Použitie genetického algoritmu na adaptáciu neurónovej siete

Zadanie

Implementujte neurónovú sieť, feed-forward, s jednou vrstvou skrytých neurónov, so sigmoidálnou prechodovou funkciou. Použite genetický algoritmus na adaptáciu takejto neurónovej siete (jej váh a prahových koeficientov), aby sieť s čo najmenšou chybou interpretovala rôzne boolovské funkcie (boolovské hodnoty nech sú z numerických dôvodou reprezentované hodnotami 0.1 a 0.9)

Obsah

Popis použitých metód

Neurónová sieť

Implementovaná neurónová sieť je trojvrstvová feed-forward. Každý neurón je spojený so všetkými neurónmi predchádzajúcej aj nasledujúcej vrstvy. Počet vstupných neurónov sa rovná kardinalite zvolenej cieľovej funkcie (z priestorových a hlavne časových dôvodov budeme pracovať s kardinalitou menšou ako 10). Počet vnútorných neurónov som zvolil rovnaký ako počet vstupných neurónov, výstupný neurón je jediný (interpretovať sa budú boolovské funkcie). Váhy jednotlivých spojov sú vyjadrené reálnym číslom z intervalu [0,1] s presnosťou 11 číslic (typ Real jazyka Pascal). Prahové koeficienty jednotilvých neurónov majú rovnakú presnosť. Pre náš konkrétny problém – boolovské funkcie by však mala postačovať aj oveľa menšia presnosť, váhy aj prahové koeficienty budú v chromozóme kódované jediným bajtom, čo zodpovedá presnosti na 2-3 číslice. Boolovský vstup je reprezentovaný reálnymi číslami 0.1 a 0.9. Prechodová funkcia je klasický sigmoid $t_\alpha(\xi) = \frac{1}{1+e^{-\alpha\xi}}$ s voliteľným parametrom α (kde ξ je suma vstupov neurónu upravená hodnotou prahového koeficientu ν, $\xi_i = \nu_i + \Sigma w_{ij}x_j$). Cieľovou funkciou, ktorú sa budeme snažiť neurónovou sieťou aproximovať, je náhodná boolovská funkcia. Generovať sa bude na začiatku programu na základe štandardnej pascalovskej funkcie Random(). Počet vstupných premenných cieľovej funkcie je voliteľný.

Genetický algoritmus

Implementovaný je štandardný všeobecný genetický algoritmus, ktorý je potom následne prispôsobený nášmu problému adaptácie neurónovej siete. Populáciu tvorí voliteľný počet chromozómov. Každý chromozóm kóduje váhy a prahové koeficienty neurónovej siete. Konkrétny spôsob kódovania môže byť rôzny a bude rozobraný ďalej. Spôsob kódovania je dôležitý, aby množina chromozómov dostatočne pokrývala priestor riešení, t.j. aby počiatočným náhodným generovaním chromozómu a aj mutáciami počas procesu evolúcie vznikali zmysluplné kódy neurónovej siete. Okrem toho od spôsobu kódovania závisí aj efektívnosť kríženia, mal by byť taký, aby jednotlivé segmenty chromozómu kódovali podľa možnosti lokálne správne riešenia, ktoré možno navzájom kombinovať. Fitness každého chromozómu je vyhodnotená takto: na základe chromozómu sú adaptované váhy a prahové koeficienty neurónovej siete, a následne sú všetky možné vstupy neurónovej siete porovnané s hodnotami cieľovej funkcie. Konkrétna funkcia realizujúca toto porovnanie môže byť rôzna a jej výber by som vyčlenil ako samostatný problém, ktorý bude rozobraný ďalej. Voľba tejto funkcie je veľmi dôležitá, pretože od nej závisí prirodzený výber a následne aj celá úspešnosť genetického algoritmu. Po vyhodnotení fitness všetkých chromozómov populácie naleduje prirodzený výber chromozómov do ďalšej generácie. Výber je realizovaný klasickým algoritmom roulette wheel. Každá vybraná dvojica sa buď prenesie do ďalšej generácie bez zmeny, alebo sa na tejto dvojici vykoná reprodukčný proces, pravdepodobnosť jednoho alebo druhého prípadu je voliteľná. Samotný reprodukčný proces pozostáva z mutácie a kríženia. Operátor mutácie prebehne po celom chromozóme a s voliteľnou pravdepodobnosťou neguje jednotlivé bity chromozómu. Operátor kríženia dvoch chromozómov môže byť realizovaný voliteľne buď jednoduchým alebo viacnásobným krížením. Pri jednoduchom krížení sa v náhodnom mieste chromozómy skrížia, jediný krát a pri každej reprodukcii. Pri viacnásobnom krížení sa prebehne pozdĺž celého chromozómu a s voliteľnom pravdepodobnosťou sa chromozómy v danom mieste skrížia. Čiže v konečnom dôsledku môže dôjsť k výmene viacerých segmentov chromozómu medzi dvoma rodičmi. Takýmto spôsobom sa vytvorí nová generácia a tento proces sa opakuje stanovený počet generácii.

Problém kódovania neurónovej siete v chromozóme

Pri implementácii sa stretneme s úlohou ako zakódovať jednotlivé váhy a prahové koeficienty neurónovej siete v chromozóme. Z hľadiska teórie by mal spôsob kódovania spĺňať nasledovné kritéria: aby množina chromozómov dostatočne pokrývala priestor riešení, t.j. aby počiatočným náhodným generovaním chromozómu a mutáciami počas procesu evolúcie vznikali zmysluplné kódy neurónovej siete a aby množina všetkých chromozómov obsahovala aj optimálne riešenie. Okrem toho, jednotlivé segmenty chromozómu by mali kódovať podľa možnosti lokálne správne riešenia, ktoré možno navzájom kombinovať, aby bolo efektívne kríženie. Ak sa kódovanie zvolí nevhodne, krížením budú vznikať slabé chromozómy a postup genetického algoritmu bude závisieť iba na ‘slepej’ mutácii, čo znamená spomalenie algoritmu.

Problém pokrytia priestoru riešení: Nakoľko neurónová sieť má aproximovať boolovské funkcie, nie je potrebné kódovať váhy spojov veľmi presne. 256 hodnôt váh by malo byť dostatočne veľa, takýto priestor chromozómov by mal pokrývať nastavenie váh v optimálnom prípade. Druhou vecou je spôsob kódovania prahových koeficientov. Pre kódovanie prahového koeficientu je potrebné uvažovať o tom, aká môže byť suma vstupov do neurónu. Keďže váhy spojov sú z intervalu [0,1] a obor hodnôt prechodovej funkcie je interval [0,1], znamená

to, že každý so vstupných spojov neurónu môže prispieť hodnotou 0 až 1. Celkovo teda môže byť suma vstupov z intervalu [0,n] kde n je počet vstupných spojov neurónu. Hodnota prahového koeficientu by mala pokrývať okolie tohto intervalu, okrem toho by mala zaistiť možnosť záporných vstupov do prechodovej funkcie (aby záporné aj kladné hodnoty vsupovali do neurónu s rovnakou pravdepodobnosťou). Podľa týchto kritérii som sa nakoniec rozhodol voliť prahové koeficienty z intervlu [-7/6n,1/6n], sa variabilitou 256 hodnôt:

```
a := _b^/256*c_BaseFunction^.cInputWidth;
DecodeThreshold := -7/6*a+1/6*a;
```

Čo sa týka druhého problému (rozdelenie na lokálne podproblémy), je tažké vymyslieť nejaké špeciálne riešenie, pretože každý neurón má potenciálne vplyv na celý zvyšok neurónovej siete, preto nemožno nijakým spôsobom spoľahlivo rozdeliť problém na lokálne podproblémy. Nakoniec som použil priamočiare riešenie kódovania váh a prahových koeficientov postupne po jednotlivých neurónoch.

Problém vyjadrenia úspešnosti neurónovej siete (fitness)

Ďalším problémom, s ktorým sa pri implementácii stretneme, je vyjadrenie úspešnosti neurónovej siete. Od tejto funkcie závisí jednak prirodzený výber, jednak diverzita populácie. Prirodzený výber musí zaistiť prežitie tých chromozómov, ktoré sa najviac podobajú na cieľovú funkciu. Nízka diverzita populácie zase signalizuje stagnáciu, pretože nie je čo kombinovať a postup je závislý už iba na 'slepej' mutácii. Z teórie: funkcia sily (fitness) chromozómu je zobrazenie F: P $\rightarrow$ R_+, ktoré každému chromozómu priradí kladné reálne číslo, vyjadrujúce jeho úspešnosť. Na základe sily chromozómu potom prebieha prirodzený výber (algoritmus roulette-wheel), preto má zvolenie tejto funkcie závažný vplyv na rýchlosť aj úspešnosť algoritmu. V našom prípade, na základe každého chromozómu bude adaptovaná neurónová sieť. Takto vytvorenou neurónovou sieťou budeme spracovávať boolovské vektory a na rozdieloch medzi výstupmi neurónovej siete a výsledkami cieľovej funkcie založíme výpočet sily chromozómu. Experimentálne som skúšal nasledujúce funkcie :

```
jednoduchá suma rozdielov neurónovej siete a cieľovej funkcie
{Fitness := correl+exp(disper*2)-1;}
{Fitness := correl+disper2/4;     }
Fitness := sqrt(correl);
{Fitness := sqrt(correl+disper2/4);}
{Fitness := correl;}
{Fitness := correl/n/Sqrt(disper1*disper2);}
```

kde correl je kovariancia medzi NS a cieľovou funkciou, disper1, disper2 sú disperzie cieľovej funkcie a NS.
Ako najvhodnejšia sa experimentálne ukázala funkcia Sqrt(correl), ktorá okrem toho že vhodne hodnotí podobnosť medzi NS a CF tak aj zachováva väčšiu diverzitu populácie, čím sa predchádza stagnácii. Bližší popis funkcií je v stati Experimentálne výsledky.

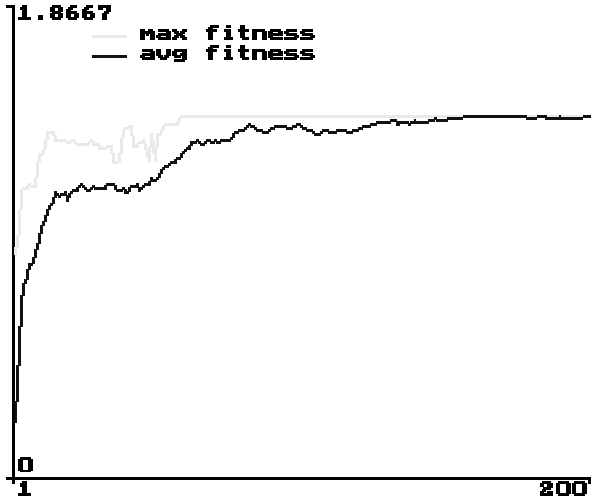
Príklad projektu

Výpis vstupného súboru:

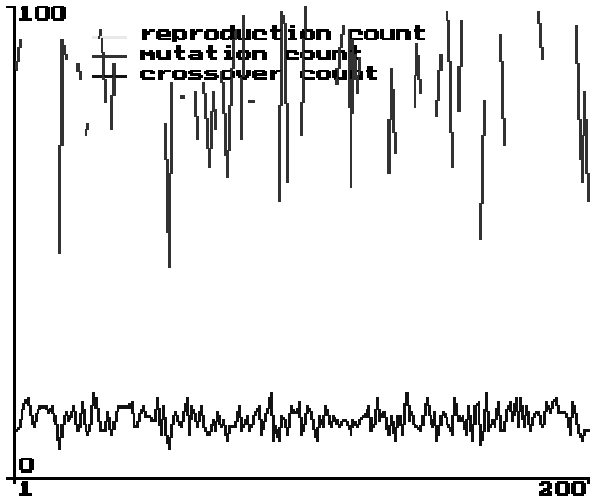
```
[POPULATION]
50   ;PopulationSize
0.5  ;ReproductionFactor  {probability of reproduction process, when creating next generation member}
0.02 ;MutationFactor  {probability of a one-bit mutation}
0    ;CrossOverType   {0=single 1=multiple}
0.01  ;CrossOverFactor {for multiple crossovers, probability of a crossover at one position}
200   ;MaxNumberOfGenerations

[BASEFUNCTION]
4  ;InputWidth    {cardinality of a base function}
0.1 ;ZeroValue     {Boolean->Real conversion}
0.9 ;OneValue      {Boolean->Real conversion}
0.5 ;Threshold     {Real->Boolean conversion}

[NEURALNETWORK]
4  ;first (input) layer neuron count
4  ;second layer neuron count
1  ;third (output) layer neuron count
4  ;SigmoidParameter
3  ;LayerCount

[STATISTICS]
```

```
1   ;ShowCrossoverCount
1   ;AnimateDelay  {how often to display progress}
```

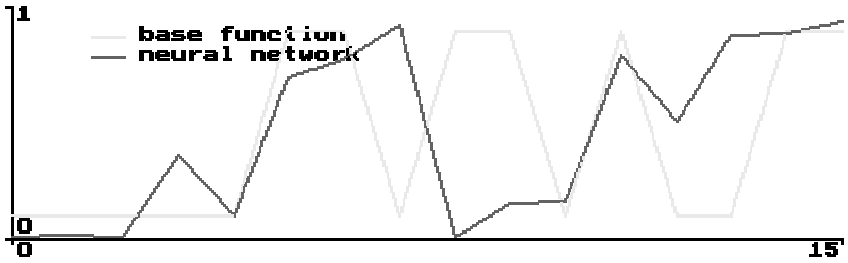
Výstupy projektu:



Graf dosiahnutej fitness



Reprodukčné štatistiky



Dosiahnutá adaptácia neurónovej siete

Vstupy a výstupy projektu

Vstupný súbor:

```
PROJEKT3 TXT           849
```

Výstupné súbory:

```
PROJEKT3 CHR           502
PROJEKT3 FNC        32,878
PROJEKT3 STA        24,012
```

Vstupný súbor je textový, jeho obsah možno modifikovať ľubovoľným editorom.
Výstupné súbory sú binárne. Na ich prehliadanie je určený program DISP.EXE.

Súbor s príponou .txt je špecifikačný súbor. Tento súbor obsahuje všetky voliteľné parametre programu, ako napríklad pravdepodobnosť mutácie, veľkosť populácie alebo architektúru neurónovej siete.
Výstupný súbor s príponou .chr obsahuje najúspešnejší dosiahnutý chromozóm.
Výstupný súbor s príponou .fnc obsahuje cieľovú funkciu, vygenerovanú v tomto jednom konkrétnom prípade.
Výstupný súbor s príponou .sta obsahuje základné štatistiky o evolučnom procese, ako napríklad maximálnu dosiahnutú fitness pre každú generáciu, alebo počet mutácií pri každej generácii.

Popis vstupného súboru

- Parametre uvedené vo vstupnom súbore sú obmedzené veľkosťou jednotlivých programových štruktúr zvolenou počas kompilácie.

[POPULATION]

50 ;PopulationSize
Integer [2,50]
Veľkosť populácie.

0.5 ;ReproductionFactor {probability of reproduction process, when creating next generation member}
Real [0,1]
Faktor reprodukcie. Pri tvorbe ďalšej generácie sa vyberú dva chromozómy na základe ich fitness. Tieto chromozómy buď vstupujú do reprodukčného procesu, alebo sa do ďalšej generácie dostávajú nezmenené. O tom, ktorá možnosť nastane, rozhoduje tento parameter. 1= reprodukčný proces nastane vždy.

0.02 ;MutationFactor {probability of a one-bit mutation}
Real [0,~0.3]
Pravdepodobnosť mutácie na jednom bite chromozómu. Mutovať môže ľubovoľný bit chromozómu, teda pravdepodobnosť mutácie v jednom reprodukčnom procese je okrem toho úmerná dĺžke chromozómu.

0 ;CrossOverType {0=single 1=multiple}
Integer [0,1]
Typ kríženia.
0 = jednoduché kríženie. Pri každom reprodukčnom procese sa stane jedno prekríženie chromozómov na náhodnom mieste.
1 = viacnásobné kríženie. Pri reprodukčnom procese sa postupuje pozdĺž chromozómu a v každom mieste môže nastane s pravdepodobnosťou CrossOverFactor kríženie, teda pri jednom reprodukčnom procese si chromozómy môžu vymeniť viacero segmentov. Pri viacnásobnom krížení sa nepostupuje po bitoch, ale po bytoch.

0.01 ;CrossOverFactor {for multiple crossovers, probability of a crossover at one position}
Real [0,~0.3]
Pri viadnásobnom krížení, pravdepodobnosť kríženia v jednom mieste chromozómu.

200 ;MaxNumberOfGenerations
Integer [2,1000]
Počet generácií evolučného procesu.

[BASEFUNCTION]

4 ;InputWidth {cardinality of a base function}
Integer [0,10]
Počet vstupných premenných cieľovej funkcie.

0.1 ;ZeroValue {Boolean->Real conversion}
Real [0,1]

Pri konverzii boolovských hodnôt na reálne, hodnota zodpovedajúca FALSE.

0.9 ;OneValue {Boolean->Real conversion}

Real [0,1]

Pri konverzii boolovských hodnôt na reálne, hodnota zodpovedajúca TRUE.

0.5 ;Threshold {Real->Boolean conversion}

Real [0,1]

Pri konverzii reálnych hodnôt na boolovské, prahová hodnota oddeľujúca TRUE a FALSE.

[NEURALNETWORK]

4 ;first (input) layer neuron count

Integer[1,15]

Počet neurónov 1. Vrstvy.

4 ;second layer neuron count

Integer[1,15]

Počet neurónov 2. Vrstvy.

1 ;third (output) layer neuron count

Integer[1,15]

Počet neurónov 3. Vrstvy.

4 ;SigmoidParameter

Real[~0.2,~10]

Parameter prechodovej funkcie, určuje jej strmosť.

3 ;LayerCount

Integer[1,3]

Počet vrstiev neurónovej siete.

[STATISTICS]

1 ;ShowCrossoverCount

Integer[0,1]

Či zobrazovať v grafe počet krížení. (Pri jednoduchom krížení sa tento počet rovná počtu reprodukcii)

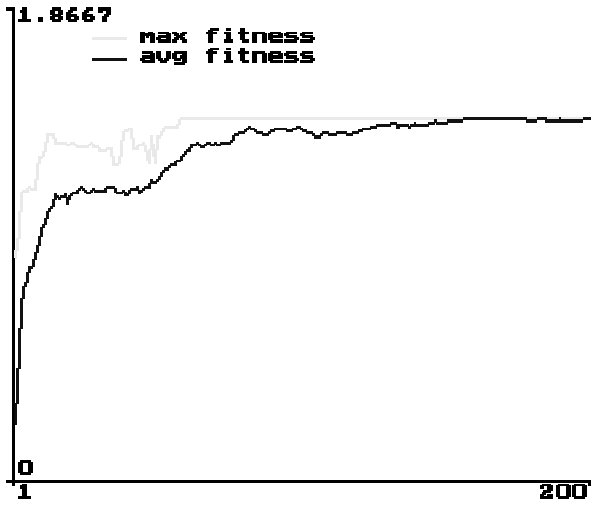
1 ;AnimateDelay {how often to display progress}

Integer[0,~50]

Pri behu programu, ako často (v generáciach) zobrazovať grafy ukazujúce postup evolučného procesu. (Pri veľkom počte generácii môže častý refresh iba zbytočne spomaľovať)

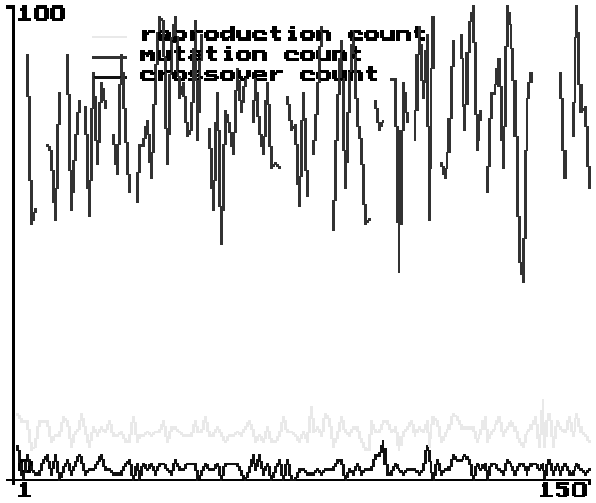
Popis výstupných grafov

Graf dosiahnutej fitness: Tento graf nám dáva prehľad o postupe evolučného procesu. Na x-ovej osi sú generácie, na y-ovej osi je dosiahnutá fitness generácie. Graf obsahuje dve zložky, maximálnu fitness generácie a priemernú fitness generácie. Čím strmšie stúpa graf maximálnej fitness, tým je evolučný proces úspešnejší. Ak sa priemerná fitness blíži maximálnej, pravdepodobne to indikuje malú diverzitu populácie a následnú stagnáciu. Číslovanie y-ovej osi nie je štandardizované, závisí od zvolenej metódy počítania fitness.



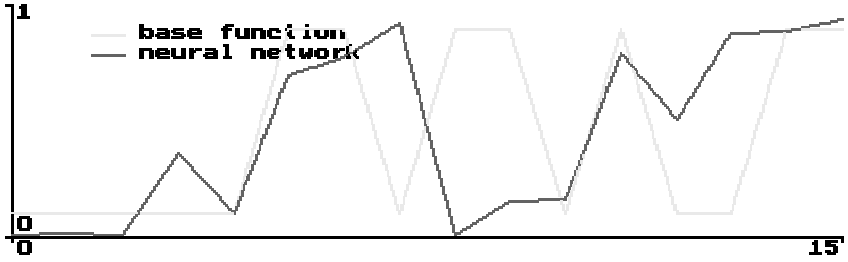
Graf dosiahnutej fitness

Graf reprodukčných štatistík: Tento graf dáva prehľad o reprodukčnom procese. Na x-ovej osi sú generácie, na y-ovej osi je príslušná početnosť reprodukcii, mutácii a krížení. Jednotlivé zložky reprodukčného procesu sa regulujú príslušnými pravdepodobnosťami vo vstupnom súbore. Tento graf znázorňuje reálny priebeh reprodukčného procesu na základe vstupných pravdepodobnotí. Graf obsahuje tri zložky. Provou je počet reprodukcií pri tvorbe ďalšej generácie, zvyšní príslušníci ďalšej generácie boli prenesení priamo z predchádzajúcej, bez reprodukčného procesu. Druhou zložkou je počet krížení. V prípade jednoduchého kríženia sa zhoduje s počtom reprodukcií, v prípade viacnásobného kríženia je určený pravdepodobnostne. Treťou zložkou je celkový počet jednobitových mutácii, ktoré nastali pri tvorbe ďalšej generácie.



Reprodukčné štatistiky

Graf adaptácie neurónovej siete: Tento graf dáva detailný prehľad o najúspešnejšej adaptácii neurónovej siete – najúspešnejšom chromozóme. Na x-ovej osi sú všetky možné vstupy cieľovej funkcie, na y-ovej osi je príslušný výsledok cieľovej funkcie a výstup neurónovej siete. Pri vyhodnotení každej generácie sa zoberie chromozóm s najvyššou fitness, na základe neho sa adaptuje neurónová sieť a vyhodnotia sa jej výstupy, ktoré možno v tomto grafe porovnať s výsledkami cieľovej funkcie.

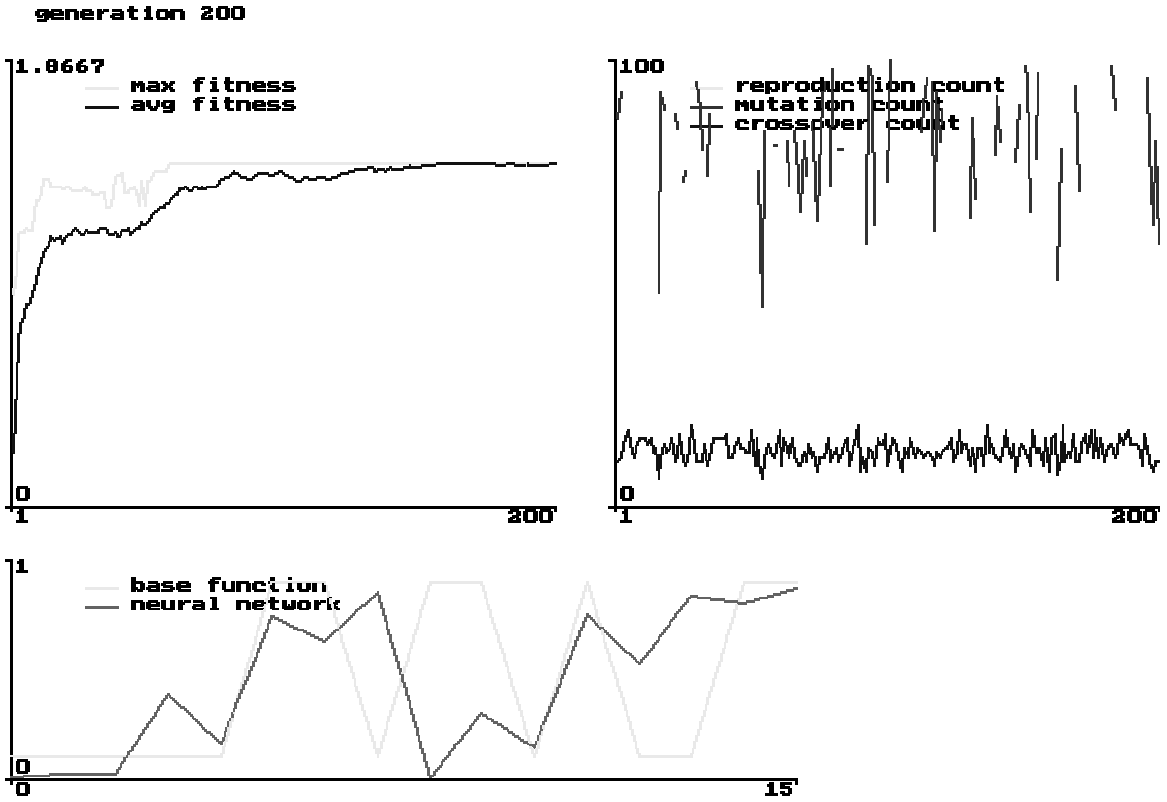


Dosiahnutá adaptácia neurónovej siete

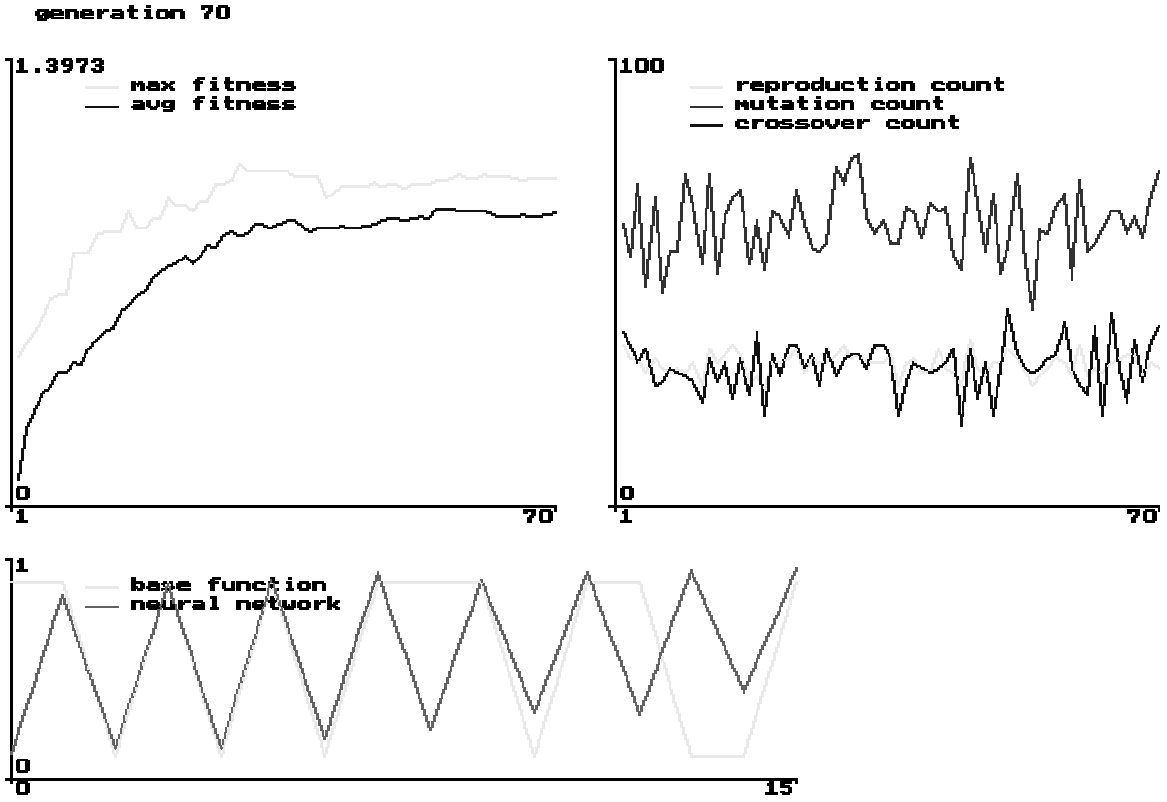
Experimentálne výsledky

Grafy

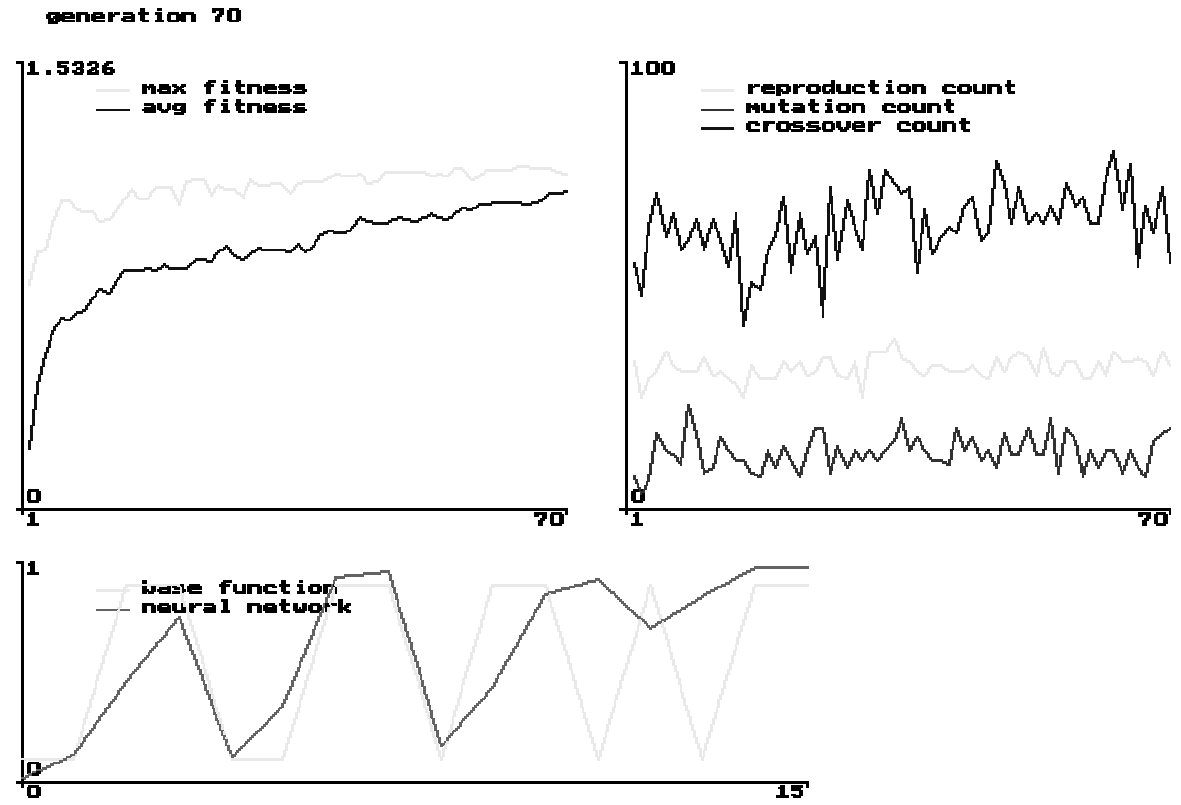
[POPULATION]
50 ;PopulationSize
0.5 ;ReproductionFactor {probability of reproduction process, when creating next generation member}
0.02 ;MutationFactor {probability of a one-bit mutation}
0 ;CrossOverType {0=single 1=multiple}
0.01 ;CrossOverFactor {for multiple crossovers, probability of a crossover at one position}
200 ;MaxNumberOfGenerations



[POPULATION]
80 ;PopulationSize
0.8 ;ReproductionFactor {probability of reproduction process, when creating next generation member}
0.005 ;MutationFactor {probability of a one-bit mutation}
1 ;CrossOverType {0=single 1=multiple}
0.04 ;CrossOverFactor {for multiple crossovers, probability of a crossover at one position}
70 ;MaxNumberOfGenerations



[POPULATION]
80 ;PopulationSize
0.8 ;ReproductionFactor {probability of reproduction process, when creating next generation member}
0.001 ;MutationFactor {probability of a one-bit mutation}
1 ;CrossOverType {0=single 1=multiple}
0.08 ;CrossOverFactor {for multiple crossovers, probability of a crossover at one position}
70 ;MaxNumberOfGenerations



Komentár

Pre zlepšenie výsledkov bolo potrebné dolaďovať nasledovné parametre:

Funkcia fitness:

`jednoduchá suma rozdielov neurónovej siete a cieľovej funkcie`

Nevhodná.

```
{Fitness := correl+disper2/4;      }
```

Vhodnejší ja kovariancia(súčín rozdielov (hodnta-aritmetický priemer hodnôt)). Disperziu som pridal, aby som docielil elimináciu viac-menej konštantných funkcii, no ukázalo sa že to nie je potrebné.

```
{Fitness := sqrt(correl+disper2/4);}
```

Funkciou Sqrt sa dá docieliť väčšia diverzita. (zmenšuje väčšie hodnota a zväčšuje menšie)

```
{Fitness := correl;}
Fitness := sqrt(correl);
```

Táto funkcia sa ukázala ako najvhodnejšia. (spôsob porovnania aj diverzita)

```
{Fitness := correl/n/Sqrt(disper1*disper2);}
```

Korelácia je štandardizovaná kovariancia. Neovplyvňuje proces výberu, iba mení výsledné hodnoty na interval [0,1] (hodnota korelácie závisí od absolútnych hodnôt a počtu hodnôt)

Diverzita:

Algoritmus má problémy so skorou stagnáciou a s tým súvisiacou nízkou diverzitou populácie. Diverzitu sa mi podarilo zvýšiť zmenou funkcie Fitness, väčšou veľkosťou populácie a vhodným nastavením faktorov mutácie a kríženia.

Reprodukčné parametre:

Kríženie: Vhodnejšie sa zdá byť viacnásobné kríženie. Častejším viacnásobným krížením sa dá docieliť väčšia diverzita populácie. Z výsledkov vyplýva, že progres algoritmu je hlavne výsledkom kríženia, nie mutácii.
Mutácia: Mutácie iba zriedka výraznejšie zvyšujú fitness, osvedčili sa skôr nižšie pravdepodobnosti mutácie.
Veľkosť populácie: Ovplyvňuje diverzitu, menšie populácie skoro stagnujú. Preto sú výhodnejšie väčšie populácie.

Záver

Metóda dokázala isté výsledky, váhy neurónovej siete nastavené na základe evolučného procesu produkovali výstup z väčšej miery zodpovedajúci vzorovej funkcii, avšak výsledky nie sú ideálne. Hlavným problémom algoritmu je dosť skorá stagnácia a relatívne malá diverzita populácie.