4.3: Porovnanie obidvoch algoritmov – tabuľka pre spätné šírenie chýb. V tabuľke je celkový čas tréningu v sekundách a chyby a úspešnosti natrénovania na trénovacej a testovacej množine.

Problém	čas	E_{train}	E_{test}	N_{train} (%)	N_{test} (%)
Bool1	0 s	0,759	0,248	50,0%	50,0%
Bool2	0 s	0,686	0,241	83,3%	50,0%
Bool3	0 s	1,394	0,499	58,3%	50,0%
Násob	0 s	19,233	9,904	37,5%	25,0%
Parita	36 s	24,268	7,980	53,1%	47,9%
Splice	28 s	18,385	47,172	99,3%	89,8%

Tabuľka 4.4: Porovnanie obidvoch algoritmov – tabuľka pre genetický algoritmus. V tabuľke je celkový čas tréningu v sekundách a chyby a úspešnosti natrénovania na trénovacej a testovacej množine.

Problém	čas	E_{train}	E_{test}	N_{train} (%)	N_{test} (%)
Bool1	3 s	0,000	0,500	100,0%	50,0%
Bool2	-	0,000	-	100,0%	-
Bool3	0 s	0,000	2,000	100,0%	0,0%
Násob	57 m 37 s	1,541	9,581	100,0%	31,3%
Parita	110 m 12 s	4,011	13,264	94,8%	89,3%
Splice	253 m 42 s	17,314	92,351	99,4%, 83,1%	

že program pokračoval v činnosti aj keď už bola chyba veľmi malá. Stále sa však dala pomaly zmenšovať, takže algoritmus neskončil.

4.3.6 Splice

Tento problém sa vyznačuje tým, že sa na neho dá relatívne ľahko natrénovať neurónová sieť pomocou algoritmu spätného šírenia chýb. Naopak, veľmi ťažko sa na neho trénuje sieť genetickým algoritmom, pretože vstupná množina dát je veľmi rozsiahla. Nakoniec však obidva prístupy vedú k dobrému výsledku.

Kapitola 5

Zhrnutie

Cieľom tohoto projektu bolo analyzovať použitie genetického algoritmu na trénovanie trojvrstvových dopredných neurónových sietí. Pri porovnaní takéhoto evolučného prístupu s klasickým trénovaním metódou spätného šírenia chýb som zistil, že každý z algoritmov má svoje výhody aj nevýhody.

Medzi výhody metódy spätného šírenia chýb patrí rýchlosť konvergencie a spoľahlivé natrénovanie siete bez rizika pretrénovania. Táto metóda však pri pokusoch nedokázala zvládnuť niektoré problémy, s ktorým si genetický algoritmus poradil, i keď s ťažkosťami a často po dlhšom čase.

Trénovanie genetickým algoritmom sa ukázalo ako pomalšie, ale o to lepšia bola jeho konvergencia. Genetický algoritmus dokázal natrénovať sieť na každý z problémov s dostatočne malou chybou E_{train}. Problémom však je, že genetický algoritmus hľadá globálne minimum optimalizovanej funkcie a nebere do úvahy možnosť pretrénovania. Preto aj sieť natrénovaná s veľmi malou chybou E_{train} môže dávať oveľa horšie výsledky na testovacej množine ako na trénovacej.

Kombináciou neurónovej siete a genetického algoritmu vznikla situácia v niečom podobná spôsobu, akým v priebehu evolúcie vznikol z najjednoduchších organizmov na Zemi taký zložitý orgán, akým je ľudský mozog. Pri trénovaní neurónovej siete pomocou genetického algoritmu sa z množiny neurónových sietí s náhodnými váhami, ktoré prakticky nie sú schopné riešiť zadaný problém, postupne vytvorí množina neurónových sietí s koeficientmi blízkymi optimálnym.

Takáto podobnosť genetického algoritmu so živou prírodou umožňuje využívať počítačovú simuláciu na lepšie porozumenie vývoju druhov na Zemi. Aj to je jeden z prínosov použitia genetického algoritmu na trénovanie neurónovej siete.

Dodatok A

Použitie programu

Program funguje pod operačným systémom Microsoft Windows 2000 alebo vyšším. Je možné ho skompilovať aj pod Unixom. Na to je potrebný prekladač GCC a programy Flex a Bison. Pod operačným systémom Windows je možné program preložiť prekladačom Microsoft Visual C++ 6.0.

Program sa ovláda z príkazového riadku. Vstupné dáta sa nachádzajú v textovom v súbore s príponou .dat. Okrem tohoto súboru musí existovať v tom istom adresári aj súbor s tým istým menom a príponou .hdr. Formát týchto súborov sa dá pravdepodobne najlepšie pochopiť prezretím už existujúcich príkladov dodaných s programom.

Na spustenie trénovania metódou spätného šírenia chýb je potrebné zadať

```
$ ./Projekt -trainbp <cesta_k_suboru_hdr>
```

Na spustenie trénovania pomocou genetického algoritmu je potrebné zadať

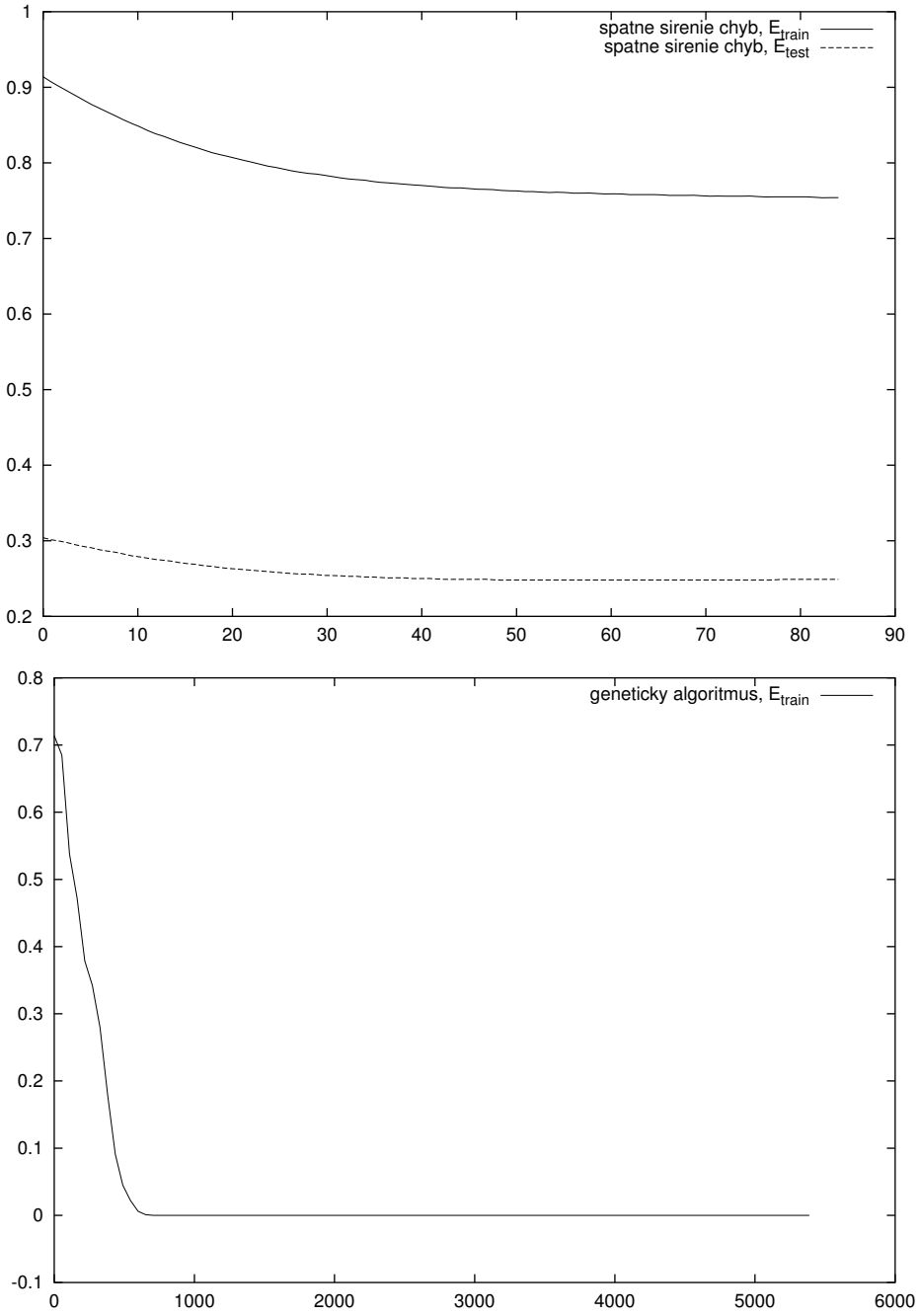
```
$ ./Projekt -traingen <cesta_k_suboru_hdr>
```

Program v priebehu tréningu vypisuje na obrazovku aktuálny stav. Tie isté informácie sa zapisujú aj do súboru `debug.txt`.

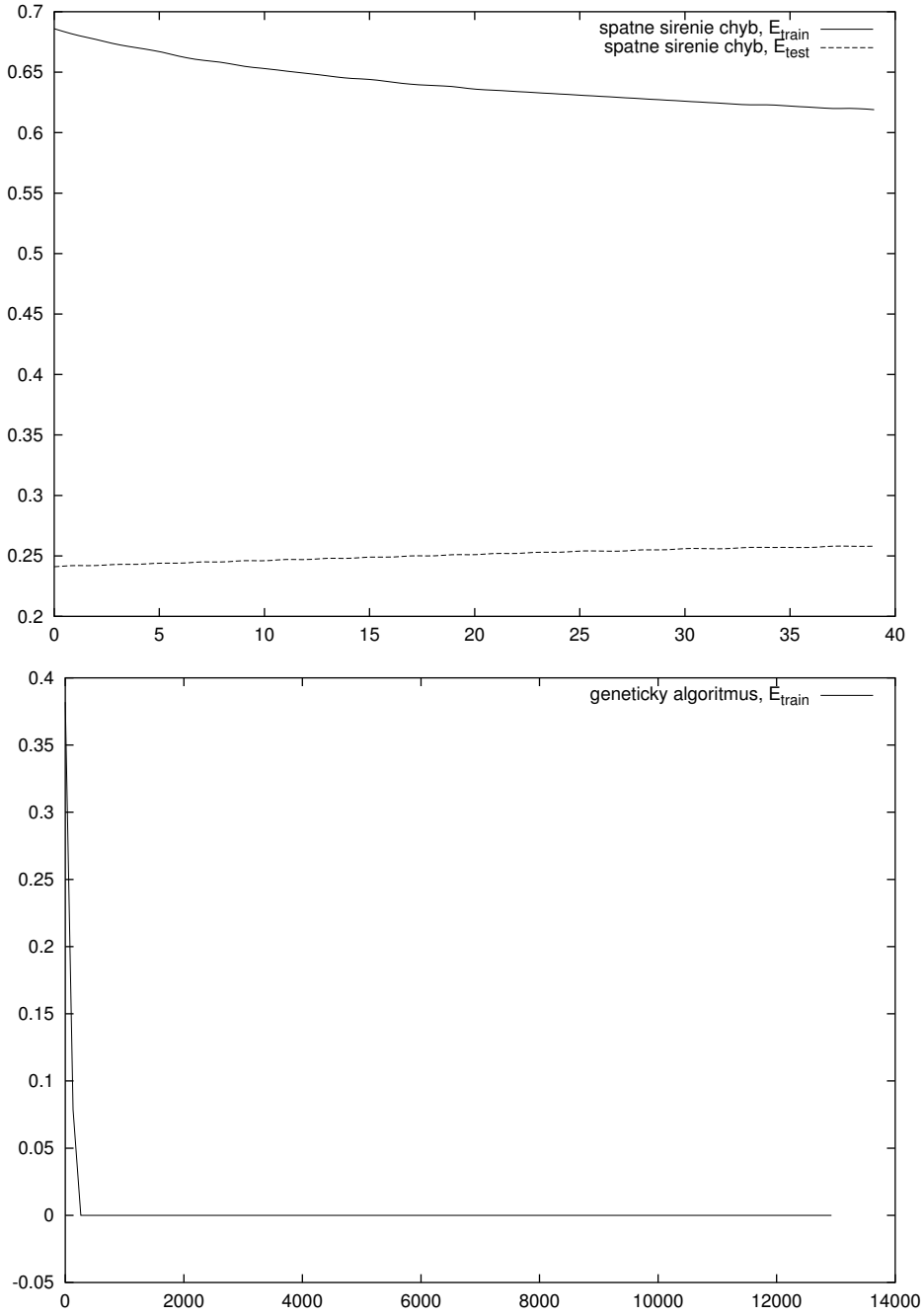
Dodatok B

Grafy priebehu chyby pri tréningu

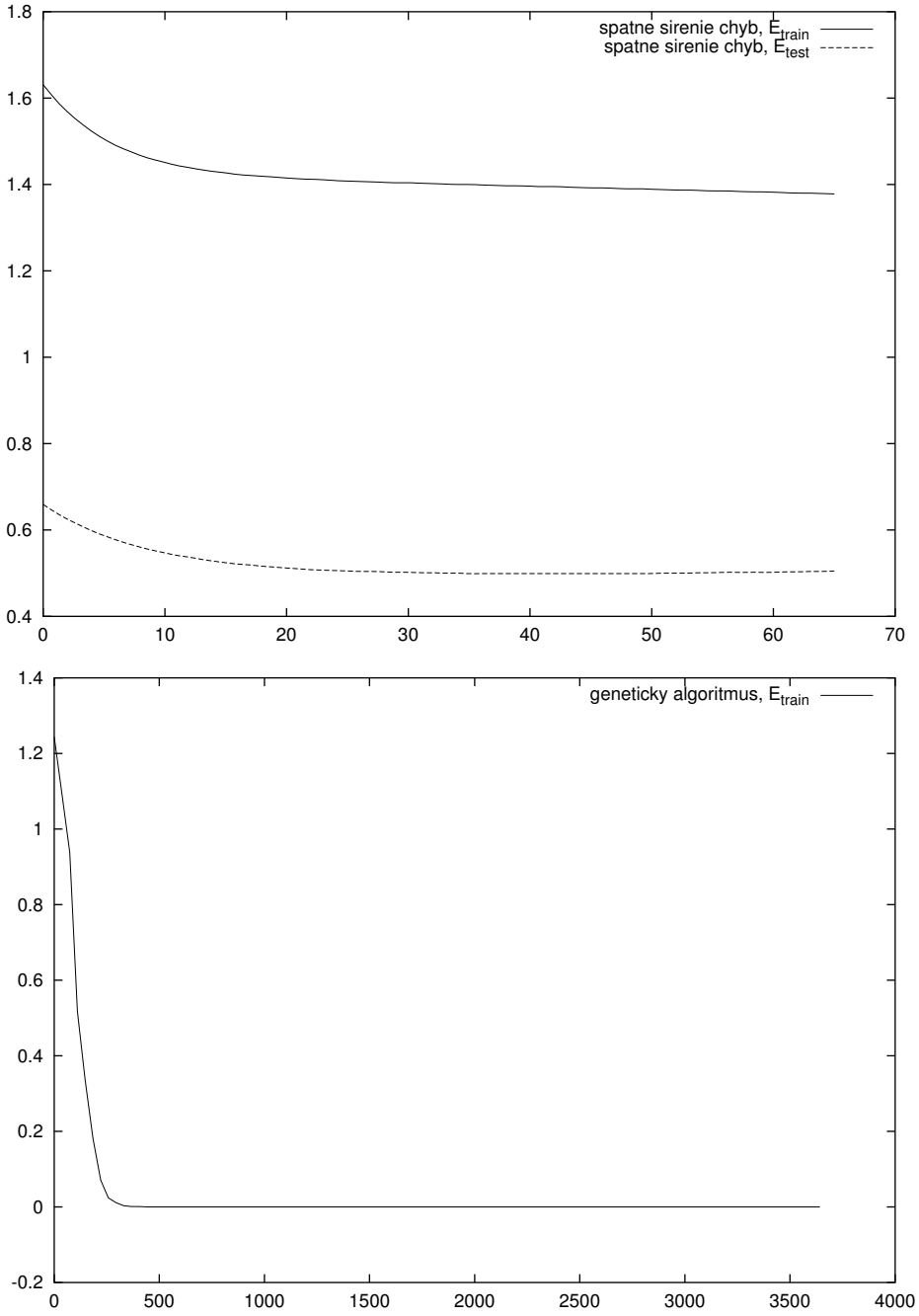
V tomto dodatku sa nachádza pre každý problém opísaný v kapitole 4.1 graf priebehu chyby počas tréningu. Pri každom probléme je znázorenený priebeh chýb E_{train} a E_{test} pre metódu spätného šírenia chýba a priebeh chyby E_{train} pre trénovanie genetickým algoritmom.



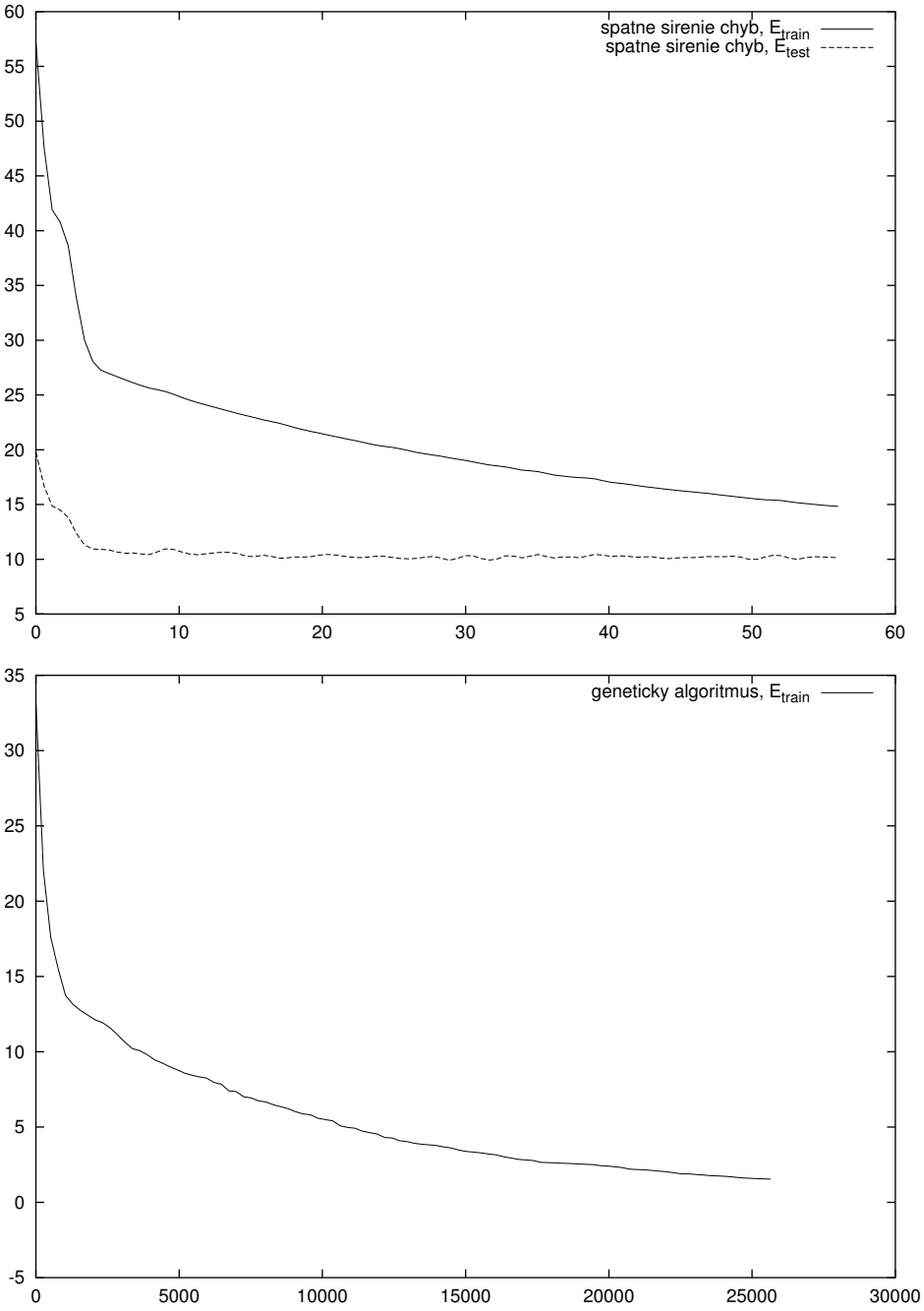
Obrázok B.1: Priebeh chýb pri tréningu neurónovej siete na problém *Boolovská funkcia 1.*



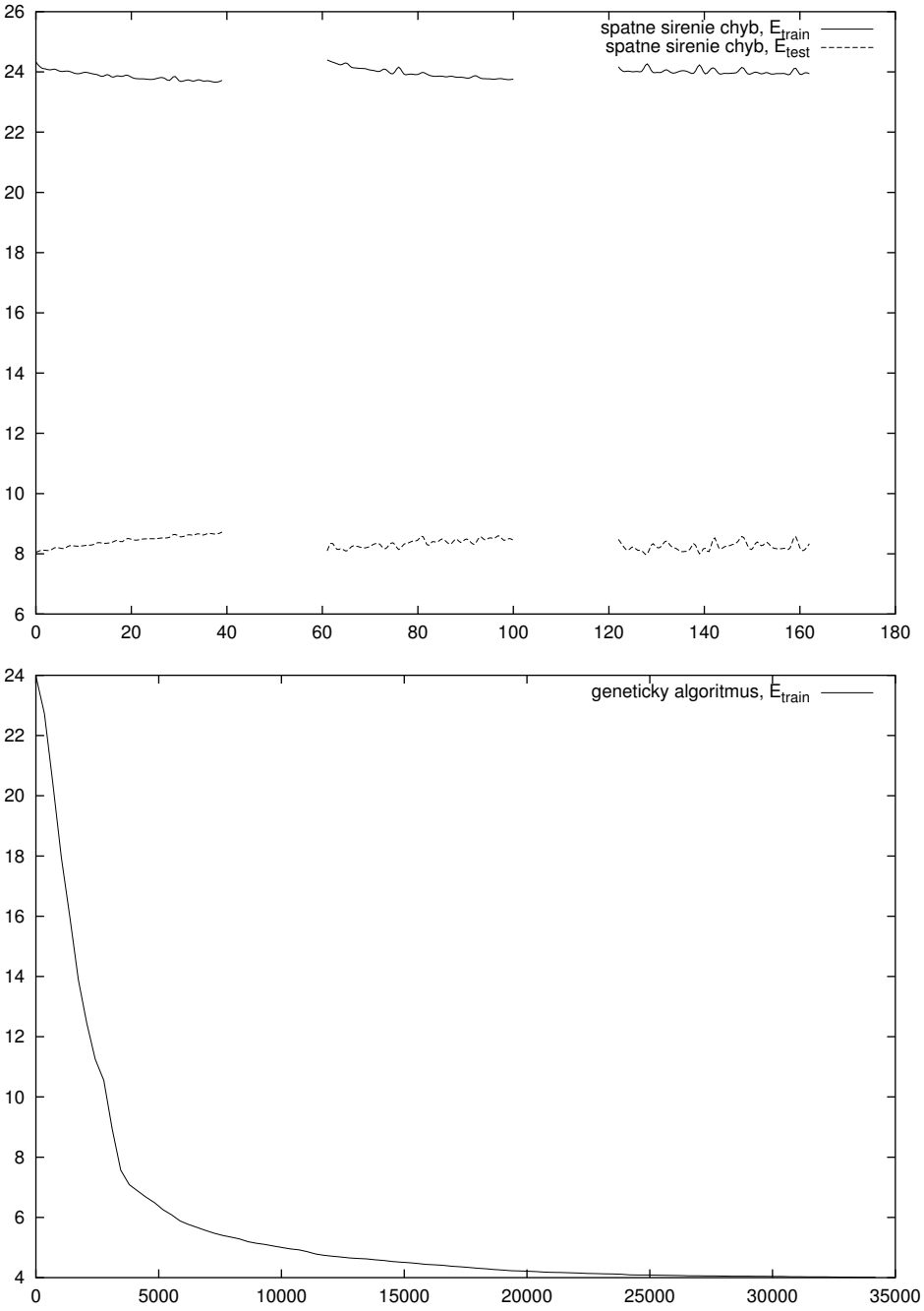
Obrázok B.2: Priebeh chýb pri tréningu neurónovej siete na problém *Boolovská funkcia 2*.



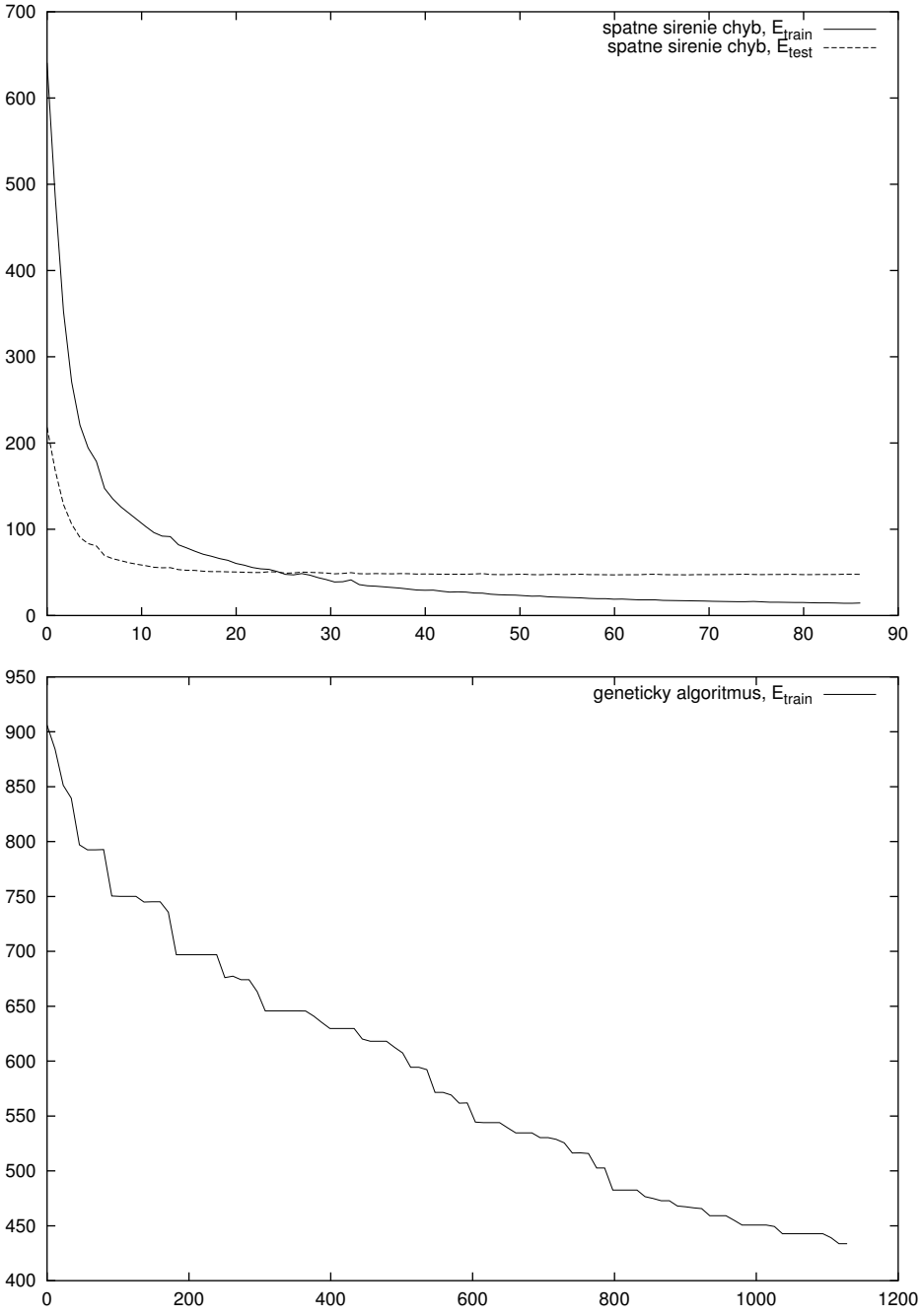
Obrázok B.3: Priebeh chýb pri tréningu neurónovej siete na problém *Boolovská funkcia 3*.



Obrázok B.4: Priebeh chýb pri tréningu neurónovej siete na problém *Násobenie*.



Obrázok B.5: Priebeh chýb pri tréningu neurónovej siete na problém *Parita*.



Obrázok B.6: Priebeh chýb pri tréningu neurónovej siete na problém *Splice*. Údaje z trénovania genetickým algoritmom sú bohužiaľ nekompletné.

Literatúra

[Kvasnička1997] Vladimír Kvasnička et al. *Úvod do teórie neurónových sietí.* IRIS, 1997.

[Kvasnička2000] Vladimír Kvasnička, Jiří Pospíchal a Peter Tiňo. *Evolučné algoritmy.* Vydavateľstvo STU v Bratislave, 2000.

[Návrat2002] Pavol Návrat et al. *Umelá inteligencia.* Vydavateľstvo STU v Bratislave, 2002.

[Rybár2002] Ján Rybár, Ľubica Beňušková a Vladimír Kvasnička, editori. *Kognitívne vedy.* Kalligram, 2002.