TECHNICKÁ UNIVERZITA LIBEREC

PEDAGOGICKÁ FAKULTA

Katedra: Informatiky

Kombinace: Fyzika – Informatika

Paralelní neuronová síť

a její využití v pedagogice

Diplomová práce: 98-PF-KIN-03

Autor:

Radek Holeček

podpis: .....

Adresa:

Ke Kamenci 72

281 23 Starý Kolín

Vedoucí práce: Ing. Jaromír Kukal

Počet:

stran	obrázků	tabulek	příloh
56	12	6	3

V Liberci, dne 27. dubna 1998

· Technická univerzita v Liberci
PEDAGOGICKÁ FAKULTA
461 17 LIBEREC 1, Hálkova 6 Telefon: 329 Telefax: 21301

Katedra: informatiky

ZADÁNÍ DIPLOMOVÉ PRÁCE

(závěrečného projektu)

diplomant: Radek H O L E Č E K
adresa: Ke Kamenci 72, Starý Kolín, 281 23

obor: Fyzika - Informatika

Název: Paralelní neuronová síť a její využití v pedagogice

Vedoucí práce: Ing. Jaromír Kukal

Termín odevzdání: 26.5. 1998

Pozn. Podmínky pro zadání práce jsou k nahlédnutí na katedrách. Katedry rovněž specifikují zadání: východiska, cíle, předpoklady, metody zpracování, základní literaturu (zpravidla na rub tohoto formuláře). Zásady pro zpracování DP jsou k dispozici ve dvou verzích (stručné, resp. metodické pokyny) v UK TUL, na katedrách a na Děkanátě Pedagogické fakulty.

V Liberci dne 25.4. 19 97

prof.Ing. Jan Ehleman, CSc.
vedoucí katedry

doc.RNDr. Jaroslav Vild
děkan

Převzal (diplomant):
Datum: 30.4.1997 Podpis: Holeček

KIT/IN 56 s.
TECHNICKÁ UNIVERZITA V LIBERCI
Univerzitní knihovna
Voroněžská 1329, Liberec 1
V 138/98 P

Prohlášení a poděkování

Prohlášení o původnosti práce:

Prohlašuji, že jsem diplomovou práci vypracoval samostatně a že jsem uvedl veškerou použitou literaturu.

V Liberci, dne 27. dubna 1998 Podpis: .....

Prohlášení o využívání výsledků DP:

Jsem si vědom těchto skutečností:

- diplomová práce je majetkem školy
- s diplomovou prací nelze bez svolení školy disponovat
- diplomová práce může být zapůjčena či objednána (kopie) za účelem využití jejího obsahu

Beru na vědomí, že po pěti letech si mohu diplomovou práci vyžádat v Univerzitní knihovně Technické univerzity v Liberci, kde bude uložena.

Jméno a příjmení: Radek Holeček

Adresa: Ke Kamenci 72, 281 23 Starý Kolín

Podpis: .....

Poděkování:

Děkuji všem, kteří se zasloužili o zdárné vypracování této diplomové práce. Děkuji Ing. Jaromíru Kukalovi za námět, konzultace a vedení diplomové práce. Dále děkuji i ostatním pedagogům TU za to, co mě naučili. Děkuji své rodině, současné i budoucí, za trpělivost, kterou se mnou museli při psaní diplomové práce mít.

Anotace

Diplomová práce se zabývá učením vícevrstvých neuronových sítí, naprogramováním strategie učení a aplikacemi neuronových sítí na různých úlohách. Je použita metoda zpětného šíření. Toto je aplikováno na příklady z matematiky, zeměpisu a na problém vyhodnocení úspěšnosti studentů. Jako programovací jazyk byl použit Borland Pascal 7.0.

Summary

Diploma word subject is learning of multilayer neural net, programming the learning strategies and the application of neural net to various tasks. Back propagation method is used. The methodology is applied to math, geographic and student profile problems. Programming language is Borland Pascal 7.0.

Анотациа

Дипломная работа занимается учебой невроной сети из несколька пластов, есть здесь тоже программа для учебы и несколько наглядных примеров. Здесь употреблен метод обратново распространия. Этот метод применян к математике, географии и к проблеме успеха студентов. Программа написана в Борланд Паскал 7.0.

Technické informace

Editor: Microsoft Word 7.0

Grafika: Paint Shop Pro 5 Beta

Programovací jazyk: Borland Pascal 7.0

Operační systémy: Microsoft Windows 95, Microsoft Windows NT 4.0

Počítače: různé klony IBM PC

Licence na zde jmenované produkty firem Microsoft a Borland vlastní Technická univerzita v Liberci.

V diplomové práci se mohou vyskytovat názvy produktů, firem apod., které jsou ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Obsah

1. ÚVOD DO NEURONOVÝCH SÍTÍ 3

1.1 Několik slov úvodem 3
1.1.1 Přírodní vzor 4
1.1.2 Mozek versus počítač 5

1.2 Historie umělých neuronových sítí 6

1.3 Příklady úspěšné aplikace umělých neuronových sítí 8
1.3.1 Rozpoznávání znaků 8
1.3.2 Rezervace letenek 9
1.3.3 Obchodování na burze 9

1.4 Výhody a nevýhody umělých neuronových sítí 10
1.4.1 Několik nevýhod umělých neuronových sítí 10
1.4.2 Hlavní výhody umělých neuronových sítí 10

2. MATEMATICKÉ MODELY NEURONOVÝCH SÍTÍ 11

2.1 Základní stavební prvek umělých neuronových sítí 11
2.1.1 Schopnosti osamoceného neuronu 14

2.2 Vzájemné spojování neuronů 16

2.3 Učení neuronových sítí 18
2.3.1 Trénovací vzory 18
2.3.2 Proces učení jednoho neuronu 19
2.3.3 Učení vícevrstvých sítí 21

2.4 Implementace neuronové sítě v jazyce Pascal 22

2.5 Popis jednotky Sit 23

3. PRAKTICKÉ PŘÍKLADY ŘEŠENÉ POMOCÍ NEURONOVÝCH SÍTÍ 26

3.1 Problém prvočísel 26

3.2 Vlajky států 30

3.3 Výsledky na střední škole a jejich vliv na přijetí na školu vysokou 33
3.3.1 Určení pohlaví podle výsledků na střední škole 34
3.3.2 Jak poznat, vyšly-li studentovi plány 36

4. ZÁVĚR 39

PŘÍLOHA A: JEDNOTKA SIT 40

PŘÍLOHA B: PŘÍKLAD POUŽITÍ JEDNOTKY SIT 49

PŘÍLOHA C: TRÉNOVACÍ VZORY 50

LITERATURA 54

REJSTŘÍK 55

Seznam obrázků:

OBR. 1: SCHEMATICKÉ ZNÁZORNĚNÍ BIOLOGICKÉHO NEURONU. 4
OBR. 2: SCHEMATICKÉ ZNÁZORNĚNÍ FORMÁLNÍHO NEURONU. 11
OBR. 3: PŘEVODNÍ FUNKCE - OSTRÁ NELINEARITA. 12
OBR. 4: PŘEVODNÍ FUNKCE - SATUROVANÁ LINEÁRNÍ FUNKCE. 13
OBR. 5: PŘEVODNÍ FUNKCE - SIGMOIDA. 13
OBR. 6: PŘEVODNÍ FUNKCE - HYPERBOLICKÝ TANGENS. 13
OBR. 7: LINEÁRNÍ SEPARACE TRÉNOVACÍCH VZORŮ LOGICKÝCH FUNKCÍ AND A OR. 15
OBR. 8: UKÁZKA FUNKCE, KTEROU NELZE LINEÁRNĚ SEPAROVAT. 15
OBR. 9: PŘÍKLAD CYKLICKÉ NEURONOVÉ SÍTĚ. 17
OBR. 10: UKÁZKA VRSTEVNÉ SÍTĚ. 17
OBR. 11: KONFIGURACE NEJJEDNODUŠŠÍ SÍTĚ ŘEŠÍCÍ PROBLÉM PRVOČÍSEL. 29
OBR. 12: KONFIGURACE NEJJEDNODUŠŠÍ SÍTĚ ŘEŠÍCÍ PŘÍSLUŠNOST K EU. 32

Seznam tabulek:

TABULKA 1: TRÉNOVACÍ VZORY LOGICKÝCH FUNKCÍ 19
TABULKA 2: TRÉNOVACÍ VZORY PRVOČÍSEL 26
TABULKA 3: UKÁZKA ČASOVÉ NÁROČNOSTI ŘEŠENÍ NS 27
TABULKA 4: SEPARACE VZORŮ PRVOČÍSEL 28
TABULKA 5: VÁHY NEURONOVÉ SÍTĚ ŘEŠÍCÍ PROBLÉM PRVOČÍSEL 29
TABULKA 6: TRÉNOVACÍ VZORY VLAJEK STÁTŮ 31

1. Úvod do neuronových sítí

1.1 Několik slov úvodem

Odpradávna se člověk snaží sám sobě ulehčit práci tím, že používá různé nástroje. Zpočátku to byly pouze kusy větví nebo nějaký kámen. Postupem času ale člověk, jakožto myslící bytost, přišel na to, že by potřeboval i takové nástroje, které se v přírodě nevyskytují nebo se vyskytují jen vzácně. A tak se stalo, že si to, co našel v přírodě, začal upravovat tak, aby se užitná hodnota nástroje zvětšila. A tak to probíhá dodnes.

Člověk i dnes využívá přírodu jako zdroj materiálu, z něhož pak tvoří své „nástroje". Postupem času se lidstvo dopracovalo k nástrojům, které už pracovaly více samy, než za pomoci člověka. Těm se začalo říkat stroje.

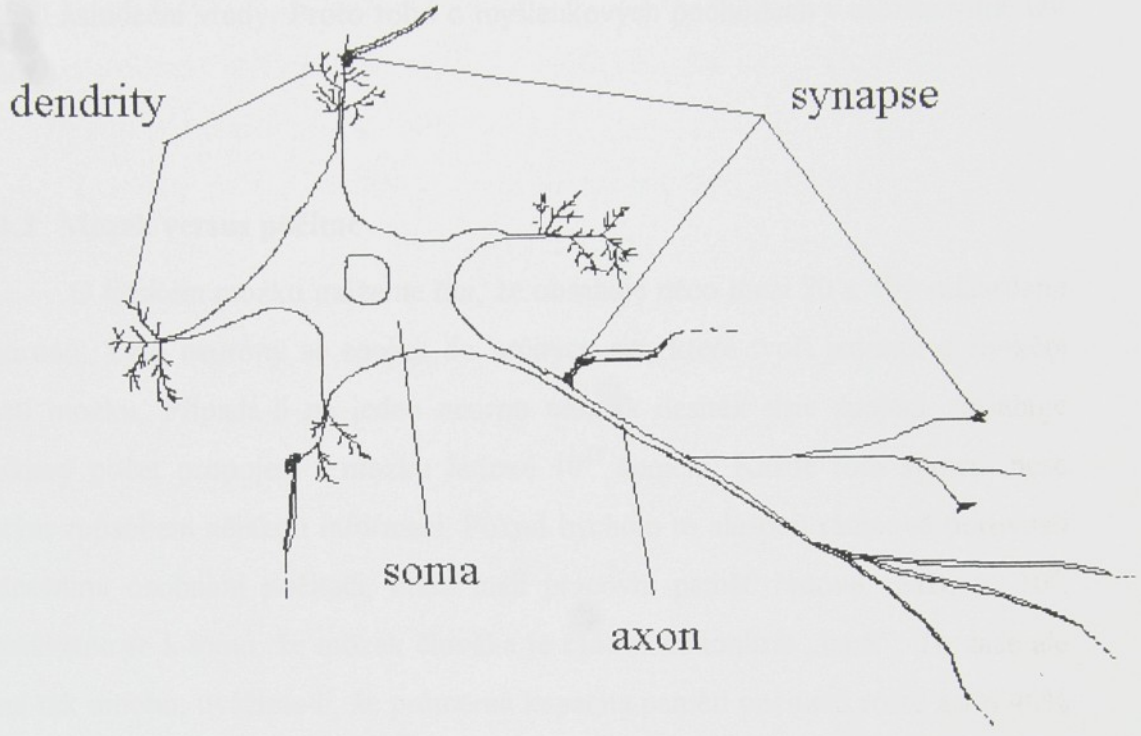
Stroje byly čím dál tím složitější, ale pořád to byla jen jednoúčelová zařízení, která dělala pořád totéž. A tak člověk přemýšlel, co s tím, a vymyslel počítač. Ten už dělal to, co mu bylo přikázáno, i když zpočátku probíhala komunikace mezi člověkem a počítačem dosti krkolomně za pomoci jistých umělých jazyků, kterými se sice dalo vyjádřit hodně, ale ne všechno. Tyto jazyky byly založeny na matematických zákonitostech, ale ukázalo se, že všechno, co se člověk snaží říci nebo udělat nemá jednoduché matematické vyjádření.

Člověk se tedy začal zabývat tím, jak to zařídit, aby se počítače učily samy. Aby jim stačilo ukázat nějaký problém a ony se ho postupně naučily řešit. V podstatě chtěl člověk na počítačích, aby se učily samy, stejně tak, jako to dělají lidé a ostatně i všechna zvířata.

Na otázku jak to udělat si člověk snažil odpovědět tím, že asi stejně, jak to funguje v přírodě. A tak se začalo zkoumat, jak proces učení probíhá v živých organismech.

1.1.1 Přírodní vzor

A tak se postupem času lidstvo dobralo k jakémusi základnímu kameni, z něhož se skládá mozek, jež je tím orgánem, který má na svědomí učení a s ním související myšlení. Tím základním kamenem je buňka, které se říká neuron.



Obr. 1: Schematické znázornění biologického neuronu.

Nákres jednoho neuronu je na Obr. 1, kde jsou vyznačeny i části, z nichž se skládá. V každém případě má neuron tělo, kterému se říká soma. Z něho vychází jediný výstup (axon), který se ale může dále větvit. Vstup informací od okolních neuronů se většinou děje přes tzv. dendrity, kterých může mít jeden jediný neuron i několik desítek tisíc. Velikost těla neuronu bývá v rozmezí od několika málo μm až do několika desítek μm. Délka vstupních dendritů bývá maximálně 2 až 3 mm. Délka výstupního axonu může být i několiksetkrát větší. U člověka jsou některé axony dokonce delší než 1m. Mluvíme-li o neuronech, často se též setkáváme s pojmem synapse. Za ním se skrývá pojmenování pro funkční část, která zprostředkovává kontakt axonu s dendrity jiných neuronů.

Říkáme-li o neuronech, že jsou základními kameny, v žádném případě nám to nic neříká o vnitřní složitosti jednoho neuronu. Neuron je nesmírně složitá buňka a do dnešního dne ji vědci ještě zcela nepochopili. To se ostatně dá o mozku říci

také. Chceme-li zkoumat myšlenkové pochody nějakého tvora, nemůžeme ho zabít a důkladně prozkoumat, protože po smrti již myšlenkové pochody neprobíhají. A tak se zkoumání omezuje pouze na vnější projevy, ale to je jako kdyby lékař místo samotného pacienta vyšetřoval jeho stín. Pozná, když mu chybí noha, ale těžko odhalí žaludeční vředy. Proto toho o myšlenkových pochodech v mozku víme tak málo.

1.1.2 Mozek versus počítač

O lidském mozku můžeme říci, že obsahuje něco mezi 20 a 100 miliardami neuronů. Tyto neurony se spojují do určitých sítí, které tvoří jednotlivé funkční části mozku. Připadá-li na jeden neuron několik desítek tisíc spojení, dosahuje celkový počet propojení v mozku řádově 10^{15} spojení. Každé toto spojení nese jistým způsobem nějakou informaci. Pokud bychom to alespoň rámcově porovnali s dnešními osobními počítači, které mají pracovní paměť řádově MB, tj. 10^9, dostáváme se k tomu, že mozek člověka je řádově milionkrát „lepší". To zase ale není tak mnoho, uvážíme-li, že průměrná kapacita paměti počítačů roste asi o 40% ročně. Na podobnou kapacitu jako má lidský mozek by se tedy běžné počítače měly dostat kolem roku 2040.

Jenže schopnost uchovávat informace je jen jeden z předpokladů myšlení. Ostatně lidská paměť není zrovna nejlepší, zato v té počítačové se uchovávají informace zcela přesně. Vypadá to, že daleko důležitější, než je paměť, je schopnost tuto paměť využívat. A v této oblasti se poměr výkonu mozku a počítače nedá srovnávat vůbec. Počítač pracuje podle programu a z dat, která zpracovává, dostane jeden výsledek. Mozek při zpracování informací nachází spoustu řešení s různou pravděpodobností a z nich teprve vybere to nejlepší či spíše to nejpravděpodobnější řešení. Dalším velkým rozdílem je přirozený paralelismus mozku. Rychlost zpracování informace je u neuronu velmi malá v porovnání s rychlostí procesoru moderních počítačů, ale všech několik desítek miliard neuronů pracuje současně a jen si předávají výsledky své práce.

Tento poslední rozdíl je také důvodem toho, že jednoduché příklady řeší počítače daleko rychleji než mozek. Například vyřešení jednoduché kvadratické

rovnice trvá počítači několik desítek mikrosekund, člověk ji řeší několik desítek sekund a to ještě ne vždy se správným výsledkem.

Složitější problém, jako je třeba rozpoznávání obrazů, ještě počítače dokáží vyřešit, ale v tomto případě už o hodně pomaleji a ne vždy se správným výsledkem. Zato člověk, který toto potřebuje ke svému životu a používá rozpoznávání obrazu v podstatě od narození, dokáže obraz zpracovat za méně než jednu sekundu a jeho úspěšnost se pro známé objekty blíží 100%.

A potom existují problémy, které už počítač řešit nedokáže, ale člověk si s nimi snadno poradí. Jak by měl třeba počítač poznat, jestli napsaný text je vtipný nebo ne? To nedokáže, ale člověku to většinou nečiní žádné potíže.

Protože jsou klasické počítače na řešení některých problémů nevhodné, začalo se uvažovat nad tím, jak sestrojit zařízení, jež by napodobovalo myšlení biologických organismů.

1.2 Historie umělých neuronových sítí

Historie umělých neuronových sítí se začala psát v roce 1921, kdy Warren McCulloch vyzdvihl velmi jasnozřivě budoucí význam neuronových sítí. Teprve však v roce 1943 uveřejnil spolu se svým šestnáctiletým studentem Walterem Pittsem v časopise *Bulletin of Mathematical Biophysics* článek *A logical calculus of the ideas immanent in nervous activity*. V něm navrhli jednoduchý matematický model neuronu. Ukázali, že jednoduchá neuronová síť je schopná počítat libovolnou aritmetickou či logickou funkci.

V roce 1948 vydal Norbert Wiener svou známou knihu o kybernetice a Shannon a Weaver knihu o informatice. Obě byly inspirovány zmíněným článkem McCullocha a Pittse, stejně jako o několik let později vydaný článek známého autora projektů elektronických počítačů Johna von Neumanna o spolehlivém provádění výpočtů pomocí ne zcela spolehlivých prvků.

Donald Hebb vydal svou knihu *The Organization of Behavior* v roce 1949. V té se snažil vztáhnout jisté vlastnosti zjištěné v psychologii na matematické modely neuronových sítí. Postupem času se ale ukazovalo, že ač se problému

umělých neuronových sítí věnuje hodně prostoru v médiích, výsledky zatím moc radosti nepřinesly.

Až v roce 1958 Frank Rosenblatt, pracující na bostonské MIT, publikoval svoji práci *The Perceptrons* v které navrhl model perceptronu, který byl určitým zobecněním McCullochova a Pittsova neuronu[1]. Tento model už nepracoval s množinou hodnot $\{-1,0,1\}$, ale používal jako parametry reálná čísla. Pro tento model navrhl Rosenblatt i algoritmus učení, který vždy vedl k řešení, pokud tedy řešení existovalo.

Po tomto průlomu se začaly objevovat na trhu první neuropočítače, které už dokázaly řešit jednoduché úlohy (např. rozpoznávání znaků). V roce 1962 formuloval Ital Caianiello tzv. integrační neuron a o tři roky později navrhl Cowan použití tzv. sigmoidní funkce v matematických modelech neuronů.

Na počátku 60. let se zdálo, že úspěšnému rozmachu umělých neuronových sítí nestojí nic v cestě, ale postupem času se ukazovalo, že přes jisté počáteční úspěchy mají umělé neuronové sítě i jistá omezení (z dnešního pohledu se zdá, že omezení tenkrát nebyla ani tak v omezenosti sítí řešit nějaké problémy, ale spíše v tom, že nebyly používány sítě odpovídající velikosti).

Koncem této etapy vývoje se stala práce Marvina Minského a Seymora Paperta *Perceptrons*, ve které připomněli známý fakt, že jeden neuron se není schopen naučit ani jednoduchou funkci exclusive-or (XOR) a z toho zcela nesprávně vyvodili, že totéž platí i pro celé sítě. Toto tvrzení bylo všeobecně považováno za matematicky dokázané a tak nastala dlouho trvající etapa vývoje (přibližně mezi lety 1967 a 1982), kdy se výzkumné týmy zaměřovaly na jiné oblasti umělé inteligence. Tento fakt byl podpořen masovým nástupem mikroprocesorů v 70. letech.

Je zajímavé, že po celou dobu útlumu prosperovala ve Spojených státech firma Adaptronics, která vyráběla specializované neuropočítače.

Počátkem 80. let se stále více ukazovalo, že neuronové sítě zase nejsou tak špatné. I přes výrazný útlum se časem nashromáždily poznatky, které vedly k obnovení zájmu o tuto problematiku. Postupem času se problematice umělých

---

[1] Zpočátku nebylo ustáleno názvosloví, pojmy neuron a perceptron, tak jak je chápeme dnes, jsou vysvětleny na str. 12-14.

neuronových sítí začala věnovat velká pozornost. Koncem 80. let bylo nadšení takové, že jistými firmami bylo předvídáno, že podíl neuropočítačů na trhu vzroste do roku 2000 na 10%.

Od té doby se situace uklidnila. Na neuronových sítích sice pracuje dost vědců, metody učení těchto sítí byly od 60. let značně vylepšeny, existuje spousta praktických aplikací, ale zatím se nezdá, že by v blízké budoucnosti byly klasické počítače vytlačeny.

1.3 Příklady úspěšné aplikace umělých neuronových sítí

1.3.1 Rozpoznávání znaků

Jednou z nejstarších aplikací neuronových sítí je rozpoznávání znaků. Máme například nasnímaný text, ať už tištěný nebo psaný rukou. Klasický počítač nejprve oddělí od sebe jednotlivé znaky textu. Každý znak se převede na matici např. 16x16 bodů. Jednotlivé body v matici jsou buď černé nebo bílé. Tato matice je vstupem neuronové sítě. Jejím výstupem je např. ASCII kód znaku.

Nejprve musí proběhnout učící fáze. Umělé neuronové síti se předkládají vždy matice a očekávaný výstup. Tím se síť učí. Když je síť naučená, dostává na vstupu obraz znaku a ona na výstupu předává jeho ASCII hodnotu. Úspěšnost takovéhoto učení dosahuje až 98%. Závisí hlavně na tom, jak kvalitně byl původní text vytištěn. U tištěných předloh bývá úspěšnost větší než u ručně psaných. Uvedená metoda se dá ještě vylepšit tím, že pokud se síť splete, dáme jí správnou odpověď, aby se mohla přiučit.

Další možnost, jak zlepšit úspěšnost, je dána tím, že síť má tolik výstupů jako je znaků a vrací nám u každého znaku pravděpodobnost[2] toho, že se jedná o ten který znak. Program ale má též k dispozici databázi slov, tedy pokud slovo po doplnění nejpravděpodobnějšího písmena není ve slovníku, pokusí se tam najít

[2] Nejedná se o pravděpodobnost v matematickém slova smyslu, protože součet pravděpodobností jednotlivých znaků by nám musel dát 1.

slovo s druhým nejpravděpodobnějším písmenem. V tomto případě se dokonce může stát, že výstup z neuronové sítě odstraní překlep z předlohy.

1.3.2 Rezervace letenek

Ve Spojených státech je o trochu jiná situace, co se týče letecké dopravy, než u nás. Vzhledem k rozlehlosti kontinentu je letecká doprava běžným typem dopravy stejně jako třeba autobusová doprava. A lidé k ní také tak přistupují. Když už to tedy musí být, tak si telefonicky zamluví letenku, ale pokud si to na poslední chvíli rozmyslí, nic s tím nedělají. Letecké společnosti tohle vědí, takže s klidným svědomím rezervují více letenek, než kolik je v letadle míst. V některých případech je toto nadhodnocení až několik desítek procent. Dělají to proto, že chtějí využít každé místo, protože z toho mají zisk. Problém ale nastane, pokud si lidé opravdu pro letenku přijdou ve větším množství, než je kapacita letadla. V takovém případě nabízejí tomu, kdo je ochoten neletět (většinou nic drastického, za pár hodin letí další letadlo), tzv. volnou letenku kamkoliv po USA. Tu může pak člověk využít pokud je letadlo zase naopak nevytíženo na 100%.

Většinou se sice najde někdo, kdo je ochoten neletět, ale pro leteckou společnost je to nepříjemné. Proto se snaží co nejlépe odhadnout chování lidí. A právě v tomto okamžiku se může ke slovu dostat neuronová síť, která podle toho, kdo si zatím objednal letenky, v kolik hodin se má let uskutečnit a ještě podle spousty dalších okolností rozhodne, jestli může letecká společnost rezervovat další letenku nebo ne.

Učení této neuronové sítě probíhá tak, že se jí dají všechna známá data z minulosti a ona se je snaží nějakým způsobem zobecnit.

1.3.3 Obchodování na burze

Chování burzy patří k jedné ze spousty věcí, které se zatím nepodařilo popsat algoritmicky a pravděpodobně se to ani nikdy nepodaří. Přesto je na burze spousta hráčů, kteří dosahují nadprůměrného výnosu i několik desítek let. Jak to dělají? To nikdo neví, protože kdyby to věděl, dělal by to tak každý a okamžitě by to přestalo fungovat. Jde o to, nalézt v dostupných informacích nějakou souvislost

a tu pak využít ke svému prospěchu. Žádný hráč na burze nemá 100% úspěšnost. Většina dobrých se pohybuje někde kolem 55%. Jsou o kapku lepší než je průměr, ale to jim stačí k tomu, aby trvale prosperovali.

O to samé se může pokoušet neuronová síť, která byla naučena na minulém vývoji cen a z nich se snaží vypočítat budoucí pohyb cen. Nekladou se na ní nijak velké nároky. Stačí, když bude lepší než těch 50%.

Jestli to funguje, to se nikdo nedoví. Pokud ano, tak to autor používá ku svému prospěchu, pokud ne, tak se přeci nebude chlubit svými neúspěchy. Pravděpodobně to však krátkodobě funguje.

1.4 Výhody a nevýhody umělých neuronových sítí

1.4.1 Několik nevýhod umělých neuronových sítí

- Dlouhá doba učení pro složitější problémy.
- Nutnost velkého počtu trénovacích vzorů.
- Pomalost při řešení problémů, které jdou snadno algoritmizovat. Tyto problémy se vždy budou řešit lépe klasickou cestou.

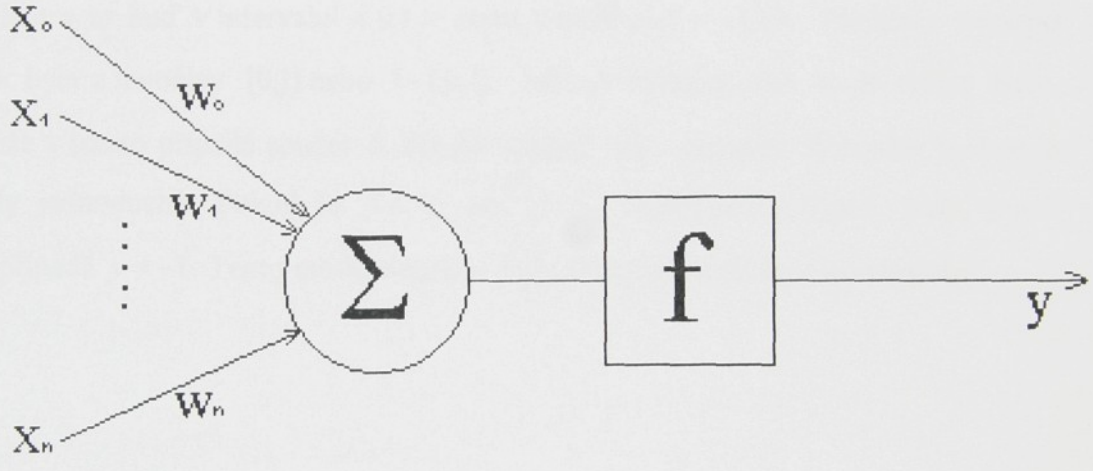
1.4.2 Hlavní výhody umělých neuronových sítí

- Schopnost zobecňovat na základě příkladů.
- Není potřeba vymýšlet algoritmus postupu.
- Při vhodné implementaci mohou všechny neurony pracovat najednou, tedy nesrovnatelně rychleji než klasický počítač.
- Univerzálnost - co je síti předloženo to zpracuje.

2. Matematické modely neuronových sítí

2.1 Základní stavební prvek umělých neuronových sítí

Základem biologických neuronových sítí je neuron. Základem uměle vytvořených neuronových sítí je matematický model neuronu, který se svými vlastnostmi podobá, někdy dost vzdáleně, své biologické předloze. Tento model se nazývá formální neuron. Samozřejmě existuje spousta matematických modelů neuronů, všechny však vycházejí ze stejného schématu (viz Obr. 2).



Obr. 2: Schematické znázornění formálního neuronu.

Dendrity jsou v matematických modelech nahrazeny vstupy x_i. Každý tento vstup má vlastní váhu w_i, která určuje významnost tohoto vstupu. Vstup x_0 je výjimečný v tom, že na něj je jako hodnota přivedena 1. Má význam jako práh neuronu, což zabezpečuje to, že neuron nebude reagovat na podněty, které leží pod tímto prahem. Všechny vstupy jsou zpracovávány tzv. sumátorem, jehož výstupem je součet vah jednotlivých vstupů s jejich hodnotami podle vztahu:

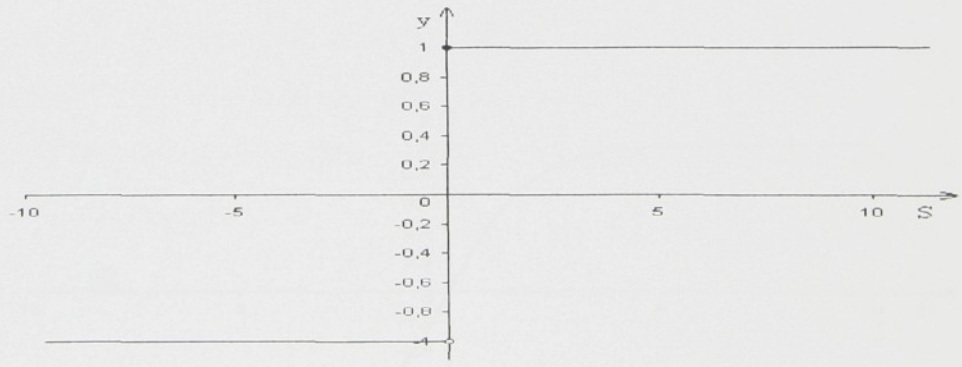
$$S = \sum_{i=0}^{n} x_i \cdot w_i \qquad (1)$$

Takto získaný součet je převeden podle tzv. převodní funkce na výstup (v biologické předloze axon) neuronu.

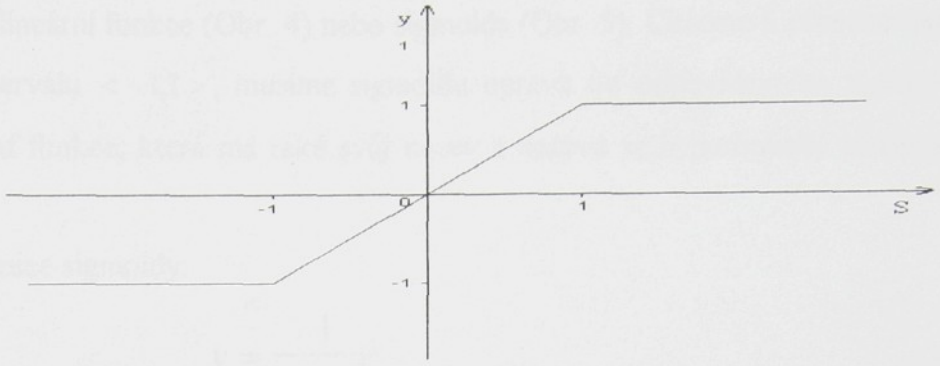
$$y = f(S) \tag{2}$$

Různé matematické modely neuronů se liší jednak reprezentací hodnot x_i a y, jednak tvarem převodní funkce.

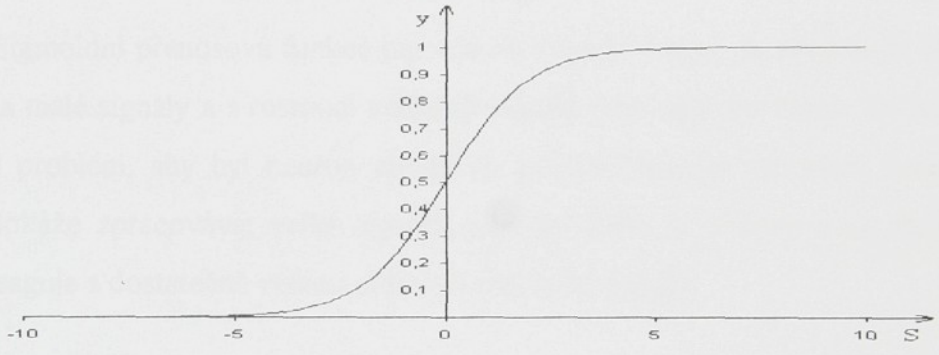
V původním modelu neuronu, jak ho navrhli McCulloch s Pittsem, mohly hodnoty vstupů x_i i hodnota výstupu y nabývat hodnot z množiny $\{-1,0,1\}$. Postupně se začaly používat modely, kdy hodnota x_i byla reprezentována reálným číslem a to buď v intervalu $<0,1>$ nebo v intervalu $<-1,1>$. Hodnota výstupu pak byla z množiny $\{0,1\}$ nebo $\{-1,0,1\}$. Jelikož hodnota vah w_i je reálné číslo, může v tomto případě součet S být jak kladný, tak i záporný. Převodní funkce je tedy jednoduchá. Pokud je $S<0$, tak $y=1$, v opačném případě bude $y=0$ popřípadě $y=-1$. Tento model neuronu dnes většinou nazýváme perceptron.



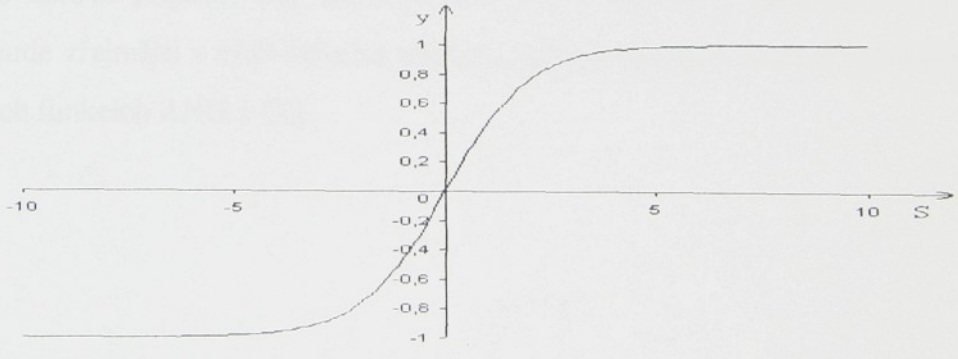
Obr. 3: Převodní funkce - ostrá nelinearita.



Obr. 4: Převodní funkce - saturovaná lineární funkce.



Obr. 5: Převodní funkce - sigmoida.



Obr. 6: Převodní funkce - hyperbolický tangens.

Mluvíme-li o neuronu i v informatice, máme většinou na mysli takový model, kde i hodnota výstupu y je z téhož intervalu jako hodnoty vstupů x_i. V takovém

případě je ale přenosová funkce komplikovanější. Nejčastěji se používá buď saturovaná lineární funkce (Obr. 4) nebo sigmoida (Obr. 5). Chceme-li přenosovou funkci v intervalu $<-1,1>$, musíme sigmoidu upravit do odpovídajícího tvaru a získáme graf funkce, která má také svůj název a nazývá se hyperbolický tangens (Obr. 6).

Rovnice sigmoidy:

$$y = \frac{1}{1+e^{-S}} \tag{3}$$

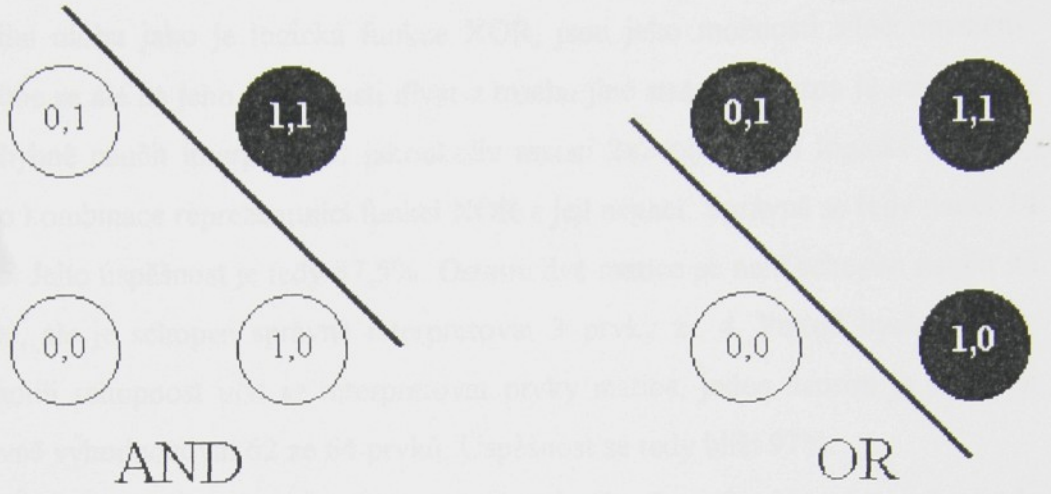
Rovnice hyperbolické tangenty:

$$y = \frac{1-e^{-S}}{1+e^{-S}} = \frac{2}{1+e^{-S}} - 1 \tag{4}$$

Sigmoidní přenosová funkce má velkou výhodu v tom, že neuron je velmi citlivý na malé signály a s rostoucí velikostí signálu jeho citlivost klesá. To mimo jiné řeší problém, aby byl neuron citlivý ve velkém rozsahu vstupních hodnot. Takto dokáže zpracovávat velké signály, aniž by došlo k zahlcení a na druhou stranu reaguje s dostatečně velkou citlivostí i na malé signály.

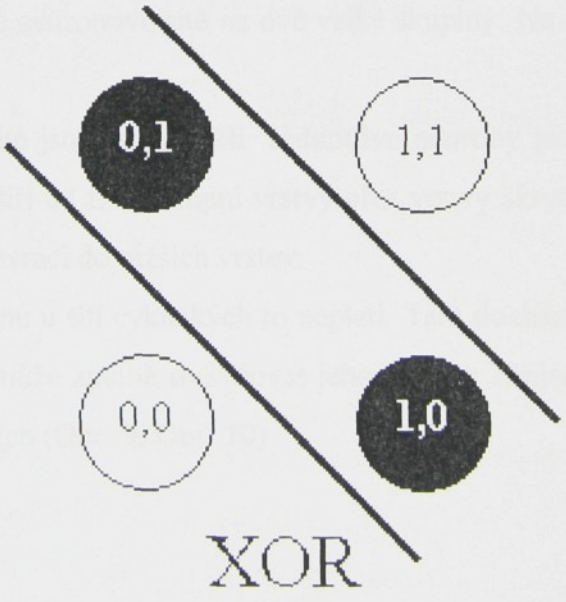
2.1.1 Schopnosti osamoceného neuronu

Pokud jde o množinu úloh, které lze řešit jediným samostatným neuronem, jedná se o takové případy, kdy lze trénovací vzory lineálně separovat. Co to znamená bude zřejmější z následujícího obrázku, kde je ukázána lineární separace na logických funkcích AND a OR.



Obr. 7: Lineární separace trénovacích vzorů logických funkcí AND a OR.

Nejjednodušší příklad funkce, kterou nelze lineárně separovat, je logická funkce XOR, která byla již zmiňovanými Minským a Papertou zneužita k diskreditaci neuronových sítí. Jak je patrno z Obr. 8, nelze rozdělit trénovací vzory jedinou přímkou, z čehož plyne neschopnost jednoho neuronu naučit se tuto funkci správně napodobovat.



Obr. 8: Ukázka funkce, kterou nelze lineárně separovat.

Možná to na první pohled vypadá, že pokud nedokáže jeden neuron řešit tak triviální úlohu jako je logická funkce XOR, jsou jeho možnosti silně omezeny. Musíme se ale na jeho schopnosti dívat z trochu jiné strany. Neuron je schopen se bezchybně naučit interpretovat jakoukoliv matici 2x2 obsahující logické hodnoty mimo kombinace reprezentující funkci XOR a její negaci. Správně se tedy naučí 14 ze 16. Jeho úspěšnost je tedy 87,5%. Ostatní dvě matice se není schopen naučit na 100%, ale je schopen správně interpretovat 3 prvky ze 4. Pokud bychom tedy hodnotili schopnost učit se interpretovat prvky matice, jeden neuron je schopen správně vyhodnocovat 62 ze 64 prvků. Úspěšnost se tedy blíží 97%.

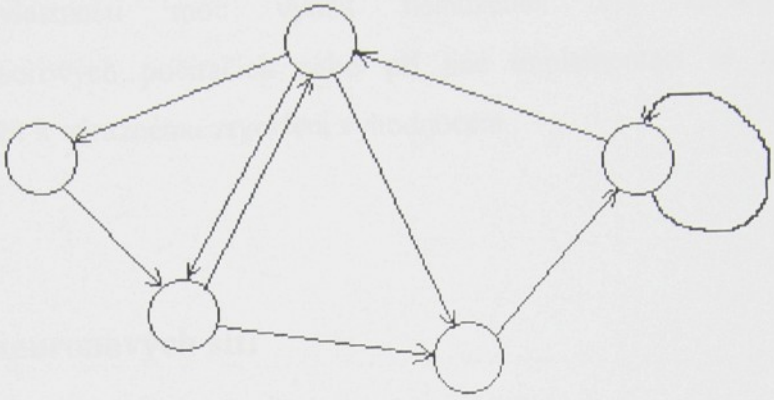
Nahlížíme-li na neuron jako na budoucí náhradu počítačů, asi se nám bude 97% zdát málo. Pokud se však na neuron budeme dívat jako na napodobeninu biologické inteligence, zjistíme, že tato hodnota je vlastně velkým úspěchem.

2.2 Vzájemné spojování neuronů

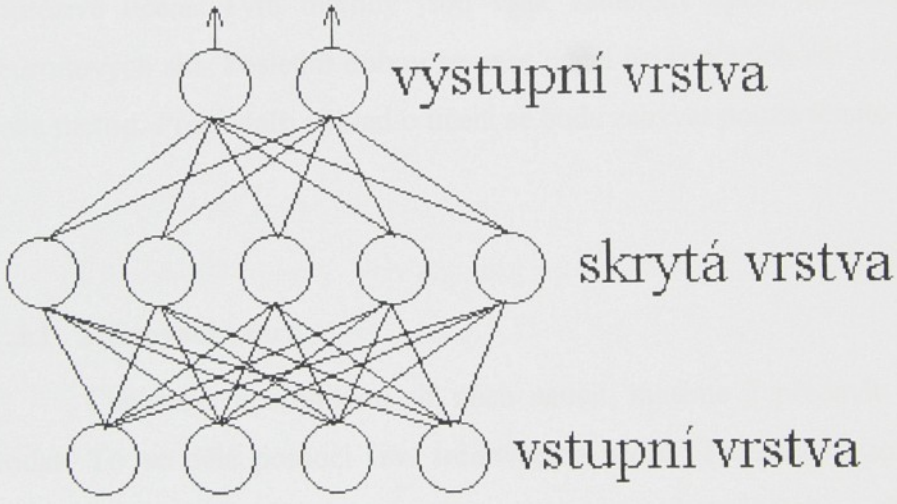
Jednotlivé formální neurony lze vzájemně propojovat, stejně jako jsou propojeny jejich biologické vzory. Vzájemně pospojované neurony nazýváme neuronovou sítí. Vzájemným spojením neuronů vzniká síť, která je schopna řešit daleko větší okruh problémů než samostatný neuron. Z hlediska vzájemného pospojování dělíme neuronové sítě na dvě velké skupiny. Na sítě cyklické a na sítě vrstevné.

Vrstevné sítě jsou jednodušší. Jednotlivé neurony jsou ve vrstvách. Vždy platí, že signál se šíří od tzv. vstupní vrstvy přes vrstvy skryté do výstupní vrstvy. Nikdy se přitom nevrací do nižších vrstev.

Naproti tomu u sítí cyklických to neplatí. Tam dochází k tomu, že výsledná hodnota neuronu může zpětně ovlivňovat jeho vstupy. Nejjednodušší bude ukázat si rozdíl na obrázcích (Obr. 9, Obr. 10).



Obr. 9: Příklad cyklické neuronové sítě.



Obr. 10: Ukázka vrstevné sítě.

Dále se budeme zabývat již jen vrstevnými sítěmi. Jak se vyhodnocuje jeden neuron bylo již ukázáno. Vrstevná síť se vyhodnocuje po vrstvách, přičemž platí, že se postupuje od vrstvy nejnižší, tedy od vrstvy vstupní[3] přes skryté vrstvy do vrstvy výstupní. V každé vrstvě se mohou vyhodnocovat všechny neurony, které vrstva

obsahuje najednou. Pokud simulujeme činnost neuronové sítě na osobním počítači, tak této vlastnosti moc využít nemůžeme. Při zpracování na několikaprocesorových počítačích nebo při jiné implementaci se může této vlastnosti využít k výraznému zrychlení vyhodnocení.

2.3 Učení neuronových sítí

Pro učení neuronových sítí existuje spousta metod. Jako jedny z prvních se začaly používat metody ze skupiny tzv. delta metod, které spočívají v tom, že váhy neuronu se nastavují podle toho, jak moc se liší očekávaný výstup od výstupu skutečného.

Později se začaly objevovat i jiné metody, např. Hebbovo učení nebo soutěživé učení. Tyto metody jsou však zaměřeny spíše na učení cyklických neuronových sítí. Poslední dobou se zase učení neuronových sítí vrací ke skupině delta metod. Proto další výklad o učení se bude zabývat pouze těmito metodami.

2.3.1 Trénovací vzory

Chceme-li neuronovou síť něco naučit, musíme jí předmět učení náležitě podat. To se dělá pomocí tzv. trénovacích vzorů. Trénovací vzory jsou jakési vzorové příklady, na kterých chceme neuronovou síť naučit řešit problém.

Každý trénovací vzor má pevnou strukturu: prvních N hodnot tvoří vstupy neuronové sítě a po nich následuje M hodnot, které jsou požadovaným výstupem na tato vstupní data.

Pro názornost si ukažme, jak by vypadaly trénovací vzory pro jednoduché logické funkce:

---

[3] Pokud mluvíme o vstupní vrstvě, většinou nemáme na mysli neurony, ale vstupy, kterými jsou přiváděny vstupní signály do neuronové sítě.

	AND	
1. vstup	2. vstup	výstup
0	0	0
0	1	0
1	0	0
1	1	1

	OR	
1. vstup	2. vstup	výstup
0	0	0
0	1	1
1	0	1
1	1	1

	XOR	
1. vstup	2. vstup	výstup
0	0	0
0	1	1
1	0	1
1	1	0

Tabulka 1: Trénovací vzory logických funkcí

U jednoduchých funkcí nemusíme nad vytvářením vhodných vzorů nijak moc přemýšlet. Prostě vezmeme všechny možnosti a máme hotovo. U složitějších příkladů, kdy už síť neučíme všechny možnosti, si musíme dát pozor na vhodný výběr vzorů.

Vzory by měly být voleny tak, aby nebyly pouze kladné[4] nebo pouze záporné. Samozřejmě by měly co nejvýstižněji charakterizovat daný problém.

2.3.2 Proces učení jednoho neuronu

Při vyhodnocování jednoho neuronu nedocházelo k žádným problémům. Pokud byla zvolena určitá přenosová funkce, tak se pomocí ní neuron vyhodnotil. Při učení neuronu se však ukazuje, že tento proces již tak triviální nebude. Nelze se však divit, pokud bychom hledali nějaký příklad z lidské zkušenosti, můžeme si vzpomenout, kolik času nám zabralo naučení se vyjmenovaných slov. Určitě to bylo minimálně několik hodin. Při psaní však píšeme slova zcela automaticky, což znamená, že vyjmenované slovo zjistíme během zlomku sekundy.

Nejjednodušší případ učení, který může nastat, je ten, kdy známe všechny učící vzory. Známe tedy jak hodnoty vstupů, tak i příslušnou hodnotu výstupu. Pak už jde jen o to nalézt co nejrychlejší metodu, jak se trénovací vzory naučit.

---

[4] Obrazně řečeno. Kladný má na výstupu 1, záporný 0. Záleží na implementaci neuronu. Má-li neuron výstup od -1 do 1, bude potom 0 znamenat v podstatě -1.

Při učení jednoho neuronu se nejčastěji používá metoda, při které se neuronu předloží nějaký vzor, on ho vyhodnotí a poté se výsledek porovná s požadovaným výstupem. Poté následuje dílčí úprava jednotlivých vah vstupů neuronu tak, aby při dalším učením vyhodnocení byl neuron blíže požadovanému výsledku.

Rychlost učení také závisí na tom, v jakém pořadí se snažíme neuron naučit jednotlivé vzory. Záleží na typu problému. Někdy vyhovuje postupné učení jednotlivých vzorů pořád dokolečka, většinou se ale používá metoda, kdy se provádí učení na náhodně vybraném vzoru.

Matematické vyjádření, které se používá při učení jednoho neuronu má následující tvar:

$$w_i = w_i + \lambda \cdot (y^* - y) \cdot y \cdot (1 - y) \cdot x_i \qquad (5)$$

kde w_i - váha příslušného vstupu x_i

y^* - požadovaný výstup

y - skutečný výstup neuronu

λ - učící konstanta, určuje rychlost učení[5]

Uvedený vzorec vychází z matematického vyjádření derivace převodní funkce. Jeho odvozením se zabývat nebudeme. Ze vzorce jsou vidět některé zásady, které vyplývají z logiky věci. Je například zřejmé, že pokud je hodnota vstupu nulová, tak se nemá cenu učením zabývat. Dále je vidět, že učení je tím rychlejší, čím je větší neshoda mezi skutečným a očekávaným výstupem. Na druhou stranu je patrno, že pokud si je neuron svým výsledkem jistý (hodnota se blíží 0 nebo 1), velmi nerad se svého přesvědčení vzdává.

Uvedený vzorec platí tehdy, jsou-li hodnoty vstupů a výstupu z intervalu $<0,1>$. Pokud bychom chtěli získat vzorec na intervalu $<-1,1>$, dostaneme matematickou úpravou tvar:

$$w_i = w_i + \lambda \cdot (y^* - y) \cdot (1 - y^2) \cdot x_i \qquad (6)$$

[5] Ale i zde platí staré pořekadlo: „Práce kvapná málo platná."

Z uvedených vzorců je patrné, že mohou pracovat jenom tam, kde známe přesně očekávaný výstup. A ten většinou známe jen u výstupní vrstvy. Touto metodou lze tedy řešit jenom jednovrstvé sítě.

Na druhou stranu tento vzorec zaručuje, že pokud existuje řešení, které lze lineárně separovat, v konečném[6] čase se k němu dostane.

Abychom zjistili, jak daleko jsme od hledaného řešení, zavádí se tzv. kvadratická[7] odchylka. Pro jeden neuron ji můžeme vypočítat třeba takto:

$$\sigma = \frac{1}{4}\sum_{i=0}^{n}(y_i^* - y_i)^2 \tag{7}$$

Konstanta $\frac{1}{4}$ není podstatná. Zaručuje nám jen, že chyba jednoho neuronu nebude vyšší než je počet jeho vstupů. Z důvodu urychlení při implementaci na počítači ji můžeme vynechat. Takto se tedy vypočítá odchylka pro jeden vzor. Při učení neuronu můžeme využívat hodnotu odchylky k stanovení rychlosti učení. Dále můžeme učit neuron tak, že se snažíme zmenšovat odchylku u toho vzoru, kde je největší. Takhle je výhodné postupovat, jde-li řešení lineárně separovat. Pokud se ale snažíme nalézt co nejlepší řešení, aniž bychom předem mohli cokoliv říci o lineární separaci, je lepší minimalizovat při učení celkovou chybu pro všechny vzory, která se vypočte:

$$\vartheta = \sum_{j=1}^{m}\sigma_j \tag{8}$$

2.3.3 Učení vícevrstvých sítí

Učení vícevrstvých neuronových sítí se ukázalo jako největší slabina neuronových sítí vůbec. Dokud nebyla navržena metoda BACKPROPAGATION, neexistoval jediný algoritmus, který by vedl s jistotou k vytčenému cíli.

Při učení nějaké skupiny vzorů jde o to, dostat globální minimum chybové funkce (rovnice (7) a (8)). Metoda BACKPROPAGATION nám sice nezaručuje, že toto globální minimum nalezne z libovolného počátečního nastavení vah, ale

---

[6] V konečném ještě zdaleka nemusí znamenat v přijatelném.

alespoň nám zaručuje, že nalezne nějaké lokální minimum. Proto se při její implementaci snažíme najít globální minimum tak, že tedy hledáme lokální minimum a pokud s ním nejsme spokojeni, přenastavíme počáteční váhy a minimum hledáme znovu. Při rozumném počtu neuronů v síti není zase lokálních minim tolik, abychom se k cíli nedobrali.

Princip metody BACKPROPAGATION spočívá v tom, že se nejprve klasickým způsobem učí výstupní vrstva. Při každé změně váhy nějakého neuronu ve výstupní vrstvě se podíváme, zda nějaký neuron ve vrstvě nižší se váže na vstup odpovídající dané váze. Pokud ano, upravíme odpovídajícím způsobem váhy i jemu.

2.4 Implementace neuronové sítě v jazyce Pascal

Chceme-li implementovat neuronovou síť na klasickém počítači PC, musíme si uvědomit, že tím ztrácíme jednu důležitou vlastnost neuronových sítí a to jejich schopnost paralelního zpracování. Na počítači musíme všechny neurony zpracovávat postupně.

Na druhou stranu je řešení neuronové sítě na počítači o hodně levnější než implementace jinými metodami. Jimi se rozumí sestavení sítě ze speciálních hardwarových neuronů, ale v poslední době také často zkoušené neuronové sítě z opravdových biologických neuronů.

Zde naznačená implementace je v programovacím jazyce Pascal či spíše v jeho variantě Borland Pascal. Tento jazyk byl vybrán pro svou přehlednost a velkou rozšířenost.

Jako základ neuronové sítě byl vybrán matematický neuron s převodní funkcí ve tvaru hyperbolické tangenty. Z toho vyplývá, že jak vstupy, tak i výstupy tohoto neuronu jsou reálná čísla z intervalu $< -1,1 >$. Váhy jednotlivých vstupů, jsou pak reálná čísla z intervalu $< -\infty, \infty >$.

V programu jsou pak všechna tato reálná čísla reprezentována reálným typem *double*. Je to proto, že výpočty pomocí tohoto typu jsou rychlejší než

---

[7] Někdy se též používá místo kvadratické pouze absolutní odchylka.

výpočty s typem *real*. Nevýhodou tohoto řešení je nutnost práce na počítači s koprocesorem, což ale snad v dnešní době není zas až tak omezující. Původním záměrem bylo napsat celou aritmetiku v celých číslech, ale ukázalo se, že urychlení o pár procent nevyváží sníženou přehlednost programu.

Implementace neuronové sítě je řešena na úrovni jednotky, ve které jsou naprogramovány základní operace s neurony. Jednotka umožňuje např. vytvoření libovolné sítě neuronů, umožňuje tuto síť vyhodnotit i naučit jednotlivé vzory. Mezi základní funkce jednotky patří také uložení celé sítě na disk, popřípadě načtení z disku.

2.5 Popis jednotky Sit

Jednotka Sit umožňuje snadné naprogramování neuronové sítě. Zdrojový text této unity je v příloze A, příklad jejího využití na konkrétním případě je v příloze B. Jednotka obsahuje následující procedury a funkce využitelné v hlavním programu.

vytvorneuron (a: word)

Procedura, která vytvoří neuron na pozici a v tabulce neuronů.

zrusneuron (a: word)

Procedura, která zruší neuron v tabulce neuronů.

vytvorsit (a, b, c: word)

Procedura, která vytvoří neuronovou síť s jednou skrytou vrstvou. Jednotlivé parametry udávají počet vstupů, počet neuronů v skryté vrstvě a počet neuronů ve výstupní vrstvě. Pro vytvoření jednovrstvé neuronové sítě se zadá b=0. Struktura této sítě je následující: Vstupy neuronů ve skryté vrstvě jsou shodné se vstupy celé neuronové sítě, zatímco vstupy výstupní vrstvy tvoří jednak vstupy celé sítě a jednak výstupy neuronů ve skryté vrstvě. Například zadáme-li *vytvorsit (7,10,2)*, pak neurony skryté vrstvy budou mít každý po sedmi vstupech, zatímco neurony vrstvy výstupní budou mít každý vstupů sedmnáct.

zrussit

Procedura, která zruší všechny neurony v síti. Tuto proceduru je nutno volat vždy před ukončením programu a vždy před načtením pomocí procedury *load*.

load (s: string)

Procedura, která načte z disku konfiguraci sítě s názvem, který je udaný jako její parametr, uloženou předtím procedurou *save*. Procedura nahraje celou konfiguraci sítě včetně vah jednotlivých vstupů neuronů.

save (s: string)

Procedura s parametrem jméno souboru, která uloží na disk soubor s konfigurací aktuální sítě včetně vah jednotlivých vstupů neuronů.

vyhodnotneuron (a: word)

Procedura, která vyhodnotí jeden neuron. Výsledek bude zapsán do pole vstupů a výstupů.

vyhodnotsit

Procedura, která vyhodnotí celou síť a to tak, že nejprve vyhodnocuje neurony ve skrytých vrstvách a nakonec vyhodnotí neurony ve vrstvě výstupní. Výsledek zapisuje opět do pole vstupů a výstupů.

naucneuron (a, b, c: word)

Procedura, která implementuje dvě metody učení. Obě učí neuron *a* vzor *b*, první metoda při *c=1* učí pouze tento neuron, druhá učí i všechny neurony, které jsou pod učeným neuronem. Druhé metody se používá při učení vícevrstvých neuronových sítí.

naucsit (c: longint)

Procedura se snaží bezchybně naučit celou síť všechny vzory. Parametr *c* udává počet sekund, po kterých má snahy zanechat. V hlavním programu se pak můžeme pokusit přenastavit váhy neuronů ve skryté vrstvě a poté zavolat tuto proceduru znovu.

Tato procedura funguje tak, že se snaží přiučit danou síť 10000 náhodných vzorů z tabulky vzorů a to náhodnou velikostí učící konstanty *lambda*. Poté zjistí, zda se jí nepodařilo nalézt řešení nebo zda nevypršel časový interval a pokud ne, tak přiučuje síť dále.

chybavzoru (b: word): double

Funkce, která vrací chybu vzoru *b*. Chyba vzoru se počítá jako odchylka výsledku po vyhodnocení od vzorové hodnoty.

chybasite: double

Funkce, která vrací celkovou chybu sítě na všech vzorech. Je to součet chyb jednotlivých vzorů.

pocetchybvzoru (b: word): word

Funkce, která určí kolik výstupních neuronů odpovídá chybně u konkrétního vzoru *b*.

pocetchybsite: word

Funkce, která vrací totéž jako funkce *pocetchybvzoru*, ale jako součet pro všechny vzory.

3. Praktické příklady řešené pomocí neuronových sítí

3.1 Problém prvočísel

Zadání: Navrhnout neuronovou síť, která by rozhodovala o tom, zda číslo je prvočíslo. Jako vstup sítě je pětice bitů reprezentující binární zápis čísla od 0 do 31. Jako výstup je ke každé pětici bit, určující, zda dané číslo je prvočíslo.

0	000000	11	010111	22	101100
1	000010	12	011000	23	101111
2	000101	13	011011	24	110000
3	000111	14	011100	25	110010
4	001000	15	011110	26	110100
5	001011	16	100000	27	110110
6	001100	17	100011	28	111000
7	001111	18	100100	29	111011
8	010000	19	100111	30	111100
9	010010	20	101000	31	111111
10	010100	21	101010		

Tabulka 2: Trénovací vzory prvočísel

Řešení: Problém prvočísel je jedním z problémů, které nejdou řešit lineární separací. Přesvědčit se o tom můžeme například tím, že úlohu zkusíme řešit jako by lineární separace byla možná.

Vytvoříme tedy jednovrstvou síť s jedním neuronem, který bude mít jako své vstupy 5 bitů čísla a jeho výstup bude určovat, zda číslo na vstupu je prvočíslo či nikoliv.

Zjistíme, že jeden neuron je schopen se naučit problém tak, že se dopouští na celé trénovací množině vzorů čtyř omylů.

Na řešení tohoto problému je tedy zapotřebí vícevrstvá síť. Nelze však předem říci, kolik neuronů musí být ve skryté vrstvě. Ze znalosti trénovací

množiny však můžeme říci, že maximální počet neuronů ve skryté vrstvě je 11. Právě tolik je totiž prvočísel, která lze reprezentovat 5 bity. Pokud by tedy každý neuron ve skryté vrstvě detekoval jedno prvočíslo a jediný neuron výstupní vrstvy by na všech 11 neuronech pod ním prováděl operaci OR, síť by dávala správné výsledky pro všechny trénovací vzory.

Toto řešení by ale zcela jistě nebylo zobecňující. Pokud bychom vynechali jakýkoliv vzor, síť by ho špatně vyhodnocovala. Je známo, že čím menší je počet skrytých neuronů, tím větší je zobecňující schopnost neuronové sítě.

Pokusíme se tedy problém řešit tak, že začneme s neuronovou sítí, která má ve skryté vrstvě 11 neuronů. Pokud se pro takovou konfiguraci sítě podaří nalézt funkční řešení, snížíme počet neuronů ve skryté vrstvě o jeden. Poté se znovu pokoušíme nalézt řešení. Pokud program řešení nenalézá, zkouší neustále náhodně měnit váhy skrytých neuronů. Je-li snaha neúspěšná, ještě to neznamená, že lepší řešení neexistuje. Zde záleží jen na čase a rychlosti počítače.

Tabulka 3 nám poskytne představu, jaká je časová náročnost hledání řešení pro jednotlivé počty neuronů ve skryté vrstvě. Zde uváděné časy jsou z testu na počítači s procesorem Pentium MMX na 250 MHz.

Akce	Čas
Spuštění programu.	22:01.20
Nalezení řešení pro 11 neuronů ve skryté vrstvě.	22:01.23
Nalezení řešení pro 10 neuronů ve skryté vrstvě.	22:09.49
Nalezení řešení pro 9 neuronů ve skryté vrstvě.	22:38.18
Nalezení řešení pro 8 neuronů ve skryté vrstvě.	22:40.23
Nalezení řešení pro 7 neuronů ve skryté vrstvě.	22:46.32
Nalezení řešení pro 6 neuronů ve skryté vrstvě.	22:52.37
Nalezení řešení pro 5 neuronů ve skryté vrstvě.	23:26.58
Nalezení řešení pro 4 neurony ve skryté vrstvě.	00:24.16
Nalezení řešení pro 3 neurony ve skryté vrstvě.	05:20.47

Tabulka 3: Ukázka časové náročnosti řešení NS

Jak je z tabulky patrno, řešení pro větší počet neuronů byla otázkou několika minut, zatímco řešení pro čtyři a tři neurony už záležitostí hodin. Přestože program běžel ještě dále, řešení pro dva neurony ve skryté vrstvě se mu už nalézt nepodařilo. To by ovšem v principu ještě neznamenalo, že toto řešení neexistuje.

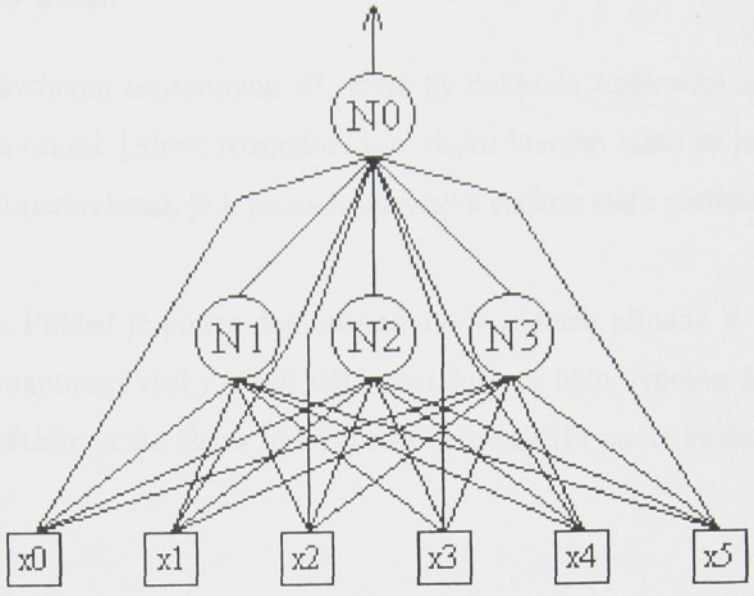
Protože byla snaha zjistit, nakolik je program na učení sítě dobrý, bylo potřeba zjistit minimální počet neuronů ve skryté vrstvě jinou metodou. Jako nejlepší se jevila metoda hrubé síly[8], kdy byly postupně vyzkoušeny všechny kombinace jednotlivých prvočísel a zjišťovalo se, zda je lze lineárně separovat. Tabulka 4 ukazuje, které kombinace prvočísel lze vzájemně separovat. Nejsou v ní takové kombinace, které jsou podmnožinou vypsaných. Například jedenáctice 11011001100 znamená, že první, druhé, čtvrté, páté, osmé a deváté prvočíslo lze separovat od ostatních čísel. Znamená to tedy, že zmiňovaná prvočísla dokáže odhalit jeden jediný neuron.

11011001100	01110001100	**01010011100**	01010001101
00110100000	00100100010	**00000100011**	00000000111

Tabulka 4: Separace vzorů prvočísel

My potřebujeme, aby se všechna prvočísla detekovala správně. Potřebujeme tedy, aby v jedenáctici byly samé jedničky. Budeme-li tedy mít ve skryté vrstvě neurony, z nichž každý dokáže separovat jednu z tučně vytištěných jedenáctic v tabulce, dostaneme jedenáctici 11011111111. Zbývá nám tedy už jen jedno prvočíslo a to už dokáže správně odhalit výstupní neuron.

[8] Mohlo by se zdát, že se jedná o časově náročnou úlohu. Ve skutečnosti výpočet trval pouze pár minut.



Obr. 11: Konfigurace nejjednodušší sítě řešící problém prvočísel.

Je tedy patrno, že stačí tři neurony ve skryté vrstvě. Konfigurace této sítě je na Obr. 11.

neuron N1	neuron N2	neuron N3	neuron N0
$w_0 = -200$	$w_0 = -200$	$w_0 = -200$	$w_0 = 150$
$w_1 = -50$	$w_1 = 40$	$w_1 = 40$	$w_1 = -50$
$w_2 = -100$	$w_2 = -120$	$w_2 = 80$	$w_2 = -50$
$w_3 = -50$	$w_3 = -40$	$w_3 = 80$	$w_3 = 50$
$w_4 = 150$	$w_4 = 80$	$w_4 = -40$	$w_4 = 0$
$w_5 = 100$	$w_5 = 120$	$w_5 = 80$	$w_5 = 50$
			$w_6 = 100$
			$w_7 = 100$
			$w_8 = 100$

Tabulka 5: Váhy neuronové sítě řešící problém prvočísel

Závěr: Na tomto příkladu je vidět, že vícevrstvá neuronová síť je schopna se naučit řešit celkem složitý problém. Cenou, v tomto případě, je dlouhá doba učení. Ta je ovšem závislá na tom, jestli hledáme jakékoliv řešení nebo se snažíme nalézt řešení v nějakém ohledu nejlepší. Na druhou stranu, klasický algoritmus by uvedený problém zvládl také a asi i podstatně snáze.

3.2 Vlajky států

Zadání: Navrhnout neuronovou síť, která by dokázala z několika zodpovězených jednotlivých otázek jednak rozhodnout, o vlajku kterého státu se jedná a poté se také pokusit rozhodnout, je-li popisovaná vlajka vlajkou státu patřícího do EU.

Poznámka: Příklad je pouze demonstrativní. V žádném případě si neklade za cíl schopnost rozpoznat vlajky všech států. Pro ilustraci bylo vybráno šestnáct vlajek. V případě většího počtu vlajek by bylo také nutné zvětšit počet kladených otázek.

Otázky:

1. Obsahuje vlajka kříž?
2. Jsou na vlajce svislé pruhy od shora až dolů?
3. Jsou na vlajce vodorovné pruhy po celé délce?
4. Je vlajka tříbarevná?
5. Vyskytuje se na vlajce modrá barva?
6. Vyskytuje se na vlajce bílá[9] barva?
7. Má vlajka klasický tvar?

Název státu / číslo otázky	1	2	3	4	5	6	7	Patří stát do EU?
Belgie		A		A			A	ANO
Česká republika				A	A	A	A	NE
Dánsko	A					A	A	ANO
Francie		A		A	A	A	A	ANO
Itálie		A		A		A	A	ANO
Japonsko						A	A	NE
Kanada		A				A	A	NE
Maďarsko			A	A		A	A	NE
SRN			A	A			A	ANO
Polsko			A			A	A	NE
Řecko	A		A			A	A	ANO
USA			A	A	A	A	A	NE
Švédsko	A				A		A	ANO
Švýcarsko	A					A		NE

[9] Ponechme stranou skutečnost, že bílá nebývá odborníky považována za barvu.

Velká Británie	A			A	A	A	A	ANO
Finsko	A				A	A	A	ANO

Tabulka 6: Trénovací vzory vlajek států

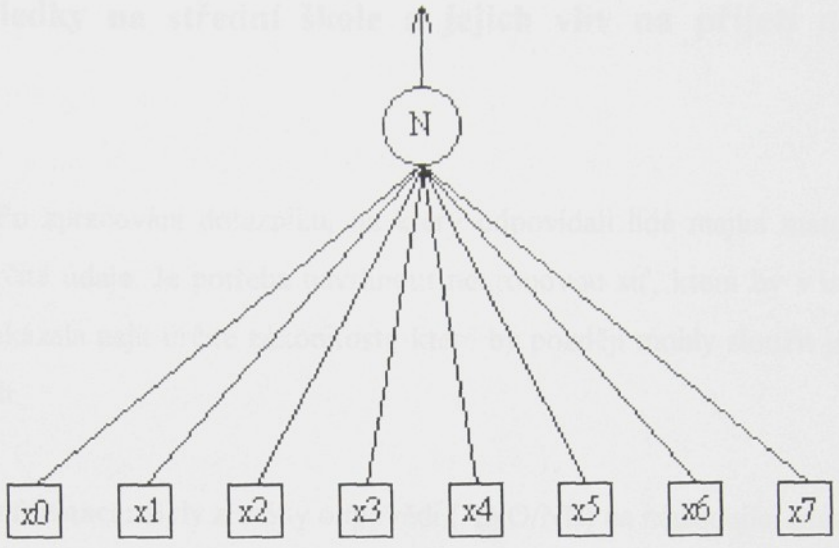
Řešení: Na rozdíl od předcházejícího problému prvočísel, se nám tento problém může jevit jako složitější. Jestli tomu tak je opravdu, se dozvíme z toho, jak rychle se nám podaří vytvořit neuronovou síť řešící tento problém.

Rozdělme problém na dvě části. V první se pokusme navrhnout síť, která by ze zadaných odpovědí určila stát. Budeme postupovat stejně jako v případě prvočísel. Nejprve zkusíme, zda náhodou nelze řešení problému lineárně separovat. Navrhneme tedy neuronovou síť, která bude mít sedm vstupů (jednotlivé odpovědi na otázky) a šestnáct výstupů (států je v našem případě šestnáct).

Při trénování této sítě se rychle ukáže schopnost sítě naučit se trénovací vzory. Můžeme tedy říci, že řešení problému bylo lineárně separabilní.

Druhým úkolem je rozhodnout, zda se jedná o vlajku státu, který je členem EU. Pokud se snažíme navrhnout neuronovou síť řešící tento problém, pak nás asi jako první napadne možnost, že vyjdeme z neuronové sítě rozpoznávající jednotlivé státy. K té pak přidáme další vrstvu s jedním neuronem, který bude rozhodovat o tom, patří-li stát do EU či nikoliv. Tím nám tedy vznikla neuronová síť mající sedm vstupů, šestnáct skrytých neuronů a jeden neuron výstupní.

Po implementaci této sítě zjistíme, že síť opravdu splňuje naše požadavky a bez chyby určuje příslušnost k EU. Musíme si však položit otázku, jestli je námi navržená konfigurace sítě ta nejlepší. Zkusíme tedy snížit počet neuronů ve skryté vrstvě a vidíme, že přestože počet neuronů ve skryté vrstvě snižujeme, síť pořád dává výsledky podle našich představ. Nakonec se ukáže, že ve skryté vrstvě nemusí být ani jeden neuron. Uvedený problém příslušnosti k EU dokáže vyřešit jediný výstupní neuron.



Obr. 12: Konfigurace nejjednodušší sítě řešící příslušnost k EU.

Závěr: Tento příklad z oblasti zeměpisu nám ukazuje, jak se neuronová síť dokáže elegantně naučit řešit problém, který se na první pohled jeví složitější než dříve zkoumaný problém prvočísel. Při řešení problému vlajek najde učící algoritmus neuronové sítě optimální konfiguraci a ještě ji stačí naučit správně interpretovat jednotlivé učící vzory během několika málo sekund.

Takhle rychle se daný problém nenaučí řešit ani člověk, který v případě s prvočísly jasně nad počítačem vítězil. Algoritmické řešení tohoto problému také není zrovna jednoduché.

Z uvedených dvou příkladů je jasně vidět, že jsou problémy, které se hodí na zpracování neuronové sítě více a naopak problémy, které nejsou na zpracování neuronovou sítí nejvhodnější.

3.3 Výsledky na střední škole a jejich vliv na přijetí na školu vysokou

Zadání: Po zpracování dotazníku, na který odpovídali lidé mající maturitu byly získány určité údaje. Je potřeba navrhnout neuronovou síť, která by v uvedených údajích dokázala najít určité zákonitosti, které by později mohly sloužit jako zdroj předpovědí.

Získané informace: Byly získány odpovědi (ANO/NE) na následující otázky:

- Je zkoumaná osoba muž?
- Je zkoumaná osoba žena?
- Dostala zkoumaná osoba někdy na střední škole z jazyků včetně češtiny na vysvědčení jedničku?

  ... dvojku? ... trojku? ...čtyřku? ... pětku?
- Je rozpětí získaných známek z těchto předmětů jedna?
- Je rozpětí získaných známek z těchto předmětů dvě?
- Je rozpětí získaných známek z těchto předmětů tři nebo více?
- Dostala zkoumaná osoba někdy na střední škole z matematiky nebo fyziky na vysvědčení jedničku?

  ... dvojku? ... trojku? ...čtyřku? ... pětku?
- Je rozpětí známek z matematiky a fyziky jedna?

  ... dvě? ... tři nebo více?
- Dostala zkoumaná osoba někdy na střední škole z biologie, zeměpisu nebo dějepisu na vysvědčení jedničku?

  ... dvojku? ... trojku? ...čtyřku? ... pětku?
- Je rozpětí známek z biologie, zeměpisu a dějepisu jedna?

  ... dvě? ... tři nebo více?
- Měla zkoumaná osoba na střední škole nejraději jazyky včetně češtiny?
- Měla zkoumaná osoba na střední škole nejraději matematiku a fyziku?

- Měla zkoumaná osoba na střední škole nejraději biologii, zeměpis a dějepis?
- Kladla zkoumaná osoba na druhé místo své oblíbenosti jazyky včetně češtiny?

... matematiku a fyziku? ... biologii, zeměpis a dějepis?

- Kladla jako nejméně oblíbené předměty zkoumaná osoba jazyky včetně češtiny?

... matematiku a fyziku? ... biologii, zeměpis a dějepis?

- Dostala se zkoumaná osoba na práva?
- Dostala se zkoumaná osoba na medicínu?
- Dostala se zkoumaná osoba na pedagogickou fakultu?
- Dostala se zkoumaná osoba na techniku?
- Dostala se zkoumaná osoba na školu ekonomického zaměření?
- Přestože zkoumaná osoba dělala přijímací zkoušky na práva, nebyla přijata. Je toto tvrzení pravdivé?
- Přestože zkoumaná osoba dělala přijímací zkoušky na medicínu, nebyla přijata. Je toto tvrzení pravdivé?
- Přestože zkoumaná osoba dělala přijímací zkoušky na pedagogickou fakultu, nebyla přijata. Je toto tvrzení pravdivé?
- Přestože zkoumaná osoba dělala přijímací zkoušky na techniku, nebyla přijata. Je toto tvrzení pravdivé?
- Přestože zkoumaná osoba dělala přijímací zkoušky na školu ekonomického zaměření, nebyla přijata. Je toto tvrzení pravdivé?

Poznámka: Protože dotazník byl položen asi stu lidí, jsou odpovědi na kladené otázky v příloze C.

3.3.1 Určení pohlaví podle výsledků na střední škole

Položme si otázku, jestli lze ze získaných údajů o středoškolských výsledcích zjistit pohlaví dotazovaného. Zkusíme tedy vytvořit síť, která by tuto otázku řešila.

Vytvoříme tedy jednovrstvou neuronovou síť s jedním neuronem, která bude mít 43 vstupů. Jako vstupy nám slouží odpovědi na otázky o prospěchu, o oblibě předmětů a o tom, na jaké vysoké školy se dotazovaný dostal či nedostal. Výstup bude odpověď na otázku, zda zkoumaný člověk je muž[10]. Správné odpovědi na tuto otázku (trénovací vzory) jsou obsaženy v kompletním seznamu otázek.

Při zkoumání výsledků neuronové sítě nás bude zajímat jak schopnost správně určit pohlaví, tak které otázky byly pro určení pohlaví nejdůležitější. To poznáme podle vah jednotlivých vstupů.

Výsledky: Uvedená síť vykazovala po skončení učení na testovaných datech tři hrubé chyby (tzn. odpovídala opačně). To je dobrý výsledek, uvážíme-li, že trénovacích vzorů bylo kolem sta. Síť tedy s úspěšností asi 97% určovala pohlaví. Co se týče důležitosti jednotlivých otázek, ukázalo se, že nejdůležitějších bylo následujících osm.

1. *Dostala někdy zkoumaná osoba z biologie, zeměpisu nebo dějepisu na vysvědčení trojku?* - kladná odpověď znamená, že to bude spíše muž.
2. *Měla zkoumaná osoba nejraději jazyky včetně češtiny?* - kladná odpověď znamená, že to bude spíše žena.
3. *Nedostala se zkoumaná osoba na techniku, přesto, že dělala přijímací zkoušky?* - pokud se nedostala, ukazuje to spíše na muže.
4. *Měla zkoumaná osoba nejméně ráda matematiku a fyziku?* - kladná odpověď ukazuje spíše na ženu.
5. *Dostala někdy zkoumaná osoba z biologie, zeměpisu nebo dějepisu na vysvědčení čtyřku?* - kladná odpověď znamená, že to bude spíše muž.
6. *Dostala se zkoumaná osoba na školu s ekonomickým zaměřením?* - kladná odpověď znamená, že to bude spíše muž.
7. *Dostala někdy zkoumaná osoba z biologie, zeměpisu nebo dějepisu na vysvědčení jedničku?* - kladná odpověď znamená, že to bude spíše žena.

---

[10] Ženu poznáme tak, že to není muž.

8. *Nedostala se zkoumaná osoba na práva, přesto, že dělala přijímací zkoušky?* - pokud se nedostala, ukazuje to spíše na muže.

Z uvedených otázek je patrno, že některé ukazují jisté trendy, například to, že skupina předmětů biologie, zeměpis, dějepis jde spíše ženám, nebo že ženy nemají rády matematiku a fyziku. Jiné otázky nám ale spíše ukazují na nedostatečný počet vstupních dat. Pokud síť usoudí, že na práva se spíše nedostane muž, pravděpodobně to nic neznamená vzhledem k malému počtu lidí ze zkoumané skupiny, kteří se na práva hlásili.

Dále ale existuje skupina otázek, které na určení pohlaví nemají téměř žádný vliv. Jsou to například:

- *Bylo rozpětí známek z biologie, zeměpisu a dějepisu dvě?*
- *Měla zkoumaná osoba nejraději biologii, zeměpis a dějepis?*
- *Měla zkoumaná osoba nejméně ráda jazyky včetně češtiny?*
- *Nedostala se zkoumaná osoba na medicínu, přestože dělala přijímací zkoušky?*
- *Bylo rozpětí známek z jazyků včetně češtiny dvě?*
- *Bylo rozpětí známek z matematiky a fyziky tři a více?*

Zhodnocení: Na uvedeném příkladu je vidět, že naučením sítě neziskáváme informace pouze o jejím výstupu, ale můžeme zjistit i informace, které jsou mnohem zajímavější (není ani tak zajímavé, jaké pohlaví má konkrétní zkoumaná osoba, jako je zajímavé, že určité předměty jdou lépe jednomu pohlaví).

3.3.2 Jak poznat, vyšly-li studentovi plány

Další poznatek, který bychom chtěli ze získaných dat zjistit je, podle čeho poznáme, bude-li student úspěšný. Jako úspěch budeme brát, neexistuje-li v jeho odpovědích žádná, která by říkala, že se hlásil na nějakou vysokou školu a přesto se na ní nedostal. Pokud se například student hlásil jenom na práva a dostal se na ně, byl úspěšný, pokud se hlásil na práva a na medicínu a dostal se pouze na práva, už ho budeme považovat za neúspěšného. Pokud se student nehlásil nikam, evidentně byly jeho plány splněny, tedy byl úspěšný.

Navrhneme tedy neuronovou síť, která by se tento problém snažila řešit. Síť bude opět samostatný neuron, jako vstupy budou všechny dostupné informace a výstupem bude odpověď na zkoumanou otázku.

Při učení této sítě vidíme, že je rychle nalezeno stoprocentně správné řešení. Síť nevykazuje ani jednu hrubou chybu. Podíváme-li se však na váhy jednotlivých vstupů, zjistíme, že správné rozhodování se děje hlavně na základě pěti otázek ptajících se na to, zda student někde neudělal přijímací zkoušky. To je ve shodě se zadáním. Řekli jsme totiž, že je neúspěšný, jestliže někde neudělá přijímací zkoušky. Nedozvěděli jsme se tedy nic nového.

Zkusíme tedy navrhnout síť, která by jako vstupy neměla otázky týkající se přijímacích zkoušek. Tato síť by nám měla poskytnout informace o závislosti na pohlaví, známkách a preferencích.

Učení takovéto sítě už probíhá výrazně pomaleji. Po dlouhé době učení se síť naučí interpretovat trénovací vzory s šesti hrubými chybami. Úspěšnost učení je ale stále ještě vysoká, kolem 94%.

Co tedy podmiňuje úspěšnost studenta? Je to těchto několik bodů:

- Neměl by mít na vysvědčení čtyřku z jazyků včetně češtiny.
- Naopak vychází, že čtyřka z biologie, zeměpisu nebo dějepisu je pro něho výhodou.
- Jeho nejméně oblíbenými předměty by měli být jazyky včetně češtiny.
- Čtyřka z matematiky či fyziky moc vhodná není.
- Pokud je rozpětí známek z matematiky a fyziky tři a více, nevěští to nic dobrého.
- A naopak, pokud je rozpětí známek z jazyků jedna, jeho šance rostou.
- Jeho nejoblíbenějšími předměty by měly být matematika a fyzika.

Opět je vidět, že mezi získanými informacemi jsou i některé nepřesnosti způsobené malým počtem trénovacích vzorů. S největší pravděpodobností nemůžeme brát vážně doporučení neuronové sítě ke čtyřce z biologie, zeměpisu nebo dějepisu. Shodou okolností je v trénovacích vzorech někdo, kdo zmíněnou čtyřku dostal a byl úspěšný a nikdo, kdo by ji dostal a byl neúspěšný. Pokud ale neuronová síť tvrdí, že nejoblíbenější předměty úspěšných studentů jsou

matematika a fyzika, asi na tom něco bude. Toto tvrzení je totiž podloženo větším počtem vzorů než předchozí.

Z vah neuronové sítě ale můžeme vyčíst i pravý opak, tedy na čem skoro vůbec nezáleží:

- Jednička z matematiky či fyziky není podstatná.
- Dvojka z biologie, zeměpisu či dějepisu je také nicneříkající.
- A taktéž trojka z jazyka mnoho nenapoví.

4. Závěr

Úkolem této diplomové práce bylo seznámení s neuronovými sítěmi a snaha o jejich využití ve škole. Ukázalo se, že škola je pro neuronové sítě zvláště vhodné prostředí. Tyto sítě tam mohou sloužit v různých předmětech a alespoň částečně demonstrovat schopnosti umělé inteligence. Může se zdát, že umělá inteligence je hodně nadnesený pojem, ale musíme si uvědomit, že v této diplomové práci je ukázán jen směr, kterým by se mohly neuronové sítě v budoucnu ve škole ubírat.

Pokud dělíme neuronové sítě na jednovrstvé a vícevrstvé, můžeme říci, že vícevrstvé sítě jsou rozhodně silnějším nástrojem, jehož mocnost je však vykoupena velkou časovou náročností učení (viz. učení prvočísel). Naproti tomu jednovrstvé sítě se sice nedokáží naučit zdaleka všechno, zato nám ale mohou v krátkém čase poskytnout zajímavé pohledy na vstupní data (viz. zpracování dotazníku).

Shrneme-li tedy poznatky získané v diplomové práci o neuronových sítích, můžeme říci, že klasické počítače jsou vhodným nástrojem na simulaci těchto sítí. Při učení metodou zpětného šíření se lze dobrat k správnému výsledku i pro vícevrstvé sítě.

Příloha A: Jednotka Sit

```
unit Sit;        {Tato unita umožňuje snadné naprogramování neuronové sítě.}

interface

uses crt, dos;

{Tyto konstanty lze volit pro lepší využití proměnných v rámci segmentu RAM}
const MPVN = 50;          {maximální počet vstupů jednoho neuronu}
      MPVZ = 128;         {maximální počet vzorů}
      MPVV = 50;          {maximální počet vstupů a výstupů}
      PN   = 20;          {maximální počet neuronů v síti}

type  neuron = record
            x: array [0..MPVN] of word;        {odkaz do tabulky vstupů}
            w: array [0..MPVN] of double;      {váha příslušného vstupu}
            y: word;                           {výstup - odkaz do tabulky vstupů}
            vrstva: word;                      {číslo vrstvy, ve které je neuron}
            spv: word;                         {skutečný počet vstupů neuronu}
      end;
      pneuron = ^neuron;

{pole, ve kterém jsou uloženy jak vstupy a výstupy, tak vzory}
{vstupy a výstupy jsou ve v[0][i], vzory v v[1..MPVZ][i]}
var   v: array [0..MPVZ] of array[0..MPVV] of double;
      n: array [1..PN] of pneuron;  {pole všech neuronů v síti}
      lambda: double;               {učící konstanta - určuje rychlost učení}
      spn: word;                    {skutečný počet neuronů}
      delkavzoru: word;             {jak je dlouhý řádek v souboru vzorů}
      spvz: word;                   {skutečný počet vzorů}

procedure vyhodnotneuron (a: word);
procedure nactivzory (s: string);
procedure naucneuron (a, b, c: word); {naučí a-tý neuron b-tý vzor metodou c}
procedure vyhodnotsit;
procedure vytvorneuron (a: word);
procedure vytvorsit (a, b, c: word); {vytvoří síť s a vstupy b skrytými neurony}
                                     {a c výstupními neurony}
procedure zrusneuron (a: word);
procedure zrussit;
procedure save (s: string);
procedure load (s: string);
function chybavzoru (b: word): double;
function chybasite: double;
procedure naucsit (c: longint);      {pokouší se c sekund učit síť}
function pocetchybvzoru (b: word): word;
```

```
function pocetchybsite: word;

implementation

procedure vyhodnotneuron (a: word);
var     s: double;
        i: word;
begin
   s:=0;
   with n[a]^ do
      begin
         for i:=0 to spv do
            begin
               s:=s+w[i]*v[0][x[i]];
            end;
         v[0][y]:=2/(1+exp(-s))-1;
      end;
end;

procedure vyhodnotsit;
var     i, j: word;
begin
   for i:=1 to 2 do
      begin
         for j:=1 to spn do
            begin
               if n[j]^.vrstva = i then
                  begin
                     vyhodnotneuron (j);
                  end;
            end;
      end;
end;


procedure copyvzor (b: word);
var     i: word;
begin
   for i:=0 to MPVV do v[0][i]:=v[b][i];
end;
```

```
procedure naucneuron (a, b, c: word);
var     p: double;
        i, j, k: word;
begin
   if c = 1 then
      begin
         copyvzor (b);
         vyhodnotsit;
         with n[a]^ do
            begin
               p:=lambda*(v[0][y]-v[b][y])*(1-v[0][y]*v[0][y]);
               for i:=0 to spv do
                  begin
                     w[i]:=w[i]-p*v[0][x[i]];
                  end;
            end;
      end;
   if c = 2 then
      begin
         copyvzor (b);
         vyhodnotsit;
         with n[a]^ do
            begin
               p:=lambda*(v[0][y]-v[b][y])*(1-v[0][y]*v[0][y]);
               for i:=0 to spv do
                  begin
                     w[i]:=w[i]-p*v[0][x[i]];
                     for j:=1 to spn do
                        begin
                           if n[j]^.y = x[i] then
                              begin
                                 for k:=1 to n[j]^.spv do
                                    begin
                                       n[j]^.w[k]:=n[j]^.w[k]-p*v[0][x[k]];
                                    end;
                              end;
                        end;
                  end;
            end;
      end;
end;
```

```
procedure nactivzory (s: string);
var     file1: text;
        i, j: word;
begin
   assign (file1, s);
   reset (file1);
   j:=0;
   while not (eof (file1)) do
      begin
         inc (j);
         readln (file1, s);
         v[j][0]:=1;
         delkavzoru:=length (s);
         for i:=1 to length (s) do
            begin
               if s[i] = '0' then v[j][i]:=-1;
               if s[i] = '1' then v[j][i]:=1;
            end;
         for i:=length (s)+1 to MPVV do v[j][i]:=v[j][i-1];
      end;
   close (file1);
   spvz:=j;
end;

procedure vytvorneuron (a: word);
begin
   new (n[a]);
   if a > spn then spn:=a;
end;

procedure zrusneuron (a: word);
begin
   dispose (n[a]);
end;

procedure zrussit;
var     i: word;
begin
   for i:=1 to spn do zrusneuron (i);
end;
```

```
procedure save (s: string);
var   file1: file of double;
      p: double;
      i,j: word;
begin
  assign (file1, s);
  rewrite (file1);
  p:=spn;
  write (file1, p);              {zapíše skutečný počet neuronů}
  for i:=1 to spn do
    begin
      with n[i]^ do
        begin
          p:=spv;
          write (file1, p);      {zapíše skutečný počet vstupů neuronu}
          p:=vrstva;
          write (file1, p);      {zapíše číslo vrstvy}
          p:=y;
          write (file1, p);
          for j:=0 to spv do
            begin
              p:=x[j];
              write (file1, p);
              write (file1, w[j]);
            end;
        end;
    end;
  close (file1);
end;

function chybavzoru (b: word): double;
var   i,j: word;
      ch: double;
begin
  ch:=0;
  copyvzor (b);
  vyhodnotsit;
  for i:=1 to delkavzoru do
    begin
      ch:=ch+0.5*sqr(v[b][i]-v[0][i]);
    end;
  chybavzoru:=ch;
end;
```

```
procedure load (s: string);
var     file1: file of double;
        p: double;
        i, j: word;
begin
   assign (file1, s);
   reset (file1);
   read (file1, p);              {přečte skutečný počet neuronů}
   spn:=round (p);
   for i:=1 to spn do
      begin
         vytvorneuron (i);
         with n[i]^ do
            begin
               read (file1, p);        {přečte skutečný počet vstupů neuronu}
               spv:=round (p);
               read (file1, p);              {přečte číslo vrstvy}
               vrstva:=round (p);
               read (file1, p);
               y:=round (p);
               for j:=0 to spv do
                  begin
                     read (file1, p);
                     x[j]:=round (p);
                     read (file1, w[j]);
                  end;
            end;
      end;
   close (file1);
end;

function pocetchybvzoru (b: word): word;
var     i, j: word;
        ch: word;
begin
   ch:=0;
   copyvzor (b);
   vyhodnotsit;
   for i:=1 to delkavzoru do
      begin
         if (0.5*sqr(v[b][i]-v[0][i])) >= 0.5 then inc (ch);
      end;
   pocetchybvzoru:=ch;
end;
```

```
procedure vytvorsit (a, b, c: word);
var   i, j: word;
begin
   for j:=1 to c do
      begin
         vytvorneuron (j);
         with n[j]^ do
            begin
               for i:=0 to a do
                  begin
                     x[i]:=i;
                     w[i]:=0;
                  end;
               for i:=a+1 to a+b do
                  begin
                     x[i]:=c+i;
                     w[i]:=0;
                  end;
               y:=a+j;
               vrstva:=2;
               spv:=a+b;
            end;
      end;
   for i:=1 to b do
      begin
         vytvorneuron (i+c);
         with n[i+c]^ do
            begin
               for j:=0 to a do
                  begin
                     x[j]:=j;
                     w[j]:=random (4);
                     w[j]:=w[j]-2;
                  end;
               y:=a+c+i;
               vrstva:=1;
               spv:=a;
            end;
      end;
   spn:=b+c;
end;
```

```
function chybasite: double;
var     ch: double;
        i: word;
begin
   ch:=0;
   for i:=1 to spvz do ch:=ch+chybavzoru (i);
    chybasite:=ch;
end;

function pocetchybsite: word;
var     ch: word;
        i: word;
begin
   ch:=0;
   for i:=1 to spvz do ch:=ch+pocetchybvzoru (i);
   pocetchybsite:=ch;
end;
```

```
procedure naucsit (c: longint);
var     cas1,cas2: longint;
        hod, min, sek, ssek: word;
        konec: boolean;
        i, j: longint;
        r: word;
        minimum, ch: double;
begin
   gettime (hod, min, sek, ssek);
   cas1:=sek+min*60+hod*3600;
   konec:=false;
   minimum:=1000;
   lambda:=0.0001*random (100);
   repeat
      for i:=1 to 10000 do
         begin
            r:=random (spvz)+1;
            for j:=1 to spn do
               begin
                  if n[j]^.vrstva = 2 then naucneuron (j, r, 2);
               end;
         end;
      ch:=chybasite;
      if ch < minimum then
         begin
            minimum:=ch;
            save ('POM.TMP');
         end
                        else
         begin
            zrussit;
            load ('POM.TMP');
         end;
      writeln (pocetchybsite, ' ', minimum:6:6, ' ', ch:6:6);
      if pocetchybsite = 0 then konec:=true;
      gettime (hod, min,sek, ssek);
      cas2:=sek+min*60+hod*3600;
      if cas2 < cas1 then cas2:=cas2+86400;
      if (cas2-cas1) > c then konec:=true;
   until konec;
end;
begin
   lambda:=0.0001;
   spn:=0;
end.
```

Příloha B: Příklad použití jednotky Sit

```
program Hledani_nejmensi_site;
uses crt, Sit;                    {budou se používat jednotky crt a sit}
var   PNSV: word;                 {počet neuronů skryté vrstvy}
begin
  clrscr;                         {vymaže obrazovku}
  PNSV:=11;                       {nastaví počet neuronů ve skryté vrstvě na maximum}
  repeat
    sound (1000);                 {zvukové upozornění, že začíná další pokus}
    writeln ('Zkouším ', PNSV,' neuronů ve skryté vrstvě.');
    delay (1000);                 {doba trvání pípnutí}
    nosound;                      {ticho!}
    nactivzory ('PRV.DAT');       {načte z disku trénovací vzory}
    vytvorsit (5, PNSV, 1);       {vytvoří síť s pěti vstupy, PNSV skrytými}
                                  {neurony a jedním výstupem}
    naucsit (120);                {120 sekund se pokouší učit síť}
    if pocetchybsite = 0 then     {pokud se učení podařilo bezchybně}
      begin
        dec (PNSV);               {sníží počet neuronů ve skryté vrstvě}
      end;
    zrussit;                      {zruší síť}
  until PNSV = 1;                 {pokud by se mu podařilo najít síť s jedním skrytým}
                                  {neuronem, tak skončí}
end.
```

Příloha C: Trénovací vzory

Odpovědi na otázky ohledně výsledků na střední škole. Jednička znamená odpověď ANO, nula znamená odpověď NE.

Sloupec A:
1. číslice - Jste muž?
2. číslice - Jste žena?

Sloupec B: Otázky týkající se jazyků včetně češtiny.
1. Dostal(a) jste někdy na vysvědčení z těchto předmětů jedničku?
2. ... dvojku?
3. ... trojku?
4. ... čtyřku?
5. ... pětku?
6. Bylo rozpětí vašich známek jedna (od nejlepší k nejhorší)?
7. ... dvě?
8. ... tři nebo více?

Sloupec C: Tatáž osmice otázek týkajících se tentokráte matematiky a fyziky.

Sloupec D: Tatáž osmice otázek týkajících se tentokráte biologie, zeměpisu a dějepisu.

Sloupec E:
1. Byly jazyky včetně češtiny vaše nejoblíbenější předměty?
2. Byly matematika a fyzika vaše nejoblíbenější předměty?
3. Byly vaše nejoblíbenější předměty biologie, zeměpis a dějepis?
4. Kladli jste na druhé místo své oblíbenosti jazyky včetně češtiny?
5. Kladli jste na druhé místo své oblíbenosti matematiku a fyziku?
6. Kladli jste na druhé místo své oblíbenosti biologii, zeměpis a dějepis?
7. Byly jazyky včetně češtiny vašimi nejméně oblíbenými předměty?
8. Byly matematika a fyzika vašimi nejméně oblíbenými předměty?
9. Byly biologie, zeměpis a dějepis vašimi nejméně oblíbenými předměty?

Sloupec F:
1. Dostali jste se na práva?
2. Nedostali jste se na práva?
3. Dostali jste se na medicínu?

4. Nedostali jste se na medicínu?
5. Dostali jste se na pedagogickou fakultu?
6. Nedostali jste se na pedagogickou fakultu?
7. Dostali jste se na techniku?
8. Nedostali jste se na techniku?
9. Dostali jste se na školu s ekonomickým zaměřením?
10. Nedostali jste se na školu s ekonomickým zaměřením?

A	B	C	D	E	F
10	11000010	11000010	11000010	010001100	0000001000
10	11000010	11100001	11000010	001100010	1100000000
01	11100001	11100001	11000010	000000000	0000110000
10	01110001	10000100	01100010	010001100	0000001000
01	01000100	01000100	10000100	001100010	0000001000
01	11100001	11100001	10000100	001010100	0000001000
01	00100100	01000100	01000100	001010100	0000001000
01	10000100	01000100	01000100	100001010	0000001000
01	11000010	11000010	11000010	100001010	0000001001
01	01000100	01100010	11000010	001100010	0000001000
01	01100010	00110010	11100001	001100010	0000011000
10	11100001	11100001	11000010	010001100	0000001000
01	11100001	01110001	11000010	100001010	0100001000
01	11000010	11000010	11000010	100001010	0000000011
01	01100010	00110010	00110010	100001010	0000100000
10	00110010	10000100	11100001	010001100	0000001000
01	01100010	01100010	01100010	100001010	0000000001
01	01000100	01100010	11000010	001100010	0000001000
01	11000010	01100010	11000010	100010001	0000000010
01	01100010	11100001	11000010	001010100	0000001000
10	11100001	11000010	11000010	001010100	0010001000
10	11000010	11000010	11000010	010100001	1000000010
01	01110001	11100001	11100001	001010100	0000011000
01	11000010	11000010	10000100	010100001	0000110000
01	11000010	01000100	10000100	100001010	0100001000
01	11100001	01000100	11000010	010001100	0000001000
10	11000010	01110001	01110001	010100001	0000100000
01	11000010	01100010	11000010	100001010	0100011000
01	11100001	11000010	11000010	010100001	0000101000
01	11000010	11000010	11000010	010001100	0000001000
01	10000100	11000010	10000100	100001010	0000001000
10	10000100	11000010	11100001	100001010	0100001000

01	11000010	11000010	11000010	100001010	0000110000
01	11000010	11000010	11000010	001100010	0000000011
01	01110001	11100001	11100001	010001100	0000001001
10	01110001	11100001	11000010	001010100	0100001001
01	11100001	01100010	11000010	100001010	0100011001
01	11000010	11100001	11000010	100010001	0000000011
01	11100001	01110001	11100001	100001010	0000101001
01	11100001	01100010	11000010	001100010	1000000000
10	00110010	01100010	11100001	010001100	0000001000
10	11100001	01100010	11000010	001100010	0000000010
10	10000100	11000010	10000100	100001010	0000000010
01	11100001	00100100	11000010	001100010	0000110000
10	11100001	01100010	11000010	001010100	0000001000
10	01100010	01100010	11100001	001100010	0000000000
10	01110001	01110001	11110001	100010001	0000001100
10	00110010	00110010	01100010	001010100	0000001100
10	11000010	01100010	11000010	000000000	0011000000
10	11000010	11000010	11000010	100010001	1000000010
01	11100001	00100100	11100001	100001010	0000000000
01	01100010	00100100	11100001	001100010	0000000000
01	11000010	01000100	11000010	001100010	0100000000
10	11100001	01100010	11100001	100010001	0000101010
01	11100001	01100010	11000010	001100010	0000000000
01	11100001	01100010	11000010	001100010	0000100000
01	11100001	01100010	11000010	001100010	0000001000
01	11100001	01100010	11000010	001100010	0000010000
01	11000010	01100010	11000010	001100010	0010000000
01	11000010	11000010	11000010	001100010	0010000000
01	11000010	01100010	11000010	100001010	0001100000
01	11100001	01100010	11000010	001100010	0000000000
10	01100010	01100010	11100001	001010100	0000110000
10	01110001	01110001	11100001	010001100	0000000100
10	11000010	11000010	11000010	010100001	0000001000
01	11000010	01100010	11000010	100001010	0000100000
01	01100010	01100010	11000010	001100010	0000000000
01	10000100	11000010	10000100	100001010	0000000010
01	01100010	01110001	11100001	001100010	0000000000
10	11000010	11100001	11000010	010001100	0000001000
01	11100001	11100001	11000010	000000000	0000110000
10	01110001	10000100	01100010	010001100	0000001000
01	01000100	01000100	10000100	001100010	0000001000
01	11000010	01000100	10000100	100001010	0100001000
01	11100001	01000100	11000010	010001100	0000001000
10	11000010	01110001	01110001	010100001	0000100000
01	11000010	01100010	11000010	100001010	0100011000
01	11100001	11000010	11000010	010100001	0000101000

01	11000010	11000010	11000010	010001100	0000001000
01	11000010	01100010	11000010	100010001	0000000010
01	01100010	11100001	11000010	001010100	0000001000
10	11100001	11000010	11000010	001010100	0010001000
01	10000100	01000100	01000100	100001010	0000001000
01	11000010	11000010	11000010	100001010	0000001001
01	01000100	01100010	11000010	001100010	0000001000
01	11100001	01100010	11000010	001100010	0000010000
01	11000010	01100010	11000010	001100010	0010000000
01	11000010	11000010	11000010	001100010	0010000000
01	01110001	11100001	11100001	001010100	0000011000
01	11000010	11000010	10000100	010100001	0000110000
01	11000010	01000100	10000100	100001010	0100001000
10	11000010	11000010	11000010	010001100	0000001000
10	11000010	11100001	11000010	001100010	1100000000
01	11100001	11100001	11000010	000000000	0000110000
10	11100001	11000010	11000010	001010100	0010001000
10	11000010	11000010	11000010	010100001	1000000010
01	01110001	11100001	11100001	001010100	0000011000
01	11100001	01000100	11000010	010001100	0000001000
10	11000010	01110001	01110001	010100001	0000100000
01	11000010	01100010	11000010	100001010	0100011000
01	11100001	11000010	11000010	010100001	0000101000
01	11000010	11000010	11000010	010001100	0000001000

Literatura

[1] Jiří Šíma, Roman Neruda: Teoretické otázky neuronových sítí.
Matfyzpress, Praha 1996

[2] Mirko Novák, Josef Faber, Olga Kufudaki: Neuronové sítě a informační systémy živých organismů.
Grada, Praha 1993

[3] Mirko Novák: Umělé neuronové sítě a neuropočítače odkud jdou a kam?
PC WORD 2: 48-57, 1991

[4] Martin Kvoch: Programování v Turbo Pascalu 7.0.
Kopp, České Budějovice 1995

Rejstřík

A

AND, 14

axon, 4

B

BACKPROPAGATION, 21

C

citlivost neuronu, 14

D

delta metoda, 18

dendrit, 4

double, 22

E

EU, 30

F

funkce

- přenosová, 19
- převodní, 12
- sigmoidní, 7

CH

chyba

- hrubá, 35
- neuronu, 21

K

koprocesor, 22

L

lineární separace. *viz* separace lineární

M

mozek, 5

N

neuron

- biologický, 4
- formální, 11

neuropočítač, 7

O

odchylka

- absolutní, 21
- kvadratická, 21

optimální konfigurace, 32

OR, 14; 27

P

perceptron, 6; 12
prvočísla, 26

R

real, 22
rozpoznávání znaků, 7; 8

S

separace lineární, 14; 21
sigmoida. *viz* funkce sigmoidní
síť
  cyklická, 16
  jednovrstvá, 20; 26
  neuronová, 6
  vícevrstvá, 26
  vrstevná, 16
soma, 4
sumátor, 11
synapse, 4

T

tangens hyperbolický, 14; 22

U

učení
  Hebbovo, 18
  soutěživé, 18
učící konstanta, 20
umělá inteligence, 7

V

váhy vstupů, 11; 20
vrstva
  skrytá, 16
  vstupní, 16
  výstupní, 16
vyhodnocení neuronu, 19
vzory trénovací, 14; 18

X

XOR, 7; 15