Poročilo o domači nalogi 5 za UOZP

Tim Kmecl, 2022

Utemeljitev pravilnosti Softmax

Pri implementaciji funkcij softmax, cost in grad sem se za matematično stran implementacije skliceval na članek Softmax Regression s Stanfordove spletne strani (http://deeplearning.stanford.edu/tutorial/supervised/SoftmaxRegression/). Enačbe, ki so navedene na tej strani, sem iz vektorske ali skalarne oblike pretvoril v matrično obliko (zaradi večje hitrosti implementacije z numpy), te pa so v moji kodi. Če so pravilne enačbe na spletni strani, bi morala pravilno delovati tudi moja implementacija.

To sem se odločil še testirati s testnimi primeri. Če so testni podatki enaki učnim in so primerki razredov dovolj narazen, bi moral biti CA enak 1, logloss pa blizu 0. Če pa dodam še regularizacijo, bi se moral logloss postopoma večati, CA pa nižati. Za zelo velik lambda se model ne more naučiti ničesar, zato bi v tem primeru moral biti CA enak 1/(št. razredov), kar je enako, kot če bi ugibali. To sem testiral na podatkih mirisX in mirisY, iz datoteke hw5_test.py, in rezultati testa se skladajo z zgoraj navedenim. Na istih podatkih sem preveril še, kaj se zgodi, če testni podatki niso enaki učnim (z uporabo svoje funkcije za cross-validation). Smiselno bi bilo, da je logloss manjši kot takrat, ko so tesni enaki učnim (tako kot je v sploščnem napaka na učni množici manjša kot na testni), CA pa mora biti vseeno bistveno večji, kot če bi ugibali. Test pokaže, da za mojo implementacijo to drži.

Moj softmax sem testiral še na sintetičnih podatkih, da bi preveril matematično modeliranje verjetnosti. V prvem primeru sem generiral 1000 primerov s 5 atributi, kjer so vsi elementi povsem naključni, prav tako pa je vsakemu od primerkov naključno dodeljen eden od treh razredov. V tem primeru bi pričakovali, da bo model vsakemu novemu naključnemu primerku dodelil verjetnost za vsak razred 1/3, kar sem testiral na treh naključno generiranih novih primerkih. Res se izkaže, da model dodeli verjetnosti skoraj točno 1/3. Generiral sem še podatke, kjer ima 300 primerkov le 1. atribut enak 1, ostala dva 0, 300 le 2. atribut enak 1 in 300 le 3. atribut enak 1. Tisti s prvim atributom 1 so v razredu 0, tisti z drugim v razredu 1 in tisti s tretjim v razredu 2. Softmax model bi moral potem vsem novim primerkom, ki imajo le 1., 2., ali 3. atribut enak 1, dati verjetnost, da so v 1., 2. oziroma 3. razredu 1, ostale pa 0. Prvi trije primerki zadnjega testa pokažejo, da je to za moj model res. Za primerek, ki ima npr. 2. in 3. atribut enak 1, prvega pa 0, bi morala biti verjetnost, da se nahaja v 1. razredu enaka 0, verjetnosti za 2. in 3. razred pa enake, in sicer ½, saj je enako verjetno, da se nahaja v katerem od the dveh razredov. Za primerek z vsemi atributi enakimi 1 pa je verjetnost za katerikoli razred enaka, toej 1/3. Test pokaže, da to pri moji implementaciji res drži.

```
====   Test podtkov miris, napoved na učni množici
lambda: 0.001  -  CA: 1.0, logloss: 0.025437888343417318
lambda: 1  -  CA: 0.9666666666666667, logloss: 0.2804352208529653
lambda: 10  -  CA: 0.9666666666666667, logloss: 0.5029558156065213
lambda: 100  -  CA: 0.3333333333333333, logloss: 0.9982713182164463
====   Test prevelike regularizacije na podatkih mirirs
lambda: 10000000  -  CA: 0.3333333333333333, logloss: 1.0982669241791234
====   Test podtkov miris, 3x prečno preverjanje
lambda: 0.001  -  CA: 1.0, logloss: 0.043068441802959836
====   Test na naključnih atributih, naključni razredi
[[0.333174   0.33335572 0.33347029]
 [0.33339562 0.33338424 0.33322013]
 [0.33335375 0.33328282 0.33336343]]
====   Test na podatkih, kjer so atributi one-hot encoding razreda
Testni podatki
[[1 0 0]
 [0 1 0]
 [0 0 1]
 [0 1 1]
 [1 1 1]]
verjetnost za razrede:
[[9.99825559e-01 8.99647912e-05 8.44762695e-05]
 [9.54907542e-05 9.99817970e-01 8.65394108e-05]
 [9.90400421e-05 9.58373528e-05 9.99805123e-01]
 [4.96735712e-05 5.15185470e-01 4.84764857e-01]
 [3.52750885e-01 3.36984006e-01 3.10265109e-01]]
```

Slika 1 Izpis rezultatov testov implementacije iz programa

Napovedovanje

Za napovedovanje sem uporabil le gole podatke, ki sem jih prebral iz datoteke, ločil atribute od razredov (razrede sem pretvoril v številke od 0 do 8), in jih dal neposredno v Softmax learner. S testiranjem različnih lambd s pomočjo moje funkcije za cross-validation na učni množici sem ugotovil, da je najboljši ragularizacijski parameter lambda=4, ki sem fa uporabil za končne napovedi. Pri tem parametru je bil dosežen CA pri prečnem preverjanju 0,76, logloss pa 0,649. Čas gradnje končnega modela na mojem računalniku (3 leta star laptop z intel i7 procesorjem) znaša približno 9 sekund.