Slovenská technická univerzita v Bratislave
Fakulta informatiky a informačných technológií

Projekt 1: Viacvrstvový perceptrón

Neuróné siete

Meno: Bc. Veronika Včlková
Ročník: 2018/2019
Študijný program: Inteligentné softvérové systémy

1 Report

Projekt bol implementovaný ako všeobecný viacvrstvový perceptrón, spracujúci vstupy na pricípe batch trénovania so spätným šírením chyby, podporujúci ľubovolné množstvo skrytých vrstiev, ľubovolné množstvo neurónov v jednotlivých vrstvách a taktiež ľubovolné množstvo architektúr modelov, na ktorých sa dáta trénujú.

Jednotlivé architektúry sa zadávajú v projekte v *main.py*, na mieste označenom *#models architecture*, ako polia, v ktorých jednotlivé prvky predstavujú počet neurónov na danej vrstve (vstupnú vsrtvu nie je nutné zadávať).

1.1 Predspracovanie dát

Vstupy, ktoré sa majú spracovať, najskôr prejdú **z-score normalizáciou**, čo znamená, že každý vstup (pre x a y osobitne) sa odčíta priemer vstupov a predelí sa štandardnou odchýlkou vstupov.

Normalizácia sa nachádza v *main.py*, označená ako *#normalization*

Následne sa vstupy, ktoré sú určené na trénovanie (odhadovanie a validáciu), náhodne rozdelia medzi 80% odhadovacích dát a 20% validačných dát

Rozdelenie dát sa nachádza v *main.py*, označené ako *#shuffle data*

K výberu architektúry, ktorá podmieňuje tvorbu váhových matíc, patrí aj výber **aktivačných funkcií**. V tomto projekte sú na výber sigmoida, tanh a softmax funkcie.

Výber aktivačných funkcií sa nachádza v *main.py*, označené ako *# sigmoid/softmax/tanh*. Zadáva sa podobne ako architektúra siete cez pole, kde jeden prvok poľa znamená aktivačnú funkciu danej vrstvy a jedno pole predstavuje aktivačné funkcie pre jednu architektúru. Samotné implementácie aktivačných funkcií sa nachádyajú v *model.py*

Inicializácia váhových matíc sa následne vykoná pre jeden model naraz, pričom rozmery každej matice rátajú už aj s biasom. Hodnoty matíc sa generujú medzi 0 a druhou odmocnicou

z (2 / rozmer aktualnej vrstvy) - predstavujúc normalizovanú inicializáciu podľa He et al (2015).

Výpočet váhových matíc sa nachádza v *main.py*, označený ako *#weight metrices*

1.2 Trénovanie

Po prichystaní všetkých dát potrebných k trénovaniu, sa následne vykoná samotné trénovanie - odhad aj validovanie, na každej zvolenej architektúre. Pre každú architektúru sa vytvorí samostatný model so svojími váhavými maticami, architektúrou a aktivačnými funkciami.

Prvým krokom je **onehot encode** dát, ktorý kategórie A,B a C zmení na 2-rozmerné pole predstavujúce kategórie vo vektorovom zápise.

Všetku funkcie spracujúce encode a decode dát, potrebný ku spracovaniu a výpočtom, sa nachádzajú v *util.py*.

Následné trénovanie prebieha v epochách, kde každá epocha obsahuje forward prechod všetkými vstupmi určenými k odhadu, po ktorom sa ráta CE (klasifikačná chyba) a RE (regresná chyba). Tá sa ráta podľa toho, aká aktivačná funckia bola na poslednej vrstve. V prípade sigmoidy a tanh, sa ráta na zákalde **chybových štvrocov**, v prípade softmax sa používa **cross - entropy**. Následne je vykonaná backpropagácia chyby a úprava váhových matíc.

Po každej 10 epoche sa zmení **learning rate** daného modelu, konkrétne sa násobí koeficientom 0,5. To spôsobí pravidelné zmenšenie tohto learning rate.

Trénovanie obsahuje aj **early stopping**. To je implementované tak, že po každej epoche sa ráta aj CE a RE vo funkcii *test*, ktorá obsahuje len forward prechod dátami, na dátach určených na validovanie. Z týchto validačných CE chýb sa pamätá minimum a v prípade, že ďalších 15 epôch nebolo schopných

dosiahnuť validačnú CE chybu menšiu ako pamätané minimu, trénovanie sa ukončí.

1.3 Testovanie

Z každého natrénovaného modelu sa pamätá posledná validačná CE chyba, na zákalde ktorej sa vyberá najlepší model. Pre tento model sa vykoná trénovanie na celom datasete cez rovnakú funkciu ako prebiehalo trénovanie aj pred tým, avšak bez early stopping. Model sa trénuje na toľkých epochách, na koľkých zastavil pošas prvotného trénovania.

Následne prebieha testovanie dát na testovacom datasete, pri ktorom sa vykoná len forward prechod a výpočet CE a RE chýb.

1.3.1 Konkrétne modely

Projekt bol trénovaný na 4 modeloch, z ktorých bol následne vybraný na zákalde najmenšej validačnej CE chyby ten najlepší.

Architektúry - počet prvkov je počet vrstiev, hodnota prvku je počet neurónov na danej vrstve (každý model obsahuje ešte prvú vstupnú vsrtvu):

- 1. - 128, 128, 128, 3 - tanh, tanh, tanh, softmax aktivačné funkcie
- 2. - 128, 3 - tanh, softmax aktivačné funkcie
- 3. - 30, 20, 3 - tanh, tanh, softmax aktivačné funkcie
- 4. - 70, 50, 40, 3 - tanh, tanh, tanh, softmax aktivačné funkcie

Trénovanie a validácia na trénovacích dátach pre všetky architektúry vygenerovali chyby, ktoré sú zobrazené na tab. 1

Tabuľka 1: Chyby trénovania a validovania dát jednotlivých modelov

		Estimation error		Validation error	
	Epochs	CE	RE	CE	RE
1	53	0,33%	0,01537	0,50%	0,01821
2	76	0,97%	0,05351	0,94%	0,05369
3	51	0,53%	0,03285	1,12%	0,03434
4	78	0,19%	0,01658	0,38%	0,01869

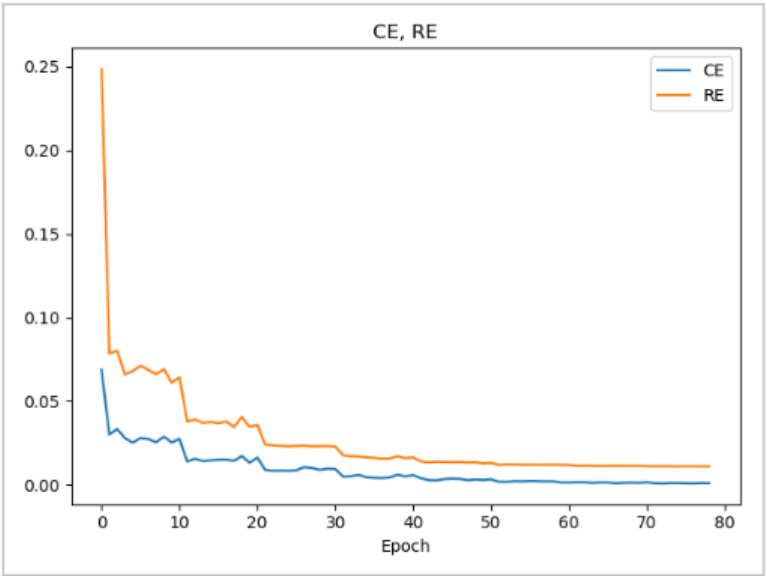
1.3.2 Najlepší model

Z toho, ako najlepší model, na zákalde najnižšej validačnej chyby bol vybratý model 4, s architektúrou 70, 50, 40, 3 - tanh, tanh, tanh, softmax. Pre jeho hyperparametre bolo vykonané trénovanie znova, tentokrát na celom datasete.

Priebeh trénovania najlepšieho modelu naprieč epochami:

```
BEST MODEL - architecture [70, 50, 40, 3]

0 epoch, CE = 6.88%, RE = 0.24842.....
10 epoch, CE = 2.75%, RE = 0.06435.....
20 epoch, CE = 1.64%, RE = 0.03590.....
30 epoch, CE = 0.95%, RE = 0.02291.....
40 epoch, CE = 0.59%, RE = 0.01639.....
50 epoch, CE = 0.34%, RE = 0.01316.....
60 epoch, CE = 0.15%, RE = 0.01190.....
70 epoch, CE = 0.16%, RE = 0.01123.....
LAST EPOCH
78 epoch, CE = 0.10%, RE = 0.01109
```

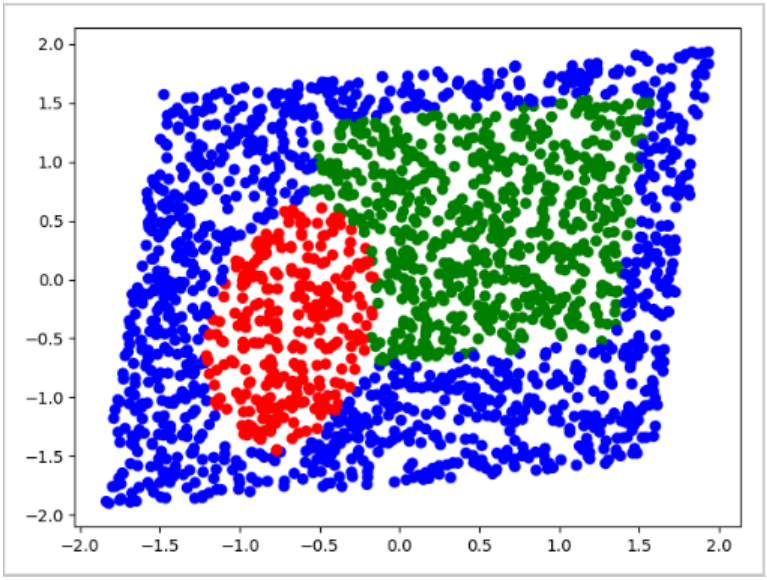


Obr. 1: Chyby trénovania najlepšieho modelu

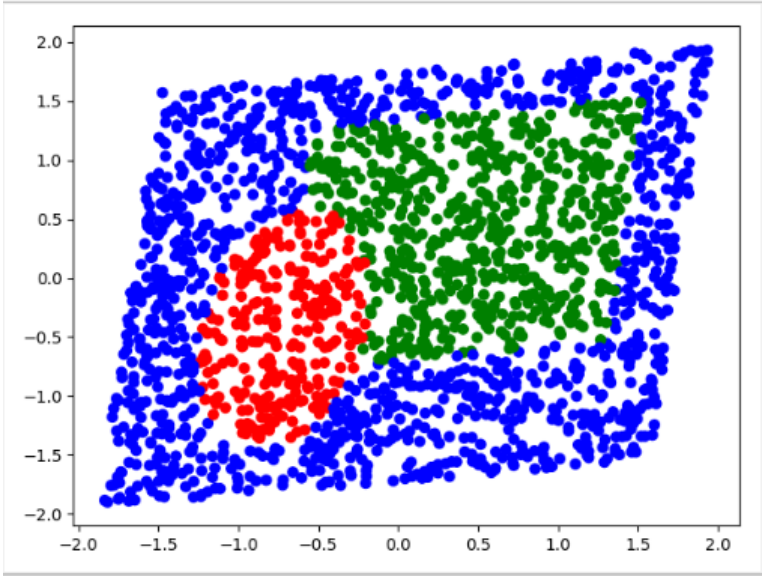
TESTOVANIE

S natrénovanými váhami prebehlo testovanie na testovacom datasete s výslednými chybami:

Final testing error: CE = 2.90%, RE = 0.08136



Obr. 2: Zadané dáta



Obr. 3: Predikované dáta

Confucion Matrix

Na tab.2 vidíme jednotlivé kategórie a ich odhadovanie sieťou.

Tabuľka 2: Confusion Matrix testovania

Confusion Matrix			Predicted Class	
		A	B	C
	A	265	11	8
Actual Class	B	0	603	27
	C	10	2	1074