Juraj Holas

JEDNODUCHÝ PERCEPTRÓN

PROJEKT Č.1, NEURÓNOVÉ SIETE, 2014/2015

ÚVOD

Úlohou projektu bolo ukázať, že spojitý perceptrón dokáže implementovať logické funkcie AND a OR pre ľubovoľné n-rozmerné vstupy. Ďalej sme skúmali vplyv zvolenej rýchlosti učenia na dobu konvergencie perceptrónu na oboch uvažovaných funkciách, a tiež porovnanie efektivity spojitého a diskrétneho perceptrónu na tejto úlohe.

IMPLEMENTÁCIA

Úloha je implementovaná v jazyku Java, pričom všetky zdrojové súbory sú dostupné v prílohe. Samotný perceptrón vrátane trénovania príkladu a celej epochy je implementovaný v samostatnej (abstraktnej) triede, od ktorej sú následne odvodené triedy samostatne pre spojitý a diskrétny perceptrón, ktoré už obsahujú iba ich špecifické funkcie.

V hlavnej triede je vyriešené generácia vstupných dát a testovanie jednotlivých častí projektu. Na jej začiatku nájdeme inicializáciu dôležitých parametrov pre testovanie, najmä `N` – rozmer vstupných dát a `MAX_EPOCH_NUM` – maximálny počet epoch pri trénovaní, po prekročení ktorého už uvažujeme že trénovací proces divergoval. Ďalšie z týchto parametrov budú popísané v jednotlivých častiach projektu.

POUŽITÉ METÓDY

Pri implementácii perceptrónu existuje viacero možností pre výber kľúčových funkcií a metód. Táto časť sa zaoberá výberom daných metód pre tento projekt.

Chybová funkcia: zvolená bola štandardná funkcia pre vyhodnocovanie chyby – štvorec rozdielu od požadovanej hodnoty: $e(y) = \frac{1}{2}(d - y)^2$. Z dôvodu správnosti ďalších vzorcov je hodnota škálovaná na polovicu, čo však nemá vplyv na našu úlohu minimalizácie chyby. Nakoľko ide o klasifikačnú úlohu, tak výstup zo spojitého perceptrónu pri posudzovaní klasifikačnej chyby je zaokrúhlený – interpretovaný na tú kategóriu, ku ktorej je bližšie.

Aktivačná funkcia: nakoľko požadovaný výstup perceptrónu je 0 alebo 1, volili sme unipolárne aktivačné funkcie, ktoré nadobúdajú hodnoty práve v tomto rozsahu:

- Spojitý perceptrón: sigmoid → $f(net) = 1/(1 + \exp(net - \theta))$
- Diskrétny perceptrón: prahová funkcia → $f(net) = \begin{cases} 1, & net > \theta \\ 0, & net \leq \theta \end{cases}$

Tieto dve funkcie sme zvolili aj z toho dôvodu, že ich priebeh je vo veľkej miere podobný. Vďaka tomu môžeme získať vierohodnejšie výsledky pri porovnávaní oboch typov perceptrónu. Pre zjednodušenie algoritmu sme tiež včlenili prah θ do vektora váh tým, že sme vstup rozšírili o konštantný prvok s hodnotou -1 – *bias*.

Učiace pravidlo: učenie perceptrónu, teda úprava jeho váh w_i pre jednotlivé vstupy x_i , prebiehalo podľa štandardného pravidla $\Delta w_i = \alpha . \delta . x_i$, kde α je zvolená rýchlosť učenia a δ je

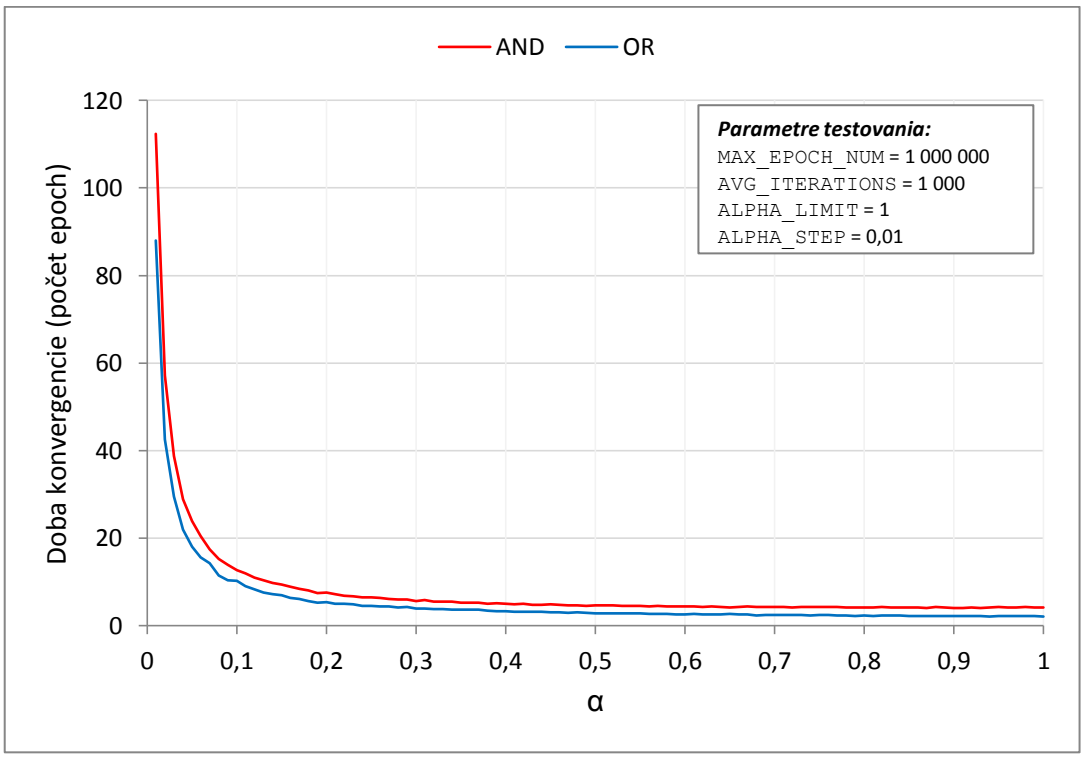
miera chyby. Nakoľko používame metódu *gradient descent*, δ je určená použitou aktivačnou funkciou. Implementované sú teda varianty:

- Spojitý perceptrón: $\boldsymbol{\delta} = (d - y).f'(net) = (d - y).f(net).\big(1 - f(net)\big) = (\boldsymbol{d} - \boldsymbol{y}).\boldsymbol{y}.(\boldsymbol{1} - \boldsymbol{y})$
- Diskrétny perceptrón: $\boldsymbol{\delta} = (\boldsymbol{d} - \boldsymbol{y})$

Trénovanie: proces trénovania perceptrónu prebiehal podľa štandardnej šablóny – v každej epoche dostal perceptrón na naučenie celú sadu trénovacích dát, pričom tento proces sa iteratívne opakoval až pokým klasifikačná chyba epochy neklesla na nulu. Na začiatku každej epochy sa tiež náhodne pomiešalo poradie trénovacích príkladov, aby sa zaručilo rovnomerné učenie. Zaznamenávala sa vždy doba konvergencie – počet epoch, po ktorých už perceptrón správne klasifikoval každý vstupný príklad. Pre prípad divergentného učenia bol proces prerušený, ak počet epoch prekročil nastavený `MAX_EPOCH_NUM`. Každé trénovanie perceptrónu predchádzala inicializácia jeho váh na náhodné hodnoty.

A) Vplyv rýchlosti učenia na dobu konvergencie

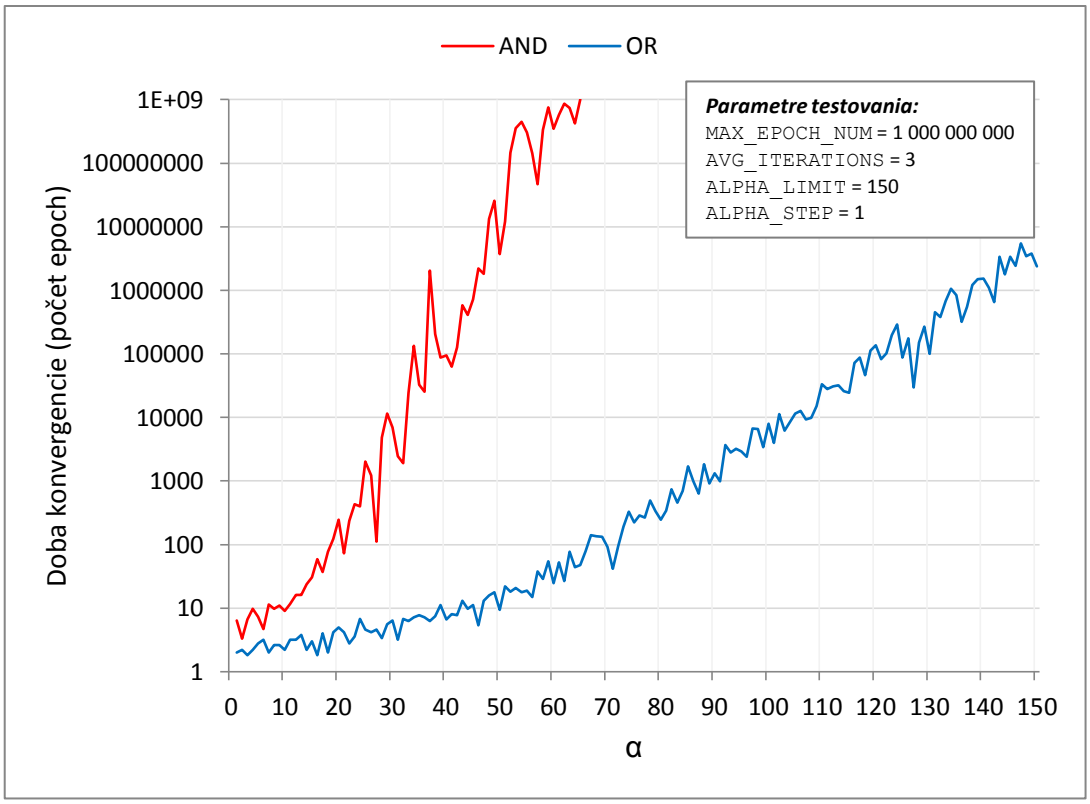
Pre obe trénované funkcie sme sledovali počet epoch, po ktorých už náš spojitý perceptrón správne klasifikoval všetky vstupy, pri rôznych nastaveniach rýchlosti učenia α. Pri každom teste sa bral do úvahy priemer doby konvergencie zo sady nezávislých učení, aby sa dosiahol presnejší a vierohodnejší výsledok. Počet týchto učení určoval parameter `AVG_ITERATIONS`, ktorý bol pri bežných testoch nastavený na 1000. Opakovaním každého učenia takto vysoký počet krát sa zabezpečilo, že sme dosiahli výsledky vo veľkej miere „očistené“ od faktoru náhody, a štatisticky sme odfiltrovali nežiaduce prípady, ako napr. keď sa váhy náhodne inicializovali priamo na správne riešenie. Všetky testy prebiehali na trojrozmerných vstupoch. Testované hodnoty pre α boli vyberané z intervalu $(0, limit\rangle$, kde *limit* bol určený parametrom `ALPHA_LIMIT`. Hodnoty boli vyberané rovnomerne po krokoch s veľkosťou `ALPHA_STEP`. Nasledujúce grafy prezentujú získané výsledky:



Graf 1 – doba konvergencie pre malé α

Na grafe 1 vidíme, že malá rýchlosť učenia očakávane predlžuje čas naučenia siete. So zvyšujúcou sa hodnotou α sa doba konvergencie znižuje po určitú hranicu: 2-4 epochy pre OR a 4-6 epoch pre AND. Tieto hodnoty sa dosiahnu zhruba pri hodnotách $\alpha > 0{,}3$, a pretrvávajú po celý zvyšok testovania. Dá sa teda usúdiť, že toto je minimálny priemerný počet epoch, ktorý potrebujeme na natrénovanie nášho perceptrónu na danú úlohu. Môžeme si tiež všimnúť, že trénovanie funkcie AND trvalo vždy o niečo dlhšie ako trénovanie funkcie OR. Spôsobené je to pravdepodobne tým, že spočiatku učenia AND sa sieť častejšie mýlila na vstupoch obsahujúcich aspoň jednu nulu. Tieto nulové vstupy ale neprispievajú ku zmene váh počas učenia. Možným riešením pri ďalších testovaniach by mohlo byť predspracovanie vstupu, kde by sa 0 (teda logické *falsie*) zmenila na -1, čím by už ku učeniu prispievala nenulovou čiastkou.

Nakoľko doba konvergencie klesala s rastúcou rýchlosťou učenia, zaujímalo nás tiež, ako sa bude tento vzťah vyvíjať pri väčších hodnotách α. Vo všeobecnosti sú hodnoty $\alpha > 1$ (príp. už menšie hodnoty blízke 1) pre učenie nevhodné, chceli sme teda overiť, či sa tento trend aplikuje aj v našom prípade. V ďalšom testovaní sme teda skúšali vysoké, priam až prehnané hodnoty α :



Graf 2 – doba konvergencie pre veľké α

Na začiatok si všimnime, že zvislá os grafu je v logaritmickej mierke. To značí, že rovnomerný rast v grafe predstavuje vlastne exponenciálny vývoj doby konvergencie, pokiaľ sa α zvyšuje do nezmyselne vysokých hodnôt. Z toho jasne vidíme, že pre učiaci proces je nanajvýš nevhodné nastavovať $\alpha \gg 1$ .To iba potvrdzuje naše znalosti nadobudnuté z teoretických základov problému.

B) Vývoj perceptrónu počas učenia

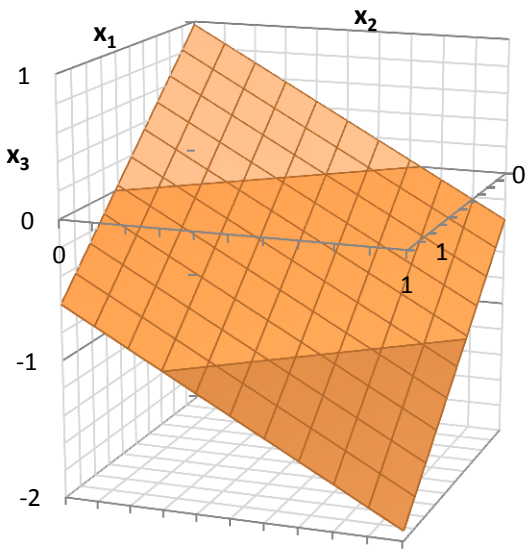
Pre demonštráciu bolo vybrané učenie funkcie OR na trojrozmernom vstupe. Zvolená bola rýchlosť učenia $\alpha = 0{,}2$, pri ktorej vieme, podľa predchádzajúcich pokusov, že priemerná

rýchlosť konvergencie je 5,396 epoch. Pri demonštrácií sa perceptrón naučil za 5 epoch, čo predstavuje pekný priemerný prípad pri zvolenej hodnote α.

Tabuľka 1 zachytáva priebeh učenia. Trénovacie príklady sú v takom poradí, v akom boli pri učení spracovávané (môžeme si všimnúť, že v každej epoche je toto poradie iné), pričom zvýraznené sú zle klasifikované prípady. Pre každú epochu je zobrazený aj vektor váh rozšírený o θ, teda váhu pre *bias* zložku. Graf 3 potom znázorňuje výslednú deliacu nadrovinu.

Epocha	x_1	x_2	x_3	d	y	$\vec{w}$ na konci epochy
1.	1	1	0	1	1	$\begin{bmatrix}0{,}394\\0{,}308\\0{,}172\\0{,}348\end{bmatrix}$
	0	0	0	0	0	
	0	1	1	1	0	
	1	0	1	1	1	
	0	1	0	1	0	
	1	0	0	1	1	
	1	1	1	1	1	
	0	0	1	1	0	
2.	0	0	1	1	0	$\begin{bmatrix}0{,}394\\0{,}333\\0{,}199\\0{,}296\end{bmatrix}$
	1	0	0	1	1	
	0	0	0	0	0	
	0	1	0	1	0	
	1	0	1	1	1	
	1	1	0	1	1	
	0	1	1	1	1	
	1	1	1	1	1	
3.	1	1	0	1	1	$\begin{bmatrix}0{,}394\\0{,}333\\0{,}225\\0{,}270\end{bmatrix}$
	0	0	0	0	0	
	1	0	0	1	1	
	0	0	1	1	0	
	0	1	0	1	1	
	0	1	1	1	1	
	1	1	1	1	1	
	1	0	1	1	1	
4.	0	1	1	1	1	$\begin{bmatrix}0{,}394\\0{,}333\\0{,}251\\0{,}244\end{bmatrix}$
	1	0	1	1	1	
	0	0	1	1	0	
	1	1	0	1	1	
	1	0	0	1	1	
	0	0	0	0	0	
	0	1	0	1	1	
	1	1	1	1	1	
5.	0	0	1	1	1	$\begin{bmatrix}0{,}394\\0{,}333\\0{,}251\\0{,}244\end{bmatrix}$
	0	1	1	1	1	
	0	0	0	0	0	
	0	1	0	1	1	
	1	0	1	1	1	
	1	1	0	1	1	
	1	1	1	1	1	
	1	0	0	1	1	

Tabuľka 1 – priebeh učenia perceptrónu



Graf 3 – výsledná deliaca nadrovina

Priesečníky deliacej nadroviny so súradnicovými osami sú v bodoch:

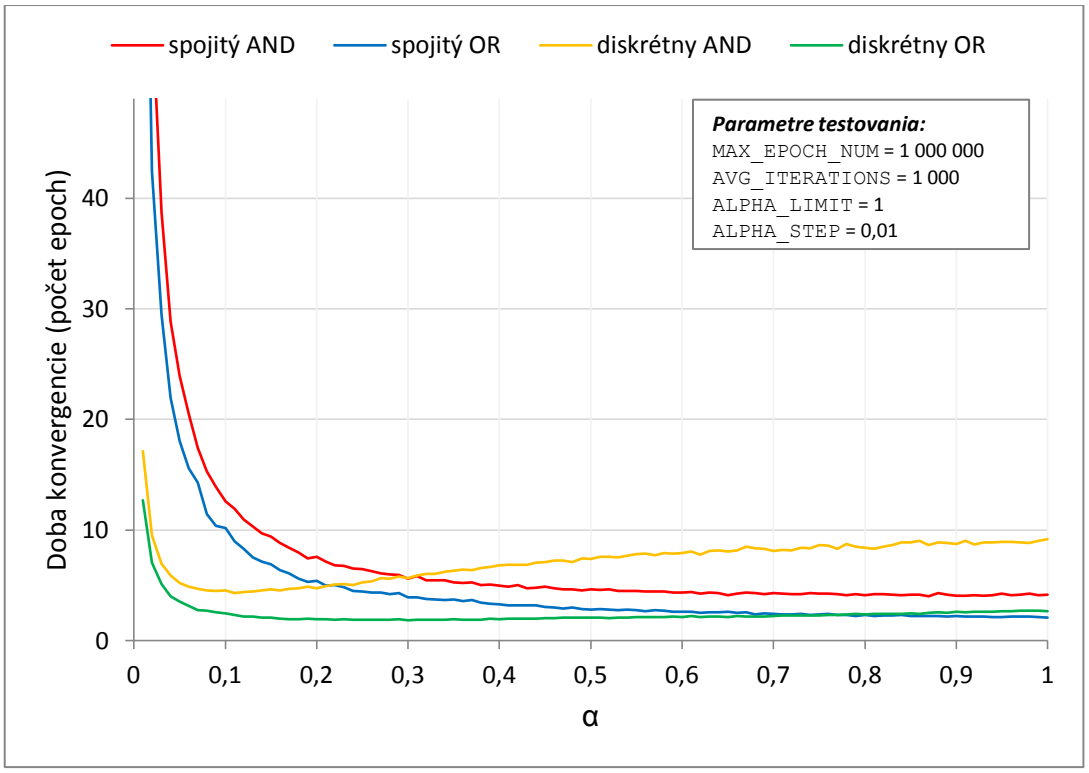
$$p_1 = [0.619, 0, 0]$$
$$p_2 = [0, 0.734, 0]$$
$$p_3 = [0, 0, 0.976]$$

C) Porovnanie s diskrétnym perceptrónom

Otázkou zadania bolo, či by sa zmenil postup, vyhodnocovanie, či správanie modelu, ak by sme na danú úlohu použili namiesto spojitého diskrétny perceptrón. V postupe by sa zmenili iba dve zásadné veci: aktivačná funkcia, a δ-funkcia použitá v učiacom pravidle. Obe funkcie sú popísané aj pre diskrétny perceptrón v časti *Použité metódy*. Ich rozdiel je, pochopiteľne, najmä v tom, že funkcie pre diskrétny perceptrón fungujú nad celými číslami, zatiaľ čo tie pre spojitý

nad reálnymi. Ich ostatné vlastnosti sú však značne podobné. Veľká zmena by nenastala ani pri vyhodnocovaní výsledkov – pri spojitom perceptróne bolo potrebné výsledok zaokrúhliť (interpretovať na jednu z kategórií), pri diskrétnom toto nie je potrebné.

Zaujímavejšou otázkou je porovnanie správania sa oboch modelov. Zopakovali sme teda znova test z časti A), tentoraz však aj pre diskrétny perceptrón. Graf 4 znázorňuje nové hodnoty:



Graf 4 - doba konvergencie spojitého a diskrétneho perceptrónu

Z grafu jasne vidíme, že pre malé hodnoty α (zhruba do 0,1) je diskrétny perceptrón podstatne efektívnejší než spojitý. So zvyšujúcou sa rýchlosťou učenia je potom doba konvergencie oboch modelov porovnateľná, pri funkcii AND sa dokonca diskrétny perceptrón správa o niečo pomalšie. Vysvetlením pre výrazný rozdiel pri nízkych rýchlostiach učenia by mohlo byť porovnanie δ-funkcií oboch modelov:

$$\delta_d = (d - y)$$
$$\delta_s = (d - y).y.(1 - y)$$

V oboch prípadoch vystupuje člen $(d - y)$, ktorý, bez ohľadu na to, či sa jedná o diskrétne hodnoty v δ_d alebo spojité v δ_s, nadobúda hodnoty z intervalu $\langle -1, 1\rangle$. Avšak v δ_s ďalej vystupuje aj člen $g(y) = y.(1 - y)$. Preskúmaním vlastností tejto funkcie na intervale $\langle 0, 1\rangle$ zistíme, že svoje maximum nadobúda v bode $y = 0{,}5$ s hodnotou $g(0{,}5) = 0{,}25$. Na zvyšných hodnotách uvažovaného intervalu dosahuje menšie kladné hodnoty. Týmto číslom z intervalu $\langle 0, 1/4\rangle$ sa následne prenásobí prvý člen $(d - y)$. To znamená, že pre všeobecný kladný výstup perceptrónu platí, že $\delta_d \geq 4\delta_s$ (pre záporné s opačným znamienkom). Diskrétny perceptrón sa teda v našom prípade učí zhruba 4-krát rýchlejšie ako spojitý. Pri bližšom pohľade na graf sa nám táto hypotéza potvrdí – vidíme, že diskrétny perceptrón nadobúda svoje minimá okolo hodnoty $\alpha = 0{,}13$, kým spojitý až okolo $\alpha = 0{,}5$, čo činí zhruba 4-násobný rozdiel.