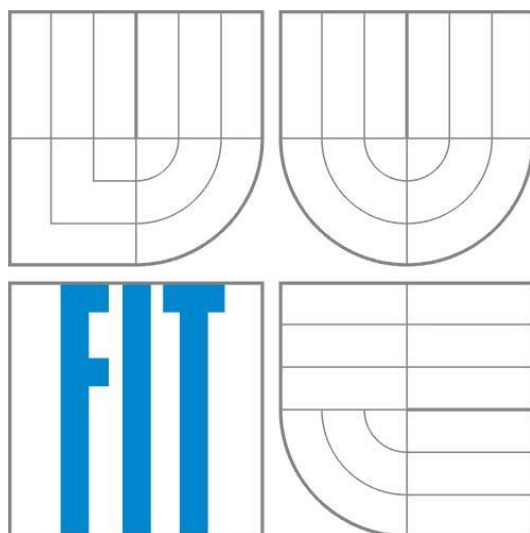


Fakulta Informačních Technologii

Vysoké Učení Technické v Brně



Dokumentácia k semestrálnemu projektu pre predmet

Soft Computing

Varianta zadania: Demonstrácia učenia algoritmom Quick propagation,
C/C++

Autor: Bc. Martin Fajčík

Dátum: 05.12.2015

Úvod

V rámci kurzu SFC sme mali možnosť oboznámiť sa z novými metódami strojového učenia využívajúcimi nepriame, aproximačné prístupy soft computingu. Popri fuzzy logike, či genetických algoritmoch sú jedným z takýchto prístupov aj umelé neurónové siete (*artificial neural network* - ANN).

Jednou z dnes najpopulárnejších typov neurónových sietí je sieť založená na algoritme *back-propagation* (BP_{ROP}). Jedná sa o doprednú ANN, ktorej váhy sú nastavované na základe spätného šírenia chyby. V roku 1988 navrhol Scott E. Fahlmann modifikáciu tohto algoritmu nazvanú *quick-propagation* (Q_{PROP})^[1].

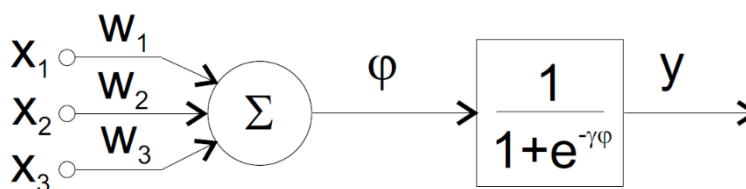
2 Algoritmy

2.1 Algoritmus Back Propagation

Tréning siete pomocou tohto algoritmu prebieha pomocou postupného cyklického výberu prvkov z trénovacej množiny a následnou aplikáciou týchto štyroch krokov:

[1] Výpočet výstupov (*Feedforward*)

- Vstupy siete predstavujú vstupy neurónov vstupnej vrstvy ANN.
- Vstupy neurónov skrytej vrstvy a neurónov výstupnej vrstvy predstavujú výstupy neurónov predchádzajúcej vrstvy
- Výstupom ANN sú výstupy neurónov výstupnej vrstvy
- Výpočet výstupu pre neurón, ktorého aktivačnou funkciou je *sigmoida* je daný spôsobom:



Obrázok 1: Kde x_i sú vstupmi, w_i sú váhami vstupov, $i = 1..3$, $\phi = \sum x_i * w_i$, γ je ostrnosť sigmoidy, a y je výstupom neurónu. Zdroj: ^[3].¹

[2] Výpočet chyby na výstupnej vrstve (*Root Mean Square*)

- Pre celú neurónovú sieť je možné získať hodnotu chybovej (energetickej) funkcie E ako:

$$E = \frac{1}{2} \sum_{i=1}^n (y_i - d_i)^2$$

kde

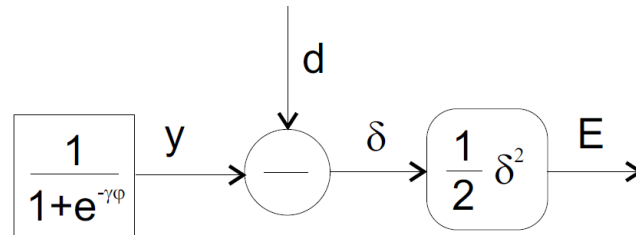
n – počet neurónov výstupnej vrstvy

y_i – i -tý výstup

d_i – i -tý požadovaný výstup

¹ Mimo obrázkov 1 a 2 v kapitole 2 budeme znak γ nazývať aj λ .

Pre výstupný neurón y_i potom bude naša schéma vyzerat' takto:



Obrázok 2: Sémantika obrázku je rovnaká ako u obrázku 1, d je požadovaný výstup neurónu, E je chybová funkcia. Zdroj: [3].

V tomto bode nás okrem chybovej funkcie E , po ktorej nájdení minima ukončíme učenie ANN zaujíma hlavne hodnota δ , podľa ktorej ďalej získame gradient. Chybová funkcia E predstavuje vlastne zakrivenú plochu a našim cieľom je čo najviac sa priblížiť k jej minimu. Preto je nutné zistiť smer kroku k minimu (*gradient*) a určiť veľkosť tohto kroku (μ). Keďže pre neuróny poslednej vrstvy platí:

$$\frac{\partial E}{\partial {}^L y_j} = -(d_j - {}^L y_j)$$

kde

L – je počet vrstiev siete, resp. index poslednej vrstvy

j – je index neurónu vo vrstve

Hodnotu δ pre poslednú vrstvu teda získame ako:

$${}^L \delta_j = -\frac{\partial E}{\partial {}^L y_j} \frac{\partial {}^L y_j}{\partial {}^L u_j} = (d_j - {}^L y_j) \lambda {}^L y_j (1 - {}^L y_j)$$

kde

l – je index vrstvy

n_l – je počet neurónov v vrstve

[3] **Spätná propagácia chýb** (*back-propagation*)

- Pre zvyšné vrstvy je s použitím vlastností reťazového pravidla (*chain rule*) odvodiť vzťah [2], na základe ktorého je možné získať hodnoty δ takto:

$${}^{l-1} \delta_j = \sum_{k=1}^{n_l} ({}^l \delta_k {}^l w_{kj}) \frac{\partial {}^{l-1} y_j}{\partial {}^{l-1} u_j} = \sum_{k=1}^{n_l} ({}^l \delta_k {}^l w_{kj}) \lambda {}^{l-1} y_j (1 - {}^{l-1} y_j)$$

kde

l – je index vrstvy

n_l – je počet neurónov v vrstve

[4] Úprava váh

- V rámci úpravy váh upravíme váhy w vstupov x pre jednotlivé neuróny o hodnotu Δ_w :

$$\Delta_{l_{w_{ji}}} = \mu \nabla E$$

kde

μ – veľkosť kroku

$$\nabla E = {}^l\delta_j {}^l x_j$$

- Tento krok býva cieľom optimalizácie rôznych modifikácií algoritmu *back-propagation*. Je možné napríklad pridať konštantu *momenta* (zotrvačná konštanta), ktorá nám umožní ľahšie prekonať lokálne minimum pri určitých tvaroch plochy E :

$$\Delta_{l_{w_{ji}}}(k+1) = \mu \nabla E + \alpha \Delta_{l_{w_{ji}}}(k)$$

kde

α – momentum

- Alebo napríklad predpokladať, že plocha má tvar paraboly a namiesto konštantných krokov v smere gradientu skúsiť aproximovať miesto kde leží jej minimum – myšlienka algoritmu *quick-propagation*.

2.2 Algoritmus Quick Propagation

Scott E. Fahlmann navrhol predpoklad že funkcia E bude mať tvar paraboly a tak je možné skočiť priamo k minimu. Okrem veľkosti kroku smerom, ktorý udáva gradient, tento algoritmus používa aj konštantu ε , ktorá udáva maximálnu veľkosť kroku. (Fahlman za tento parameter odporúča 1.75) [4].

Samotný algoritmus je inšpirovaný Newtonovou metódou a kvôli veľkosti jeho krokov sa sieť chová pri jeho použití viac chaoticky, než u klasického *back-propagation*.

Úprava kroku [4] **QPROP**:

- Vypočítame pre daný neurón ∇E a $\Delta_{l_{w_{ji}}}$
- Následne aplikujeme algoritmus:

```
dstep = 0
if (Δlwji(k) > 0) then
    if (∇E(k) >  $\frac{\varepsilon}{1+\varepsilon} \nabla E(k-1)$ )
        dstep = dstep + μΔlwji(k)
    else
        dstep = dstep + Δlwji(k-1)  $\frac{\nabla E(k)}{\nabla E(k) - \nabla E(k-1)}$ 
    if (∇E(k) > 0)
        dstep = dstep + ε∇E(k)
else if (Δlwji < 0) then
    if (∇E(k) <  $\frac{\varepsilon}{1+\varepsilon} \nabla E(k-1)$ )
        dstep = dstep + μΔlwji(k)
    else
        dstep = dstep + Δlwji(k-1)  $\frac{\nabla E(k)}{\nabla E(k) - \nabla E(k-1)}$ 
    if (∇E(k) < 0)
        dstep = dstep + ε∇E(k)
else
    dstep = dstep + ε∇E(k)
lwji = lwji + dstep
```

Algoritmus 1: Quick Propagation

3 Detaily implementácie²

Program pracuje ako konzolová aplikácia, ktorá prijíma parametre zadané užívateľom. Hlavnou súčasťou programu je trieda *QPNetwork*, ktorá sa stará o alokáciu siete, jej tréning, testovanie, získavanie informácií o sieti či dealokáciu siete. Pri tvorbe tejto triedy je nutné nastaviť konfiguračné parametre:

- koeficient učenia μ ,
- momentum α ,
- krivosť λ ,
- maximálna veľkosť kroku ε
- prah globálnej chyby, pri ktorej sa zastaví učenie *threshold*,
- maximálny počet iterácií siete pri učení *maxiterations*,
- počet vrstiev *L*,
- vektor udávajúci počet neurónov v jednotlivých vrstvách \vec{n} .

Parametre je potrebné nastaviť do konfiguračného súboru, ktorý je potom možné v programe načítať metódou `read_configuration` a pomocou štruktúry typu `Config` predať konštruktoru tejto triedy³.

```
layers_inputs_neurons 2 2 2 1    #2-count of network layers #2 - count of inputs
                                   #2-first layer neuron count #1 second layer neuron count
threshold 0.001
maxiterations 25000000
mi 0.7
momentum 0.05
lambda 1.0
epsilon 1.75 #value recommended by Scott E. Fahlmann
```

Príklad 1: Ukážka konfiguračného súboru

Učenie triedy je možné vyvolať metódou `QPNetwork::learn`, ktorej sú predané dáta, informácie o počte vzoriek dát a informácia o to či má učiť pomocou algoritmu BPROP alebo QPROP.

Testovanie siete je následne možné uskutočniť metódou `QPNetwork::test`, ktorej sú predané dáta a informácie o počte vzoriek dát. Trénovacíu, v respektíve testovaciu množinu dát je nutné vytvoriť v externom súbore a je možné je načítať pomocou metódy `read_datafile`.

```
4 3    #size of input matrix
0 0 0
0 1 1
1 0 1
1 1 0
```

Príklad 2: Ukážka vstupného dátového súboru.

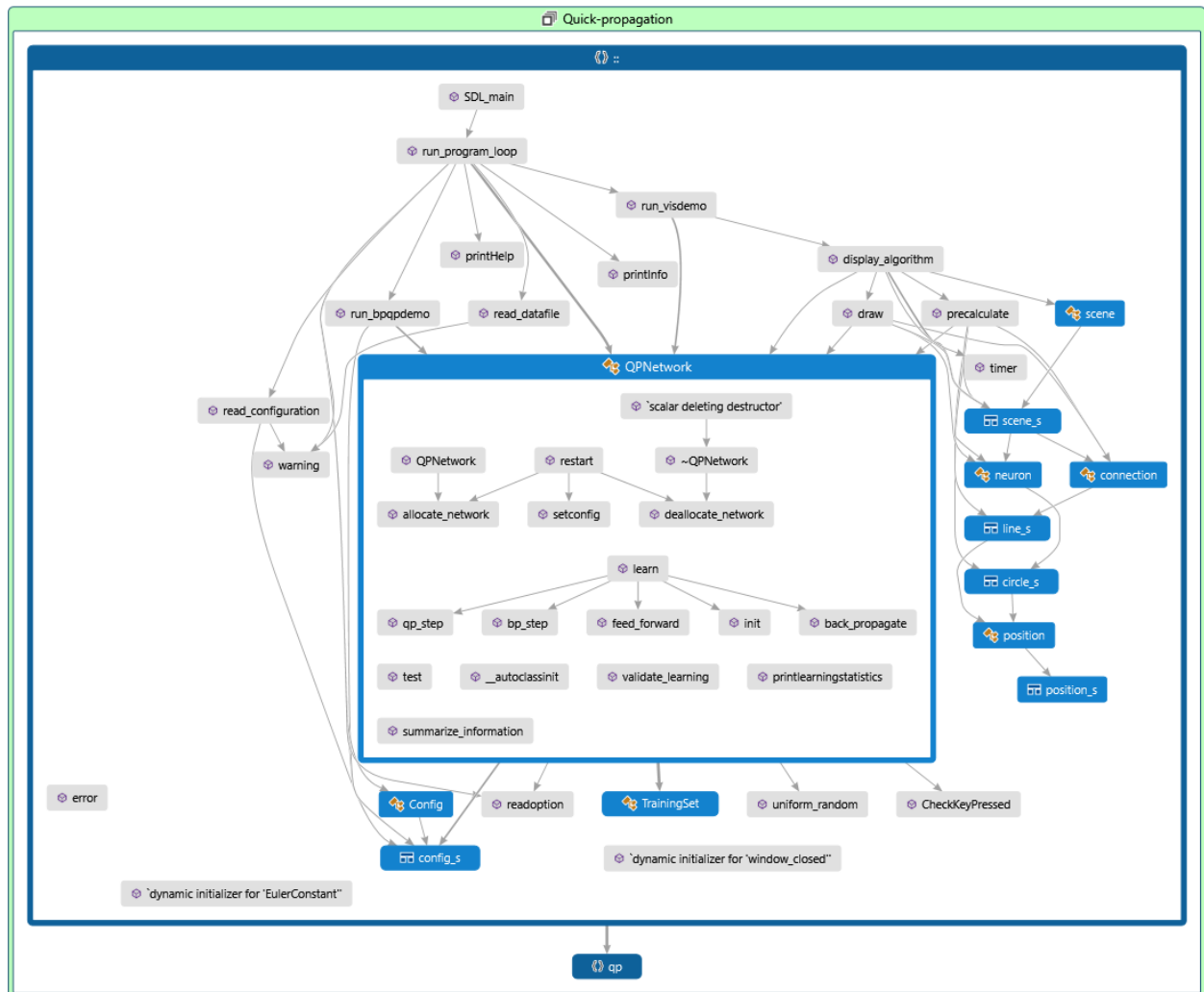
V prípade súboru určeného k učeniu je prvých *i* stĺpcov braných ako vstupy a zvyšné ako očakávané výstupy (kde *i* je počet vstupov siete).

V prípade súboru určeného k testovaniu siete sú všetky stĺpce brané ako vstupy

² Pri algoritme QPROP pomocou metódy sa v prvých dvoch iteráciách učenia (tzn. po výbere 2 prvkov z trénovacej množiny) využije algoritmus BPROP – dôvodom je určenie (*zdanlivo*) správneho smeru klesania

³ Alternatívou je metóda `QPNetwork::setconfig`, ktorou je možné modifikovať parametre siete po jej vytvorení.

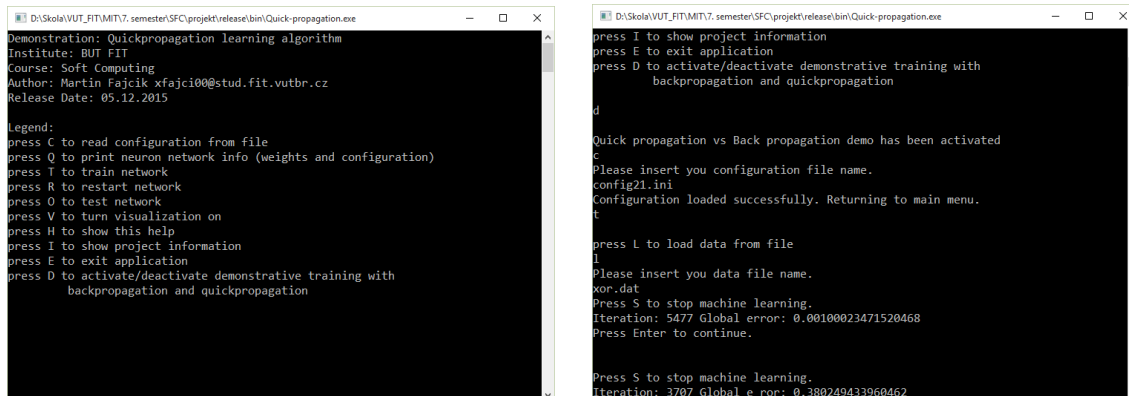
Za pomoci parametrov je možné nastaviť aj rôzne módy demonštrácie ako porovnanie rýchlosti učenia siete pomocou algoritmov BPROP a QPROP pri rovnakých počiatočných váhach, či vizualizácia učenia pomocou knižnice *SDL* a jej nadstavby *Cairo*. Vizualizácia pomocou *SDL* beží na samostatnom vlákne, aby čo najmenej zaťažovala výpočet. Pri spustení vizualizácie dôjde najprv k prepočítaniu pozícií prvkov neurónovej siete (funkcia *precalculate*) a následne k sústavnému renderovaniu scény neurónovej siete pomocou metódy *draw*.



Obrázok 3: Schéma tried, funkcií a závislostí kódu vygenerovaná nástrojom CodeMap

4 Ovládanie programu

Program je možné ovládať pomocou parametrov zadanych na štandardný vstup. Samotný program obsahuje v menu vysvetlivky parametrov, pomocou ktorých je ho možné ovládať.



```
D:\Skola\VUT_RIT\MIT\7. semester\SFC\projekt\release\bin\Quick-propagation.exe
Demonstration: Quickpropagation learning algorithm
Institute: BUT FIT
Course: Soft Computing
Author: Martin Fajcik xfajci00@stud.fit.vutbr.cz
Release Date: 05.12.2015

Legend:
press C to read configuration from file
press Q to print neuron network info (weights and configuration)
press T to train network
press R to restart network
press O to test network
press V to turn visualization on
press H to show this help
press I to show project information
press E to exit application
press D to activate/deactivate demonstrative training with
        backpropagation and quickpropagation

press I to show project information
press E to exit application
press D to activate/deactivate demonstrative training with
        backpropagation and quickpropagation

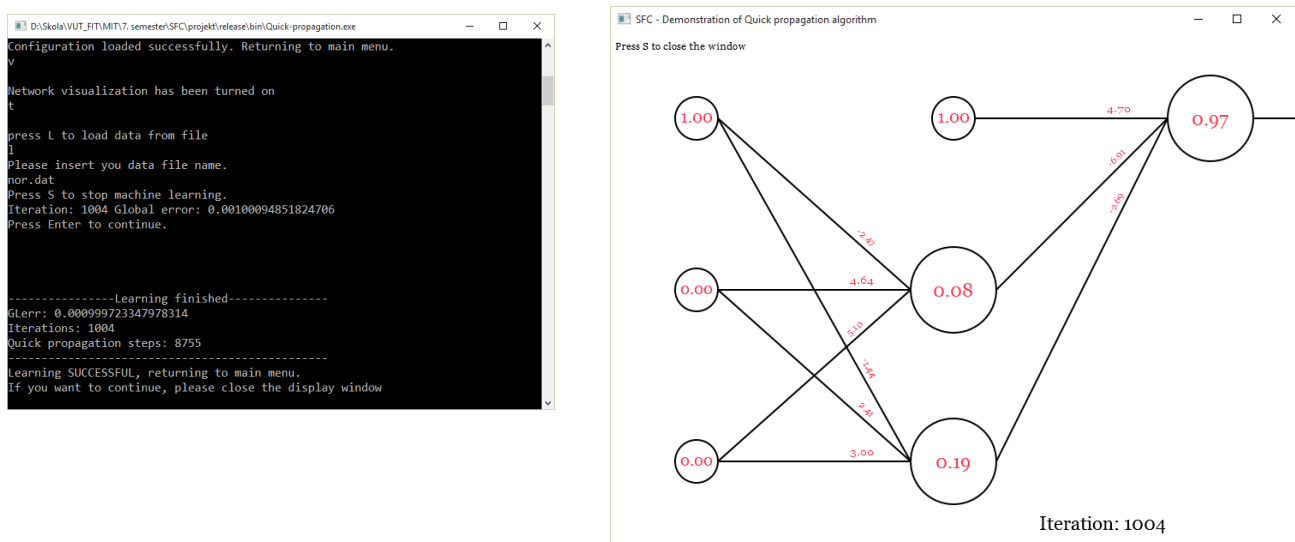
Quick propagation vs Back propagation demo has been activated
Please insert you configuration file name.
config21.ini
Configuration loaded successfully. Returning to main menu.

press L to load data from file
l
Please insert you data file name.
xor.dat
Press S to stop machine learning.
Iteration: 5477 Global error: 0.00100023471520468
Press Enter to continue.

Press S to stop machine learning.
Iteration: 3707 Global error: 0.380249433960462
```

Obrázok 4: Konzolové rozhranie užívateľa o všetkých akciách informuje. Obrázok demonštruje spustenie demonštrácie učení BPROP a QPROP.

Pomocou parametru R je možné sieť reštartovať (dôjde k opätovnej inicializácii náhodných váh), zmeniť konfiguráciu (pri zmene topológie siete dôjde k automatickému reštartu) alebo je možné si pomocou parametru Q vytlačiť informácie o sieti. Priebeh učenia podľa algoritmu QPROP je navyše možné vizualizovať pomocou parametru V a následným spustením tréningovania.



Obrázok 5: Vizualizácie siete pri učení problému NOR. Pri spustení tréningovania QPROP navyše program ukazuje počet skokov prevedených pomocou QPROP (else vetvy Algoritmu 1)

Záver

Program, ktorý vznikol v rámci tohto projektu umožňuje experimentovať a demonštrovať chod algoritmu *quick-propagation* ako aj *back-propagation*. V rámci experimentov zo základnými logickými funkciami (*OR*, *AND*, *NOR*, *XOR*) sa modifikácia v podobe QPROP javí ako rýchlejšia (*nie však pri každej konfigurácii*), je však nutné podotknúť že čím rozmanitejšia je zakrivená plocha funkcie *E* tým menej vhodné je použitie tejto modifikácie (keďže dochádza k „prestrelu“ minima za pomoci nepresnej aproximácie priveľkého kroku).

Algoritmus taktiež nerieši problém lokálnych miním funkcie *E* a preto môže učenie siete uviaznuť a skončiť neúspešne. V niektorých prípadoch nie je možné ani pomocou konštanty *momenta* prekonať tieto minimá a je vhodnejšie reštartovať učenie z novými náhodne zvolenými hodnotami váh, či skúsiť implementovať ďalšie modifikácie BPROP ako sú SuperSAB alebo Resilient propagation [5].

Bibliografia

- [1] Fahlman, Scott E. *An empirical study of learning speed in backpropagation networks* [online]. 1988 [cit. 2015-12-06]. Dostupné z: <http://repository.cmu.edu/cgi/viewcontent.cgi?article=2799>
- [2] Zbořil, F. V.: *Soft Computing : SFC*. Brno: Vysoké učení technické, Fakulta informačních technologií 2015
- [3] Jirsík, V.: *Umělá Inteligence : FEKT-LUIN*. Brno: Vysoké učení technické, Fakulta elektrotechniky a komunikačních technologií, 2013
- [4] Ventura, João Carlos Negrão. *Neural networks implementation in parallel distributed processing systems* [online]. 2006. [cit. 2015-12-06]. Dostupné z: <https://venturas.org/sites/venturas.org/files/mydmbp.pdf>
- [5] Braga, Antonio P., PARMA a Benjamim R. MENEZES. *Back-propagation learning guided by control technique* [online]. 1999 [cit. 2015-12-06]. Dostupné z: <http://www.cpdee.ufmg.br/~parma/artigos/smcbp.pdf>