

# Neurónové siete – Projekt 1

Juraj Onderik

11.3.2004

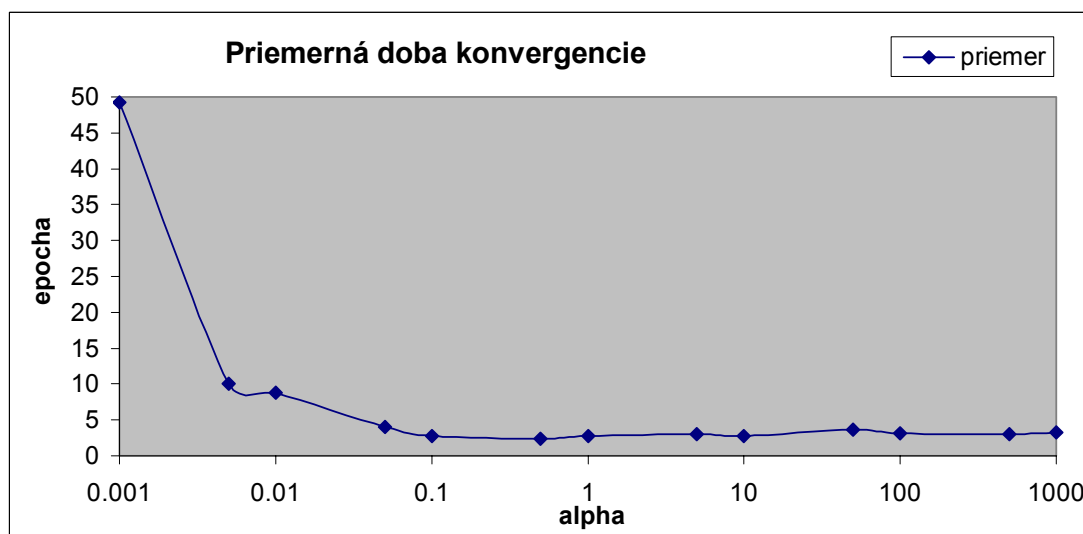
## Časť A.

V prvom experimente som pre rôzne rýchlosti učenia 10 krát náhodne (-0.1,+0.1) nastavil váhy perceptrónu a zmeral jeho dobu konvergencie na štandardnej OR-trénovacej množine. Údaje sú uvedené v **Tab1** a **Graf1**.

**Tab1**

alpha	experiment - doba konvergencie										priemer
0.001	168	10	45	0	67	98	5	18	63	18	49.20
0.005	2	12	17	10	4	5	12	30	9	0	10.10
0.01	6	2	3	7	17	6	0	13	14	20	8.80
0.05	5	3	4	8	5	0	3	5	3	4	4.00
0.1	4	1	3	0	3	4	5	1	3	4	2.80
0.5	3	3	3	2	3	4	0	3	3	0	2.40
1	3	2	3	2	3	3	3	3	3	3	2.80
5	4	2	2	4	3	2	4	3	3	3	3.00
10	3	2	3	3	2	3	4	2	3	3	2.80
50	5	4	4	4	3	5	2	4	2	3	3.60
100	3	2	3	4	3	4	3	3	4	3	3.20
500	2	3	3	4	3	3	2	4	3	3	3.00
1000	3	3	3	4	3	4	3	3	4	3	3.30

**Graf1**

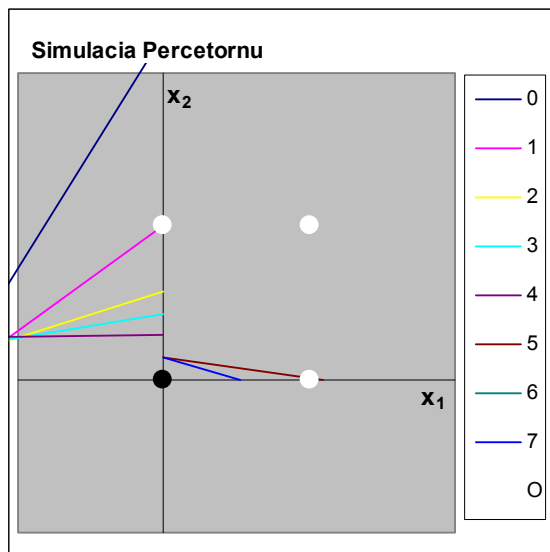


V druhom experimente som zachytil konkrétnu simuláciu učenia na jednom vstupe (alpha=0.01). **Tab2** zobrazuje pre danú epochu vstupy, požadované výstupy, odozvu, váhy a chybu. Chyba je počet nesprávne klasifikovaných výstupov. **Graf2** znázorňuje separujúcu priamku. **Graf3** zobrazuje pre danú epochu chybu odozvy (počet nesprávnych odpovedí).

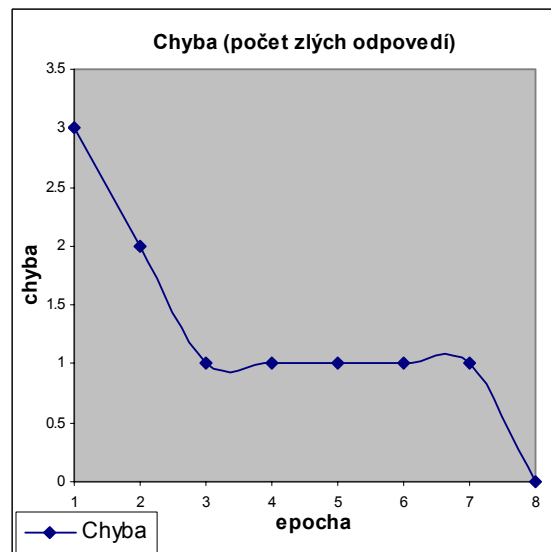
Tab2

epocha	x0	x1	x2	y	d	w0	w1	w2	chyba
1	-1	0	0	0	0	0.0899838	-0.0608448	0.0400678	3
	-1	0	1	0	1				
	-1	1	0	0	1				
	-1	1	1	0	1				
2	-1	0	0	0	0	0.0599838	-0.0408448	0.0600678	2
	-1	0	1	1	1				
	-1	1	0	0	1				
	-1	1	1	0	1				
3	-1	0	0	0	0	0.0399838	-0.0208448	0.0700677	1
	-1	0	1	1	1				
	-1	1	0	0	1				
	-1	1	1	1	1				
4	-1	0	0	0	0	0.0299838	-0.0108448	0.0700677	1
	-1	0	1	1	1				
	-1	1	0	0	1				
	-1	1	1	1	1				
5	-1	0	0	0	0	0.0199838	-0.00084475	0.0700677	1
	-1	0	1	1	1				
	-1	1	0	0	1				
	-1	1	1	1	1				
6	-1	0	0	0	0	0.00998384	0.00915525	0.0700677	1
	-1	0	1	1	1				
	-1	1	0	0	1				
	-1	1	1	1	1				
7	-1	0	0	1	0	-1.62E-05	0.0191552	0.0700677	1
	-1	0	1	1	1				
	-1	1	0	1	1				
	-1	1	1	1	1				
8	-1	0	0	0	0	0.00998384	0.0191552	0.0700677	0
	-1	0	1	1	1				
	-1	1	0	1	1				
	-1	1	1	1	1				

Graf2



Graf3



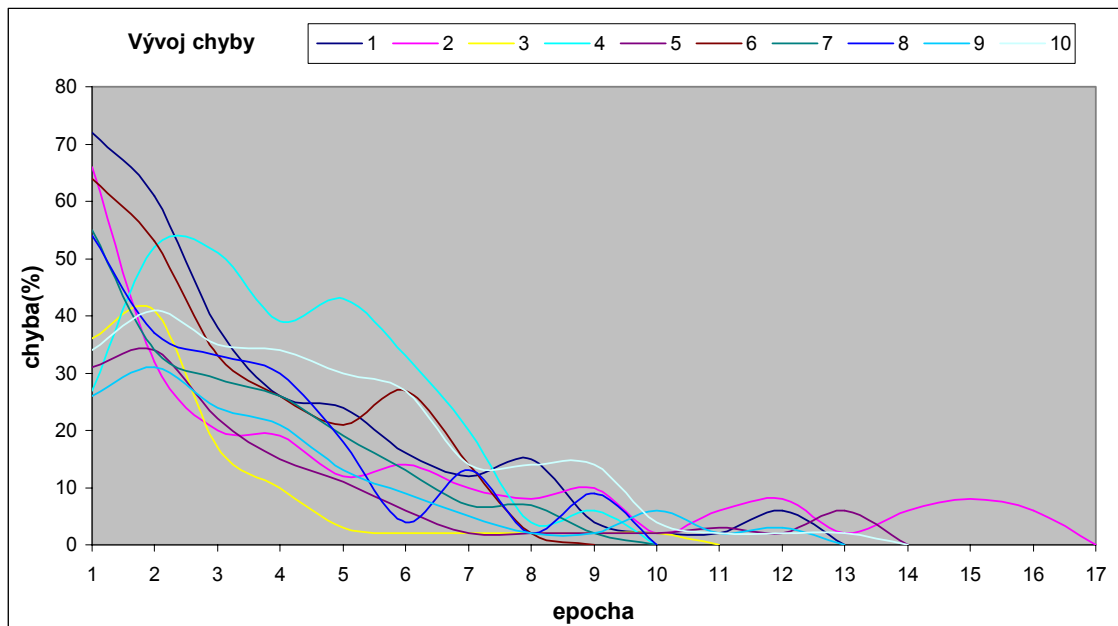
## Časť B

V tomto experimente som pre rýchlosť ( $\alpha=0.002$ ) učenia 10 krát náhodne  $(-0.1,+0.1)$  nastavil váhy perceptrónu a zmeral jeho chyby v jednotlivých epochách na trénovacej množine *p2.dat*. (obsahuje rovnakú trenovaciú množinu ako *perceptron2.dat*, zmenený je len formát vstupu – vhodný pre môj software) Údaje sú uvedené v **Tab3** a **Graf4**.

**Tab3**

epocha	Chyba (počet chybných zo 100)									
1	72	66	36	27	31	64	55	54	26	34
2	61	32	41	52	34	53	34	37	31	41
3	38	20	17	51	22	33	29	33	24	35
4	26	19	10	39	15	26	26	30	21	34
5	24	12	3	43	11	21	19	18	13	30
6	16	14	2	33	6	27	13	4	9	27
7	12	10	2	20	2	14	7	13	5	14
8	15	8	2	4	2	2	7	2	2	14
9	4	10	2	6	2	0	2	9	2	14
10	2	2	2	0	2		0	0	6	4
11	2	6	0		3				2	2
12	6	8			2				3	2
13	0	2			6				0	2
14		6			0					0
15		8								
16		6								
17		0								

**Graf4**



## Príloha – fragmenty zdrojového kódu

Program je napísaný v C++. Skompilovaný v Borland CBuilder 6 (resp. Dev-CPP4). Je organizovaný do dvoch objektov : *CMatrix* – Štandardná algebraická matica a *CNet* – model neurónovej siete. Ďalej uvádzam niekoľko metód týkajúcich sa simulácie perceptrónu.

```
// main.cpp

// activation function
void fx(CMatrix<float> &xx)
{
    for(int i=0; i<xx.m; i++) xx(i,0) = (xx(i,0) < 0) ? 0 : 1;
}

// calculate output (odozva)
void signal()
{
    for(int j=0; j<w.n; j++) { mul( w(j), x(j), x(j+1) ); fx( x(j+1) ); }
}

// reset weights
void reset()
{
    for(int j=0; j<w.n; j++) { w(j).rndset(); } // (-0.1,+0.1)
}

// train network - only one layer now
void train(int k)
{
    CMatrix<float> &w0=w(0), &x0=x(0), &x1=x(1), &o=out(k);
    x(0) = in(k);
    signal();
    for(int i=0; i<w0.m; i++)
        for(int j=0; j<w0.n; j++) w0(i,j) = w0(i,j) + alpha*(o(i,0) - x1(i,0))*x0(j,0);
}

// train network with whole trainset
void trainAll()
{
    for(int k=0; k<in.n; k++) train(k); if (echo) *fout << endl;
}

// train until 100% (returns number of epoches)
int train100()
{
    int e=0,err;
    while((err = compare()) != 0) { *fout << err << " "; trainAll(); e++; }
    *fout << err << endl << "train epoches : " << e << endl;
    return e;
}
```