ML – Brain scan detection

Polifronie Dragos-Constantin, 344

Rezolvarea acestui task se bazeaza classifierul multi-layer de perceptroni din libraria sklearn.

Au fost conduse mai multe teste in ceea ce priveste gasirea unei variante mai bune, printre care:

Modificarea numarului de layere, modificarea solverului si rata de invatare initiala, dar si prelucrarea in prealabil a datelor (in prima instanta normalizarea lor, urmand mai apoi testarea acuratetii dupa reducerea zgomotului din imagini si aplicare threshold – testate in late submission)

General:

- Imaginile au fost incarcate cu ajutorul librariei opencv, iar procesul a fost grabit folosind multiprocessing.
- Imaginile au fost mai apoi normalizate printr-un StandardScaler din libraria sklearn, in functia normalize_data(train_data, test_data, type='standard')
- Intrucat imaginile au fost numeroase, iar timpul si resursele nu mi-au permis folosirea tuturor, doar un procent (ales convenabil) din pozele de training au fost folosite, fapt ce probabil a tras in jos performanta modelului. In submisiile trimise, doar 10% din imagini au fost folosite pentru antrenare.

Parametrii testati inainte de submisii: (testing score se refera la pozele carora le stim clasa, adica poze din validation si /sau training dar care nu au fost folosite pentru antrenare, intrucat pozele de testare nu sunt clasificate)

In urmatoarele teste vor fi puse niste imagini ce redau rezultatele testarii mai multor functii de activare, impreuna cu solverul folosit ca parametru

- Primul test a fost facut pe un clasificator cu 2 layere a cate 100 de neuroni fiecare si cu functia de activare = identity.

  Intrucat functia de activare identity este una lineara iar datele nu sunt, asa cum ne-am astepta, rezultatul nu este satisfacator, iar aceasta metoda nu este intocmai eficienta pentru modelul nostru.

Rezultate pentru functia de activare identity, pe o retea cu 2 layere a cate 100 neuroni.

Solver	Iterations	Training loss	Testing score
sgd - constant	200	nan	0.387352
sgd - invscaling	200	nan	0.387352
adam - init=0.01	57	0.242794	0.648221
adam - init=0.001	118	0.039542	0.683794
lbfgs - alpha=0.0001	200	0.153843	0.652174
lbfgs - alpha=0.00001	200	0.116240	0.628458

Se poate observa ca functia sgd nu a putut converge din cauza linearitatii functiei de activare.

- Rezultate pentru functia de activare tanh

Solver	Iterations	Training loss	Testing score
sgd - constant	200	0.396053	0.679842
sgd - invscaling	68	0.657834	0.592885
adam - init=0.01	30	0.696097	0.612648
adam - init=0.001	16	0.674090	0.450593
lbfgs - alpha=0.0001	200	0.063070	0.735178
lbfgs - alpha=0.00001	200	0.073599	0.691700

- Rezultate pentru functia de activare logistic:

Solver	Iterations	Training loss	Testing score
sgd - constant	200	0.653397	0.727273
sgd - invscaling	14	0.714827	0.387352
adam - init=0.01	16	0.688389	0.660079
adam - init=0.001	37	0.636158	0.723320
lbfgs - alpha=0.0001	200	0.352143	0.667984
lbfgs - alpha=0.00001	200	0.369488	0.703557

- REZULTATE PENTRU FUNCTIA DE ACTIVARE RELU:

Solver	Iterations	Training loss	Testing score
sgd - constant	200	0.156805	0.691700
sgd - invscaling	146	0.430830	0.700803
adam - init=0.01	67	0.523427	0.687747
adam - init=0.001	101	0.002098	0.707510
lbfgs - alpha=0.0001	200	0.000015	0.735178
lbfgs - alpha=0.00001	200	0.000008	0.743083

Observam ca in cazul acestei functii, rezultatele deja se imbunatatesc, asa ca functia relu va fi folosita pentru urmatoarele testari.

In urmatoarele teste, vom modifica numarul de neuroni din layere, folosind de aceasta data 3 layere a cate 200. (functia de activare ramane aceeasi – relu)

Rezultate pentru activation = relu, hidden_layer_sizes = (200, 200, 200)

Solver	Iterations	Training loss	Testing score
sgd - constant	200	0.018041	0.695652
sgd - invscaling	200	0.769314	0.592885
adam - init=0.01	69	0.333450	0.699605
adam - init=0.001	96	0.000965	0.731225
lbfgs - alpha=0.0001	115	0.000024	0.762846
lbfgs - alpha=0.00001	116	0.000009	0.766798

In afara de solverul sgd- invscaling, toate scorurile de testare au crescut. Vom incerca in urmatorul test o retea cu 3 layere a cate 800 de neuroni fiecare

Solver	Iterations	Training loss	Testing score
sgd - constant	111	0.008159	0.620553
sgd - invscaling	200	0.700378	0.608696
adam - init=0.01	70	0.640721	0.683794
adam - init=0.001	53	0.001142	0.743083
lbfgs - alpha=0.0001	91	0.000066	0.679842
lbfgs - alpha=0.00001	90	0.000011	0.687747

Un ultim test, cu 3 layere, avand 800, 400, respectiv 200 neuroni:

Solver	Iterations	Training loss	Testing score
sgd - constant	191	0.007907	0.707510
sgd - invscaling	200	0.770376	0.604743
adam - init=0.01	94	0.052771	0.695652
adam - init=0.001	112	0.001718	0.747036
lbfgs - alpha=0.0001	76	0.000063	0.687747
lbfgs - alpha=0.00001	78	0.000022	0.719368

Printre aceste rezultate, solverul adam cu rata initiala de invatare 0.001 a adus in medie cele mai bune rezultate, asa ca acesta a fost ales pentru submisii.

Avand in vedere aceste rezultate:

Prima submisie (cea prezentata in **model1.py**) care a primit scor **0.27** , a folosit 3 layere a cate 200 de neuroni, fara early stopping cu un tol de 0.001.

A doua submisie (cea prezentata in **model2.py**) a primit un scor de aprox **0.32** si a folosit 3 layere a cate 800 de neuroni, fara early stop, cu aceeasi valoare de tol.

A fost “testata” problema unui overfit, prin a 3a submisie care a folosit un early stopping = True, dar care a avut un rezultat mult sub primele, si anume 0.04

In **late submissions** (folosite pentru adunarea de informatii pentru documentatie), algoritmii au fost alterati, marind numarul de imagini folosite pentru antrenare si batchul fiecarei epoci in clasificatorul mlp.

De altfel, imaginile au fost preprocesate pentru a obtine date mai utile, prin reducerea zgomotului din imagini cu ajutorul unui filtru gaussian cu kernel 5x5 (a fost folosita biblioteca opencv). In aceste cazuri, totusi, acuratetea nu a fost imbunatatia, ci dimpotriva, diminuata, cele mai bune rezultate locale fiind de 0.38 iar publice 0.30

Imaginile au fost trecute printr-un blur gaussian iar mai apoi un threshold adaptiv a fost aplicat pentru a evidentia partile de interes din imagine. In acest caz, insa, poate fi vorba de un overfit, intrucat pe multimea de validare, accuratetea a ajuns chiar pana la 90%, dar cu toate acestea, la testele finale (submisii) aceste modele au obtinut un scor mai mic decat submisia nr 2.

In concluzie, modelul mlp propus ar putea fi imbunatit pentru a aduce o acuratete mai mare prin prelucrarea mai buna a imaginilor (probabil un sharpening al imaginii inainte de threshold ar ajuta) si printr-o mai buna gestionare a evolutiei modelului, intrucat este posibil ca modelul sa fi ajuns la overfit si sa nu mai poata clasifica imaginile de testare. Cu toate acestea, modelul nu pare promitator in a atinge o acuratete foarte mare (majoritatea rezultatelor, indiferent de schimbarile aduse, se invart in jurul acuratetei de 0.3-0.4)

-biblioteci folosite:

* Tensorflow `2.11.0`
* `opencv 4.5.4`
* `sklearn 1.0.2`