Porovnanie klasifikačných metód pre trojvrstvovú neurónovú sieť

Jakub Bahyl[1]
FMFI, UK v Bratislave

V tejto práci prezentujem základné optimalizačné algoritmy aplikované na umelý dataset, v ktorom sa neurónová sieť učí klasifikovať tri triedy. Spomedzi všetkých metód vyberám jednu, najlepšiu, s ktorou po 300 epochách dosahuje klasifikačnú chybu 1.81 % na validačnej vzorke a 2.20 % na testovnej.

Keywords: Neurónová sieť, klasifikácia

"I think the brain is essentially a computer and consciousness is like a computer program. It will cease to run when the computer is turned off. Theoretically, it could be re-created on a neural network, but that would be very difficult, as it would require all one's memories."
Stephen Hawking

I. INTRO

Celá programovacia časť bola tvorená v pomerne nedávno vytvorenom jazyku Julia v0.5.1. Prílohou k tomuto projektu je teda zdrojový kód, ktorý je navyše spustiteľný bez akýchkoľvek inštalácií cez Juliabox. Prihlásiť sa tam dá cez Google+, Github alebo LinkedIn a uploadnúť kód, ďalej sa funguje na Jupyteri.

V kóde využívam knižnice *PyPlot* (od Pythonu) na vizualizáciu dát a *DataFrames* (od Rka) na import dát. V kóde sa nachádza množstvo headerov a komentárov, takže by malo byť zrejmé, čo sa kde deje.

V projekte najskôr ukazujem, ako predpripravujem dáta, následne nastavujem základné fixné parametre, testujem rôzne modely a na konci prezentujem ten najlepší. V celej práci sa obmedzujem na použitie jednej skrytej vrstvy pre neuróny.

II. IMPORT & PRE-PROCESSING

Prvý pohľad na surové dáta vyzerá takto:

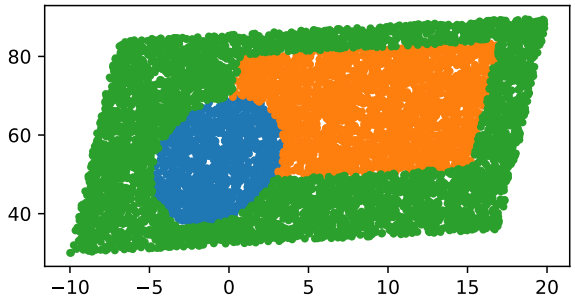

FIG. 1. Dátové body sú zobrazené guličkou, pričom 3 farby rozlišujú 3 triedy.

Trénovacie dáta (*full_set*) načítavam ako DataFrame a konvertujem na typ Array, aby sa s nimi dalo pracovať v maticovom zmysle. Keďže triedy sú pomenované ako A, B, C, preradím ich na $\{A, B, C\} \to 1, 2, 3$, vďaka čomu máme *full_set* ako Float pole.

Škálovanie a dekorelácia

Pre optimálnu a jednoduchú inicializáciu váhových matíc v (neskoršej časti) zoškálujeme dáta tak, aby sa nachádzali vo štvorci $\forall x, y \in \langle -0.5, 0.5 \rangle$. Všimnime si ešte, že dáta sú "strihnuté" do akéhosi kosoštvorca. Podľa sklonov strán som vytvoril lineárnu transformáciu na štvorec a teda po dekorelácii dostávame:

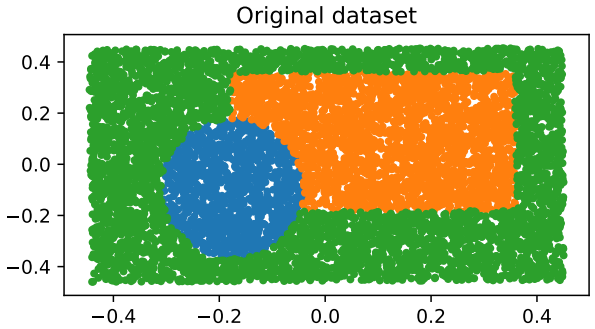

FIG. 2. Dátové body upratané do štvorca.

Teraz si možno všimnúť, že priestor troch tried je postupne väčší štvorec, menší štvorec a elipsa.

III. NEURÓNOVÁ SIEŤ

Výsledky prezentované v tejto práci sú výsledkom neurónovej siete, ktorá má na vstupnej vrstve 1 dátový bod (x_i, y_i), na jednej jedinej skrytej vrstve *dim_hid* (o veľkosti tejto vrstvy píšem neskôr) neurónov a na výstupnej vrstve trojzložkový vektor, ktorého každá zložka reprezentuje jednu triedu, kam sa neurónová sieť snaží zaklasifikovať každý dátový bod, ktorý na vstupe dostane. Keď si domyslíme bias units, tak naša sieť má architektúru 3:*dim_hid*:3.

Inicializácia váh a propagácia

Transformácia medzi vrstvami prebieha lineárnym zobrazením váhovými maticami $W_{\text{in}}^{\text{dim\_hid}\times(2+1)}$, $W_{\text{out}}^{3\times(\text{dim\_hid}+1)}$ a aktivovaním jednou z aktivačných

funkcií {sigmoid, tanh}. Váhové matice generujem ako rovnomerné náhodné čísla v intervale $(-\epsilon, \epsilon)$, pričom ϵ je zvolený ako $\epsilon = 0.01$.

Tréning a validácia siete

Algoritmus, akým učíme a validujeme neurónovú sieť, sa dá popísať takto:

- Zamiešame *full_set* a rozdelíme ho na *train_set* a *val_set* v pomere 7:1. Pri vyhodnocovaní modelov použijeme aj k-fold cross validation, no zatiaľ to netreba.
- Inicializujeme váhové matice $W_{\text{in}}, W_{\text{out}}$.
- Vyššie popísanou propagáciou vypočítame predpokladanú klasifikáciu sieťou pre daný dátový bod.
- Na základe definície chyby siete a jej derivácie aplikujeme back-propagáciu. (závisí od modelu)
- Opakujeme pre optimálny počet cyklov = epoch.
- Na konci vyhodnotíme sieť na validačnej vzorke.

V ďalších častiach píšem o tom, ako som sa rozhodol nastaviť parametre: počet neuróny, dĺžka krok λ, počet epoch. Kým nepovieme inak, používať budeme štandardný back-prop algoritmus a vyhodnocovanie chyby siete formulou:

$$\text{Error} = \frac{1}{N}\sum_i (y_i - d_i)^2 \,,$$

kde $\{y\}$ sú výsledky podľa siete a $\{d\}$ skutočné triedy.

Minimálny počet neurónov dim_hid

Predtým, než ukážem výsledky pre väčšie počty neurónov, chcem demonštrovať, prečo jeden ani dva neuróny principiálne nemôžu stačiť na rozlíšenie troch tried, pričom dve z nich sú obkolesené treťou. Jeden neurón nedokáže nič iné, než len jednou priamkou rozdeliť štvorec a klasifikovať dve najpočetnejšie triedy (FIG. 3). 2 neuróny toho tiež veľa nezmôžu, pretože tie vedia maximálne spojiť dve priamky do tvaru "U" a teda opäť klasifikovať len dve triedy (FIG. 4). 3 neuróny už vedia vygenerovať trojuholník a teda uzavrieť napríklad modrú oblasť. (FIG. 5) Táto sieť už vie v akejsi úbohej miere klasifikovať aj tri triedy. Pre takých 30 neurónov už vizualizácia vyzerá dramaticky lepšie. (FIG. 6)

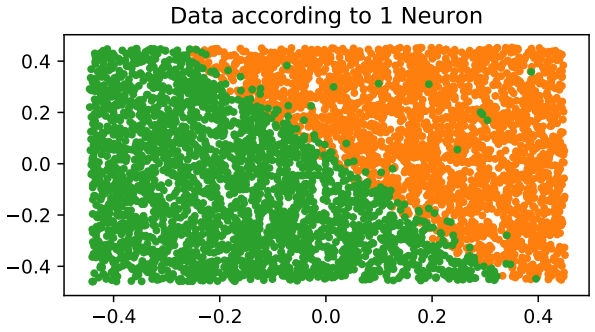

FIG. 3.

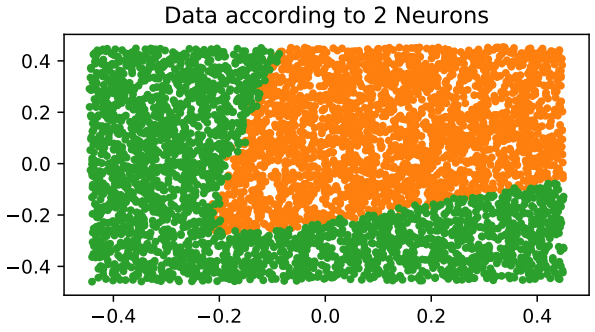

FIG. 4.

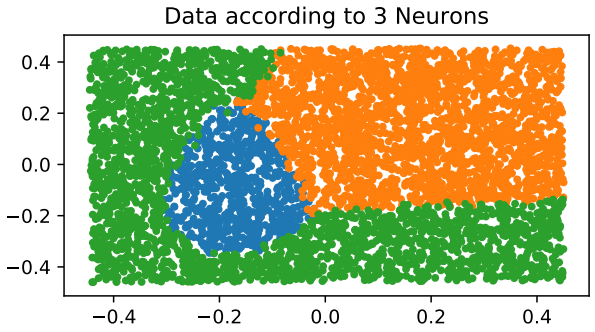

FIG. 5.

Pre modely som chcel použiť optimálny počet neurónov, ktoré sú jednak schopné problém klasifikovať a dvak nespomaľujú príliš celý proces. Na nasledujúcom grafe (FIG. 7) je graf konečnej kvadratickej chyby pre neurónovú sieť s rôznymi počtami neurónov v skrytej vrstve. Počet epoch bol schválne veľký (1000) a krok schválne malý (0.01), tieto parametre taktiež zoptimalizujeme v ďalšej časti.

Na grafe vidíme, že od určitého množstva neurónov sa chyba klasifikácie už výrazne nemení. Pre ďalšie výpočty vo zvyšku práce teda volím množstvo dim_hid=30.

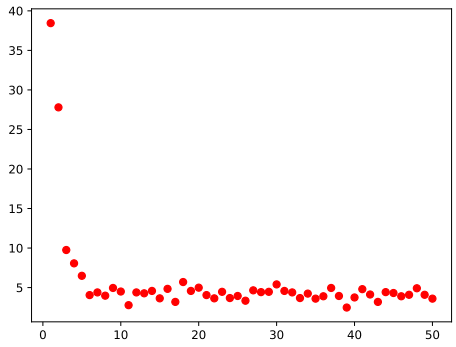

FIG. 6. Chyba klasifikácie pre rôzne počty neurónov.

Záverom tejto podkapitoly uvádzam vizualizáciu pre 30 neurónov, na prvý pohľad sa dá povedať, že ide o pomerne dobrý výsledok:

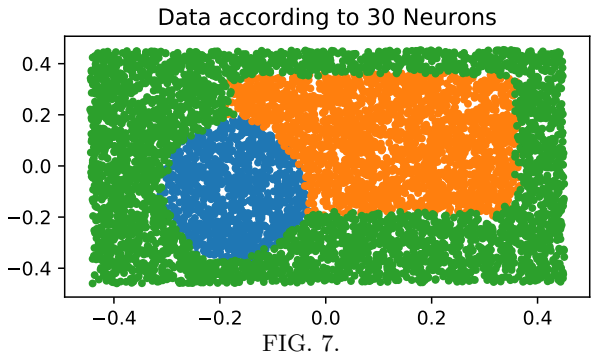

FIG. 7.

Voľba dĺžky kroku α

Počet epoch nastavujem opäť ešte veľký (1000) a spustil som cyklus pre rôzne dlhé kroky. Vzhľadom na to, že ani rádovo netuším, aký veľký by mal byť, skúšam logaritmický sampling na rôznych mocninách:

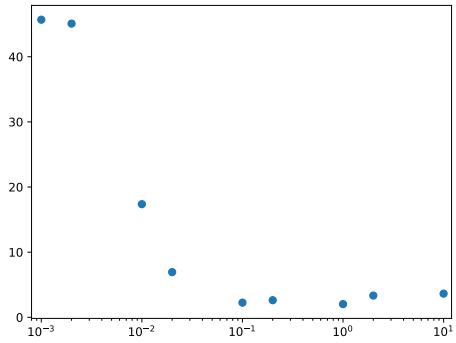

FIG. 8.

Chyba klasifikácie je v tomto prípade najmenšia zhruba pri $\alpha = 0.2$ (FIG. 8). Pre všetky neskoršie výpočty budeme teda pracovať s takýmto krokom (a teda nebudem sa venovať žiadnym jeho adaptáciam).

Efektívny počet epoch

Efektívny počet epoch, cez ktoré musí sieť prejsť, aby sa jej chyba z dlhodobého hľadiska ustálila (nemení sa o viac ako niekoľko percent na 10 epoch) sa ukázal byť okolo hodnoty 300. Vtedy vývoj kvadratickej chyby siete vyzerá ako na grafe FIG. 9 a model považujem za ustálený.

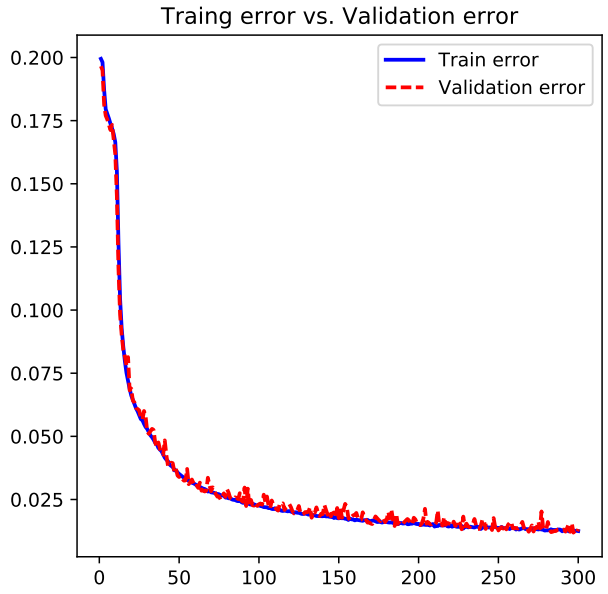

FIG. 9.

Klasifikačné funkcie

Posledná vec, ktorú treba otestovať, je vhodnosť aktivačných funkcií. Porovnávať budem dve známe klasifikačné funkcie: sigmoidu a hyperbolický tangens. Výsledky možno vidieť v tabuľke nižšie.

klasifikátor	Train. class error	Val. class error
sigmoid	2.33 %	1.99 %
tanh	10.31 %	9.01 %

Vidíme, že chyby tanh ďaleko presahujú chyby sigmoidy, takže v hodnoteniach modelov budeme teda používať len sigmoidálnu aktivačnú funkciu.

IV. VYHODNOCOVANIE MODELOV

Všetky modely budem vyhocovať na základne finálnej klasifikačnej chyby pre trénovaciu a validačnú vzorku po 300 epochách. Počet neurónov bude vždy 30 a krok 0.2. Finálna klasifikačná chyba bude určená ako priemerná hodnota k-fold procesu ($k = 8$ podľa zadania).

Klasický back-prop

V tomto modeli aplikujeme klasický back-prop algoritmus a error počítame kvadratickou formou, ako bolo uvedené už vyššie.

Regularizácia

Ak regularizujeme, tak sa v nejakom zmysle bránime príliš komplikovanému modelu. Implementačne pre úpravu matíc a počítanie chyby to vyzerá takto:

$$\mathrm{d}W_{ij} = \alpha\delta_i h_j + \lambda W_{ij}\,,$$

$$\text{Error} = \frac{1}{N}\sum_i (y_i - d_i)^2 + \frac{1}{N}\lambda\sum_i\sum_{k,l} W_{kl}^2\,,$$

Po testovaní modelu pre rôzne hodnoty λ nakoniec vyberám $\lambda = 0$, pretože evidentne zahrnutie regularizácie ničomu nepomáha aj pri extrémne malých hodnotách (FIG. 10).

Niet sa však čomu diviť. Účelom takejto metódy je potláčanie prehnane zložitých hypotéz My však na vstupe máme len samotné dátové body, žiadne ich variácie (mocniny, lineárne kombinácie...), takže odporúčanie pre nulovú lambdu podľa grafu sa dalo očakávať.

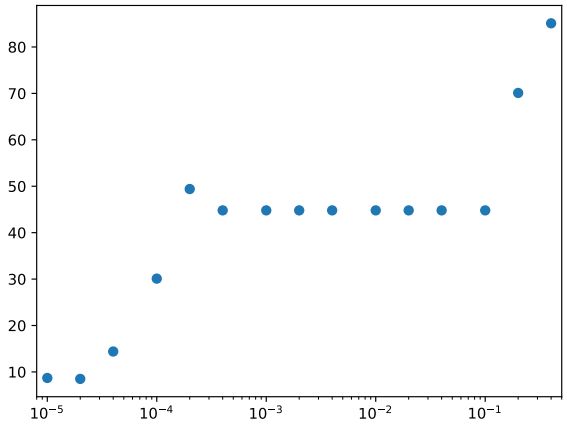

FIG. 10.

Momentum gradient

Momentum gradient sa stará o to, aby sme robili kroky v hľadaní minima našej error funkcie rozumnejšími smermi. Úprava váhových matíc vyzerá nasledovne:

$$\mathrm{d}W_{ij}(t) = \alpha\delta_i h_j + M\mathrm{d}W_{ij}(t-1)\,,$$

Model som testoval pre viaceré rády a hodnoty M. Prekvapivo, výsledky pre klasifikačnú chybu neukazujú žiadny trend (FIG. 11). Vyberám teda také M, ktoré dáva najmenšiu chybu $M = 0.002$ a uvidíme, ako dopadne klasifikačná chyba pri serióznom k-fold testovaní.

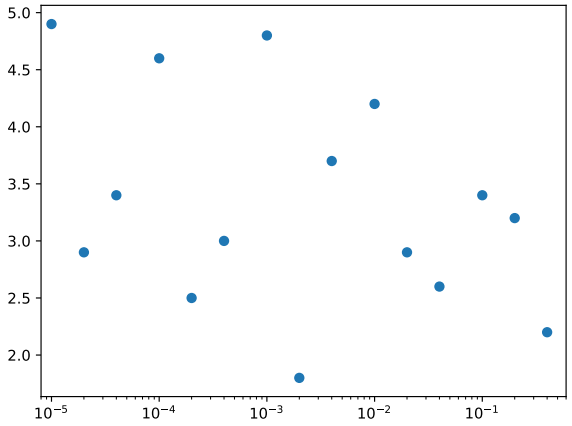

FIG. 11.

Zhodnotenie modelov

Nižšie v tabuľke uvádzam porovnanie všetkých spomenutých modelov. Použili sme klasikcký back-prop, na ktorom sa aplikovala regularizácia a momentum gradient. Výsledky pre back-prop a back-prop s regularizáciou sú totožné, pretože sme v predošlej časti zistili, že regularizácia je najlepšie vtedy, keď nie je žiadna.

Metóda	Train. class error	Val. class error
Back-prop	2.3 3%	1.99 %
Regularizácia	2.33 %	1.99 %
Momentum	1.95 %	1.81 %

Z tabuľky vidno, že najlepším modelom je back-prop s nulovou regularizáciou a momentom rovným 0.2. Na finálny test teda vyberám tento model.

V. TESTOVANIE NAJLEPŠIEHO MODELU

Test prebehol tak, že som namiesto do inputu hlavnej časti programu namiesto validačnej vzorky vložil testovaciu *2d.tst.dat*. Počet epoch, neurónov a dĺžku kroku zachovávam z predošlých modelov (epoch=300, $\alpha = 0.2$, *dim_hid=30*). Podľa zadania už v tomto prípade netreba robiť k-fold cross validáciu, preto *full_set* rozdelím náhodne na 7000 vzoriek na tréning a 1000 na validáciu. Nižšie možno vidieť graf (FIG. 12) vývoja chyby pre estimáciu a validáciu.

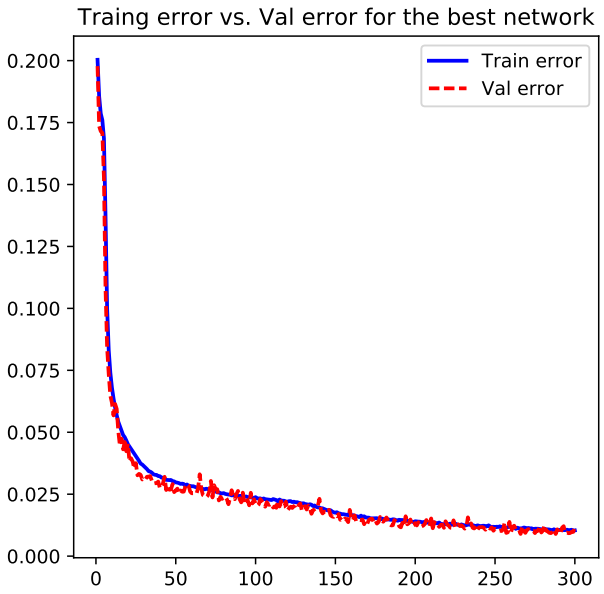

FIG. 12.

Finálna chyba estimácie a validácie sa pohybujú približne okolo hodnoty Error = 0.01 po 300 epochách (po 3000 epochách je to iba = 0.003). Po natrénovaní siete na 7000 vstupoch spúšťam jednoduchý forward algoritmus na testovacej množine (2000 vzoriek) a výsledky vizualizujem (FIG. 13).

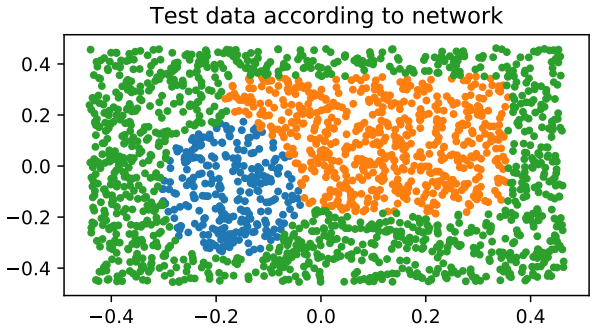

FIG. 13.

Klasifikačné presnosti pre všetky triedy možno vidieť v tabuľke nižšie. V riadkoch je farba triedy, ktorú poznáme z testovacieho datasetu a v stĺpcoch trieda klasifikovaná neurónovou sieťou. Dát je v tabuľke dokopy 2000 a pri každej bunke je klasifikačný pomer zatriedenia farby riadku do farby stĺpca.

data / net	Blue	Orange	Green
Blue	261 (91.9%)	13 (4.6%)	10 (3.5%)
Orange	0 (0.0%)	592 (94.0%)	38 (6.0%)
Green	2 (0.2%)	11 (1.0%)	1073 (98.8%)

Všimnime si, že postupne 13 a 10 modrých dát bolo klasifikovaných ako oranžové a zelené. Naopak, žiadna oranžová nebola zatriedená (!!) ako modrá a 38 bolo ako zelených. Trochu to vyzerá, ako keby bola testovacia množina posunutá uhlopriečkovo doprava hore voči testovacej. Nakoniec, len dve zelené dáta boli zatriedené ako modré a 11 ako ornažové.

Ideálny výsledok by bol diagonálna matica, kedy je celková chyba nulová. Celková klasifikačná úspešnosť našej siete na trénovacej vzorke činí 2.2% na 300 epochách (po 3000 epochách je to iba 0.8%).

Celková cross-entropická chyba siete:

$$\text{Entropy Error} = -\frac{1}{N}\sum_i (d_i \log(y_i) + (1-d_i)\log(1-y_i))$$

$$\text{Entropy Error} = 0.06$$

VI. ZÁVER

Vzhľadom na to, že sme pracovali s veľmi jednoduchou architekúrou neurónovej siete (jedna skrytá vrstva, jednoduchá reprezentácia vstupov) a pomerne jednoduchým optimalizačným algoritmom (momentum), viedla si sieť na daných dátach dobre. Po 3000 epochách vykazuje takmer žiadnu klasifikačnú chybu, čo považujem za sympatický výsledok.