Peter Csiba 09.04.2011

Neurónové siete – Multilayer Network - Backpropagation

Úvod

Úlohou bolo naprogramovať viacvrstvový perceptrón. Ako predloha nám slúžili slide-i z prednášok a zopár externých zdrojov. Využívali sme klasický Backpropagation s k-fold cross validation. Menili sme rôzne parametre sietí a hľadali sme model s minimálnou primernou chybou na validačnej množine (CV). Tento model sme potom testovali na testovacej množine a jej výsledky sme skúmali bližšie.

Algoritmus

Postupne vymenujeme najdôležitejšie časti použitého algortimu.

Úprava váh na jednom vstupe pomocu Backpropagation

Klasický Backpropagation algoritmus spočíva v tom, že máme v sieti zopár skrytých vrstiev neurónov, ktoré pracujú veľmi podobne ako Perceptrón. V prvej fáze nazývanej Forward pass si štandarným spôsobom postupne spočítame výstup na 1. , 2. , … až poslednej vrstve vrátane výstupnej.

V druhej fáze nazývanej Backward pass si vypočítame delty. Toto číslo sa snaží vyjadrovať aký podiel na chybe má daný neurón. Delta sa počíta po vrstvách od zadu, pričom na poslednej vrstve sa vypočíta špeciálne ako rozdiel očakávaného a vypočítaného výstupu.

Keď sme určili nakoľko boli jednotlivé neuróny “chybné”, aplikujeme podobné pravidlo ako pri bežnom perceptróne. V tomto projekte sme implementovali aj momentum, ktorý zohľadňuje zmenu váhy v minulom vstupe.

Testovanie siete pomocou K-cross validation

Rozdelíme trénovaciu množinu na estimačnú a validačnú v pomere K-1:1. Na estimačnej množine sieť natrénujeme v jednej epoche. Potom ju otestujeme na validačnej množine a zistíme priemernú chybu (e_val), ktorú si zapamätáme.

Na základe e_val môžeme určiť Stopping criteria – podmienku kedy so simuláciou skončíme. V tomto porjekte sme zvolili priemer každých 20 e_val, ak dva priemery za sebou stúpali, tak sme skončili. Týmto prístupom sme sa snažili odhadnúť kedy začína chyba na validačnej množine stúpať.

Hlavná myšlienka K-cross validation spočíva v tom, že vstupnú množinu rozdelíme na K rovnakých častí a K-krát nezávisle trénujeme, pričom za validačné množiny postupne volíme týchto K častí.

Hľadanie optimálneho modelu

Na začiatku si vygenerujeme parametre, z ktorých budeme jednotlivé siete vytvárať. Tieto údaje si ukladáme v CaseData:

```
public class CaseData {
	public int max_epoch_count = 1000;
	public double alpha = 0.15;
	public double mi = 0.5;
	public int k_fold = 10;
	public Vector<Integer> layer_sizes = new Vector<Integer>();
	public boolean sigmoid_tangent = false;
```

K týmto dátam postupne priradíme prislúchajúce siete a výpočtom do tejto štruktúry pridáme CV. V jednotlivých sieťach si pamätáme všetky inštancie tohto modelu použitých pri K-cross validation.

Podľa najmenšieho CV modelov určíme náš výstupný model. Tento testujeme na testovacej množine a pozorujeme výtupy jej klasifikácie.

Kritické časti programu

Activation Function

```
	public static double activationFunction(double net){
		if(Scout.sigmoid_tangent)
			return Math.tanh(net);
		else
			return 1.0 / (1.0 + Math.pow(Math.E, -net));
	}
```

Derivation of Activation Function

```
	public static double derivationOfActivationFunction(double _y){
		if(Scout.sigmoid_tangent)
			return (1+_y)*(1-_y);
		else
			return _y*(1-_y);
	}
```

Backpropagation – Forward pass (on a concrete neuron)

```
	public void computeY(Layer input_layer){
		if(this.weight == null) //Input and Threshold neurons
			return;

		double net = 0.0;
		for(int i=0; i<this.weight.size() ; i++){
			net += this.weight.get(i) *  input_layer.get(i).y;
		}

		this.y = Scout.activationFunction(net);
	}
```

Backpropagation – Backward pass (on a concrete neuron)

```
	public void computeDelta(Layer output_layer, int this_index){
		if(this.weight == null) //Input and Threshold neurons
			return;

		this.delta = 0.0;
		for(int i=0; i < output_layer.size() ; i++){
			//Threshold is not counted.
			this.delta += output_layer.get(i).delta
                                        *
output_layer.get(i).getWeightAt(this_index);
		}
		this.delta *= Scout.derivationOfActivationFunction(this.y);
	}
```

Backpropagation – Learning rule with momentum

```
public void learn(Layer input_layer, double alpha, double mi){
```

```
        if(this.weight == null) //Input and Threshold neurons
                return;

        //w_ij update
        for(int j=0 ; j < this.weight.size() ; j++){
                double delta_w = alpha * this.delta * input_layer.get(j).y;
                this.weight.set(j,
                //alpha * delta_i * input_j + momentum * last_delta_weight
                this.weight.get(j)+delta_w + mi * this.last_delta_weight.get(j)
                );
                this.last_delta_weight.set(j, delta_w);
        }
}
```

K-Cross validation

```
for(int i=0; i<k; i++){
        Supervisor estimator = supervisor.extractComplement(i, i+1, k);
        Supervisor validator = supervisor.extract(i, i+1, k);

        double e_val = this.trainNetwork(
                epoch_count,
                estimator,
                validator,
                log
        );
        //Remember this network.
        this.addNetworkToInstances(estimator, validator, e_val);
        this.resetNetwork();
        global_cv += e_val;
}
```

Výsledky

Skúšali sme rôzne hodnoty alfa (0,05;0,1;0,2;0,3), momentov (0,0;0,1;0,3;0,5) a topológie sietí (jedna alebo dve skryté vrstvy s 16 alebo 32 neurónmi). Tangentovú aktivačnú funkciu sme nebrali v úvahu, lebo dávala naoko horšie výsledky ako sigmoida. Dokopy sme vyskúšali 96 rôznych modelov, hľadanie bežalo 3-4 hodiny.

Tabuľka chýb modelov

V nasledujúcej tabuľke zobrazujeme chyby jednotlivých modelov. Cv, est[%], val[%] zodpovedajú primerným chybám inštancií modelov. Est[%], val[%] sú chyby klasifikácie v percentách na estimačnej, resp. validačnej. (Počet nominálnych chodnôt chybnej klasifikácie sme nepovažovali za tak dôležitý údaj.)

	Alpha	Moment	ActFn	LayCount	Layers	CV	est[%]	val[%]
1	0,05	0	Sigmoid	1	16	0,060275	1,622746	3,291139
2	0,05	0	Sigmoid	1	16	0,060014	1,428571	3,544304
3	0,05	0	Sigmoid	1	16	0,05984	1,525659	2,658228
4	0,05	0	Sigmoid	2	16,32	0,048799	1,636616	3,291139
5	0,05	0	Sigmoid	2	16,32	0,048363	2,122053	2,911392
6	0,05	0	Sigmoid	2	16,32	0,051169	1,830791	3,544304
7	0,05	0,1	Sigmoid	1	16	0,06424	1,595007	3,037975
8	0,05	0,1	Sigmoid	1	16	0,052494	1,400832	1,64557
9	0,05	0,1	Sigmoid	1	16	0,058996	1,414702	3,037975
10	**0,05**	**0,1**	**Sigmoid**	**2**	**16,32**	**0,042159**	**1,719834**	**2,531646**
11	0,05	0,1	Sigmoid	2	16,32	0,05205	1,775312	3,417722
12	0,05	0,1	Sigmoid	2	16,32	0,057778	2,038835	3,924051
13	0,05	0,3	Sigmoid	1	16	0,055358	1,49792	3,037975
14	0,05	0,3	Sigmoid	1	16	0,054919	1,636616	3,037975
15	0,05	0,3	Sigmoid	1	16	0,059157	1,442441	3,037975
16	0,05	0,3	Sigmoid	2	16,32	0,056737	1,886269	3,417722
17	0,05	0,3	Sigmoid	2	16,32	0,055967	2,094313	3,924051
18	0,05	0,3	Sigmoid	2	16,32	0,059968	2,385576	3,924051
19	0,05	0,5	Sigmoid	1	16	0,05916	1,525659	3,291139
20	0,05	0,5	Sigmoid	1	16	0,056749	1,608877	2,911392
21	0,05	0,5	Sigmoid	1	16	0,052221	1,608877	1,898734
22	0,05	0,5	Sigmoid	2	16,32	0,051289	2,149792	2,911392
23	0,05	0,5	Sigmoid	2	16,32	0,061451	1,983356	4,303797
24	0,05	0,5	Sigmoid	2	16,32	0,053918	2,23301	3,037975
25	0,1	0	Sigmoid	1	16	0,054976	1,595007	3,291139
26	0,1	0	Sigmoid	1	16	0,051565	1,816921	3,037975
27	0,1	0	Sigmoid	1	16	0,056952	1,567268	3,797468
28	0,1	0	Sigmoid	2	16,32	0,056681	2,482663	3,797468
29	0,1	0	Sigmoid	2	16,32	0,065468	2,857143	3,924051
30	0,1	0	Sigmoid	2	16,32	0,060496	2,773925	4,177215
31	0,1	0,1	Sigmoid	1	16	0,057142	1,719834	3,037975
32	0,1	0,1	Sigmoid	1	16	0,055178	1,983356	2,658228
33	0,1	0,1	Sigmoid	1	16	0,053142	1,442441	2,405063
34	0,1	0,1	Sigmoid	2	16,32	0,067382	2,649098	5,063291
35	0,1	0,1	Sigmoid	2	16,32	0,056063	2,496533	3,544304
36	0,1	0,1	Sigmoid	2	16,32	0,05591	1,955617	3,797468
37	0,1	0,3	Sigmoid	1	16	0,057583	1,85853	2,658228
38	0,1	0,3	Sigmoid	1	16	0,061233	1,900139	4,303797
39	0,1	0,3	Sigmoid	1	16	0,058408	1,955617	3,417722
40	0,1	0,3	Sigmoid	2	16,32	0,06623	2,59362	4,556962
41	0,1	0,3	Sigmoid	2	16,32	0,078002	2,718447	5,189873
42	0,1	0,3	Sigmoid	2	16,32	0,066458	3,037448	4,936709
43	0,1	0,5	Sigmoid	1	16	0,056433	1,47018	2,78481
44	0,1	0,5	Sigmoid	1	16	0,058064	1,705964	3,544304
45	0,1	0,5	Sigmoid	1	16	0,061131	1,803051	3,417722
46	0,1	0,5	Sigmoid	2	16,32	0,061195	2,884882	3,670886
47	0,1	0,5	Sigmoid	2	16,32	0,060108	2,760055	4,177215

48	0,1	0,5	Sigmoid	2	16,32	0,075939	3,051318	5,189873
49	0,2	0	Sigmoid	1	16	0,053974	1,775312	3,417722
50	0,2	0	Sigmoid	1	16	0,051899	1,692094	3,037975
51	0,2	0	Sigmoid	1	16	0,055203	1,85853	2,658228
52	0,2	0	Sigmoid	2	16,32	0,078706	2,524272	5,316456
53	0,2	0	Sigmoid	2	16,32	0,052851	2,829404	3,670886
54	0,2	0	Sigmoid	2	16,32	0,071512	2,662968	4,43038
55	0,2	0,1	Sigmoid	1	16	0,057066	2,191401	3,670886
56	0,2	0,1	Sigmoid	1	16	0,054555	2,011096	3,291139
57	0,2	0,1	Sigmoid	1	16	0,052999	1,830791	2,911392
58	0,2	0,1	Sigmoid	2	16,32	0,089528	3,356449	5,189873
59	0,2	0,1	Sigmoid	2	16,32	0,089403	3,384189	5,949367
60	0,2	0,1	Sigmoid	2	16,32	0,062836	2,912621	4,556962
61	0,2	0,3	Sigmoid	1	16	0,059043	2,080444	3,164557
62	0,2	0,3	Sigmoid	1	16	0,058942	2,038835	3,670886
63	0,2	0,3	Sigmoid	1	16	0,052555	1,900139	3,037975
64	0,2	0,3	Sigmoid	2	16,32	0,061031	2,454924	3,797468
65	0,2	0,3	Sigmoid	2	16,32	0,070456	3,245492	4,810127
66	0,2	0,3	Sigmoid	2	16,32	0,075078	3,495146	5,316456
67	0,2	0,5	Sigmoid	1	16	0,059346	2,260749	4,177215
68	0,2	0,5	Sigmoid	1	16	0,060721	1,678225	3,670886
69	0,2	0,5	Sigmoid	1	16	0,069893	2,246879	4,303797
70	0,2	0,5	Sigmoid	2	16,32	0,078663	3,619972	5,56962
71	0,2	0,5	Sigmoid	2	16,32	0,053076	3,065187	3,797468
72	0,2	0,5	Sigmoid	2	16,32	0,063352	2,649098	4,43038
73	0,3	0	Sigmoid	1	16	0,060665	2,094313	3,544304
74	0,3	0	Sigmoid	1	16	0,052655	1,886269	3,291139
75	0,3	0	Sigmoid	1	16	0,062673	2,011096	4,303797
76	0,3	0	Sigmoid	2	16,32	0,066545	3,522885	3,924051
77	0,3	0	Sigmoid	2	16,32	0,062164	3,31484	3,924051
78	0,3	0	Sigmoid	2	16,32	0,070211	3,051318	4,303797
79	0,3	0,1	Sigmoid	1	16	0,053047	2,23301	3,037975
80	0,3	0,1	Sigmoid	1	16	0,054021	2,302358	3,037975
81	0,3	0,1	Sigmoid	1	16	0,067848	2,427184	4,303797
82	0,3	0,1	Sigmoid	2	16,32	0,070542	2,815534	4,810127
83	0,3	0,1	Sigmoid	2	16,32	0,097135	4,479889	6,962025
84	0,3	0,1	Sigmoid	2	16,32	0,074302	3,467406	5,189873
85	0,3	0,3	Sigmoid	1	16	0,06136	2,21914	3,670886
86	0,3	0,3	Sigmoid	1	16	0,0597	2,413315	3,291139
87	0,3	0,3	Sigmoid	1	16	0,064967	1,969487	4,683544
88	0,3	0,3	Sigmoid	2	16,32	0,08764	3,439667	6,329114
89	0,3	0,3	Sigmoid	2	16,32	0,095749	3,509015	5,949367
90	0,3	0,3	Sigmoid	2	16,32	0,057962	2,787795	3,417722
91	0,3	0,5	Sigmoid	1	16	0,061026	2,801664	3,291139
92	0,3	0,5	Sigmoid	1	16	0,049538	2,21914	3,291139
93	0,3	0,5	Sigmoid	1	16	0,058837	2,288488	3,797468
94	0,3	0,5	Sigmoid	2	16,32	0,103602	4,521498	6,329114
95	0,3	0,5	Sigmoid	2	16,32	0,069793	3,536755	4,556962
96	0,3	0,5	Sigmoid	2	16,32	0,076208	2,912621	6,075949

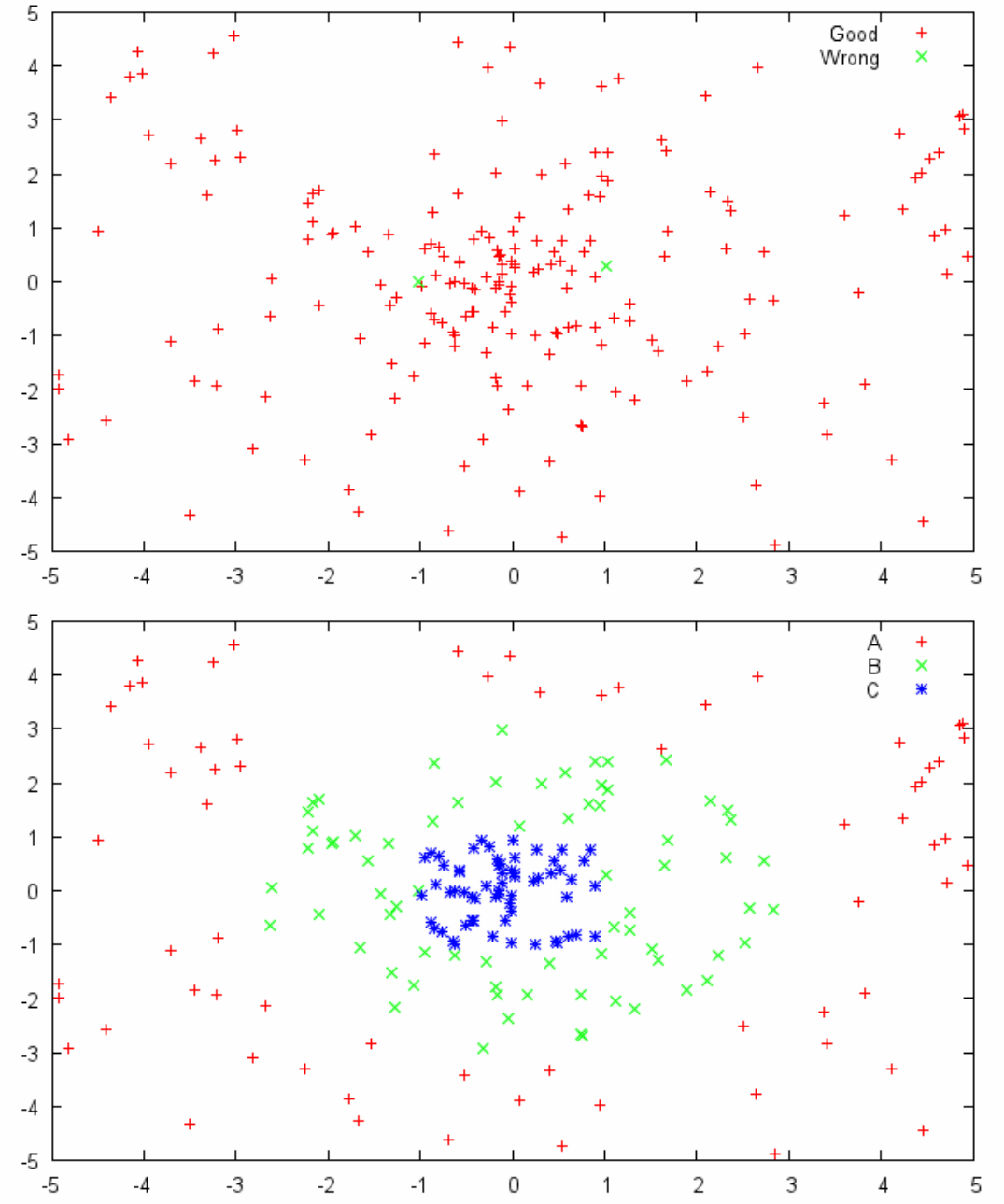
Confussion matrix najlepšieho modelu

V riadkoch sú požadované výsledky a v stĺpcoch vypočítané výsledky na najlepšom modeli.

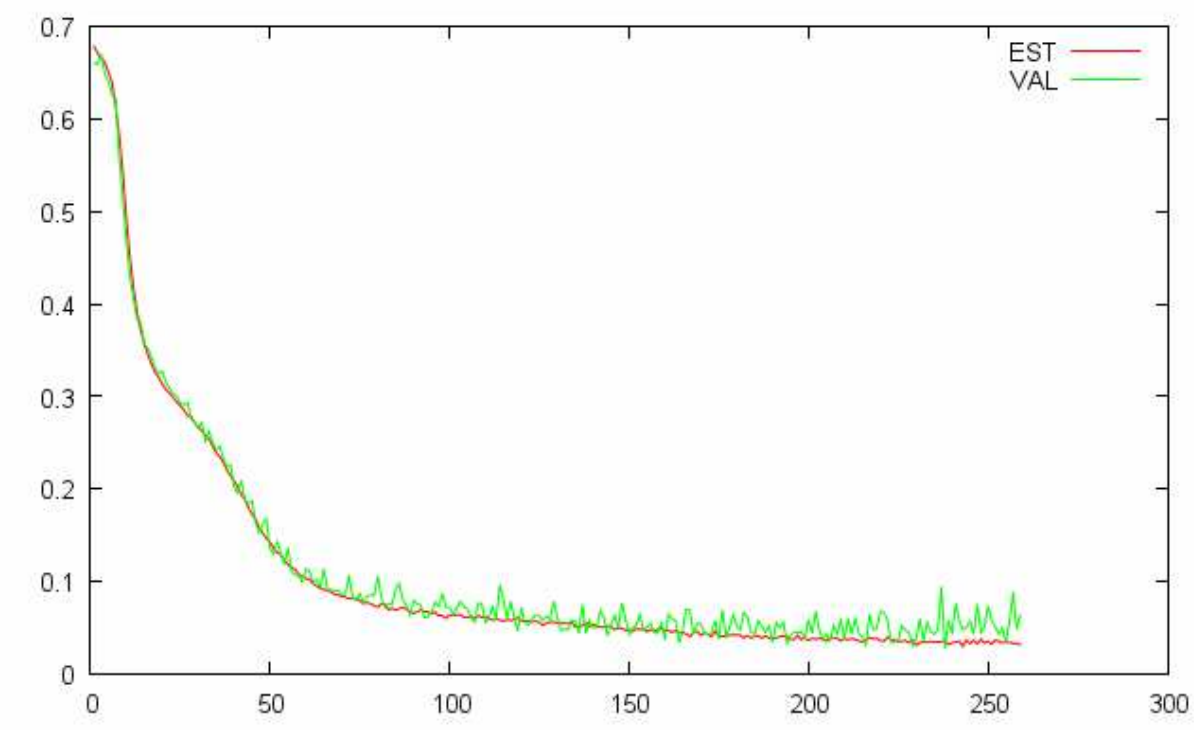
	A	B	C
A	67	0	0
B	0	69	2
C	0	0	62

2D znázornenie chyby klasifikácie najlepšieho modelu na testovacách dátach

Vstupom boli dvojrozmerné dáta, ktoré sme mali klasifikovať. Na prvom obrázku sú znázornené zle klasifikované body zelenou. Na druhom obrázku sú znázornené jednotlivé triedy podľa ktorých sme mali klasifikovať.



Vývoj estimačnej a validačnej chyby najlepšieho modelu počas epôch
Po 260 epochách pri tomto rozdelení na validačnú a estimačnú množinu boli splnené stopping criteria. Vidieť, že priemerná chyba na klasifikačnej množine začala mierne stúpať. Takisto vidno, že oscilácia tejto chyby sa zvyšuje počtom epôch.



Dodatky

Global.log
Chyby na jednotlivých modeloch a ich inštanciách. Tento súbor popisuje všetky vyskúšané modely a priemerné chyby jednotlivých inštancií na validačných množinách a percentuálne chyby klasifikácie na estimačnej a validačnej množine.
Jeden model je popísaný na 11tich riadkoch. Prvých desať sú CV, est[%] a val[%] na jednotlivých inštanciách modelu. Jedenásty riadok popisuje parametre použité na výrobu tohoto modelu ako sedmice (Alfa, Moment, ActFn, LayerCount, LayerSizes, CV). Tieto demice sú zapísané v tabuľke „Tabuľka chýb modelov“.

Best_classification.log
„Taktiež znázornite fungovanie modelu (jednu inštanciu) na testovacej množine ako výstupy modelu v 2D priestore so znázornením príslušnosti klasifikácie do jedno tlivých
tried“
Na každom riadku je štvorica (x, y, target_class, output_class).

Zdrojové kódy
Projekt bol programovaný v Jave. Kolega programoval to isté veľmi podobným spôsobom v C++ a program mu bežal 5-krát rýchlejšie. Preto odporúčam do budúcnosti podobne výpočotovo náročné porjekty robiť v jazykoch, ktoré sa kompilujú to strojového kódu.

Referencie

http://ii.fmph.uniba.sk/~farkas/Courses/NeuralNets/Slides/mlp.4x.pdf
http://ii.fmph.uniba.sk/~farkas/Courses/NeuralNets/Slides/mlp2.4x.pdf
http://page.mi.fu-berlin.de/rojas/neural/chapter/K7.pdf
Neural Networks – A Comprehensiveb Foundation – Simon Haykin
http://en.wikiversity.org/wiki/Learning_and_Neural_Networks
http://galaxy.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Použitý software
Eclipse
Java Virtual Machine
Gnuplot
Microsoft Excel
Notepad2