6. domača naloga: Gradientni boosting

Luka Krsnik (63110179)

31. maj 2015

1 Uvod

Cilj šeste domače naloge je implementacija gradient boostinga, overitev gradienta in cenilne funkcije z metodo končnih razlik ter ovrednotenje njegovih rezultatov.

2 Metode

Odločitvena drevesa Ta drevesa za napovedni model uporabijo opažanja o stvareh, ki jih klasificirajo. Konkretno, vzamejo nek atribut, preko katerega razdelijo rezultat na dva dela, na naslednjem nivoju uporabijo drug atribut itd.

Naključni gozd Naključni gozd je izpeljanka iz odločitvenih dreves. Deluje tako, da iz večih odločitvenih dreves naredi povprečje, s čimer se izogne prevelikemu prileganju podatkov učni množici.

Linearna regresija Regresija je postopek, s pomočjo katerega je na podlagi učnih podatkov zgrajena "funkcija", ki iz značilk čim bolje predvidi tisto kar je naučena. Osnovni princip njenega izračuna je tak, da je izražena funkcija za izračun vsote kvadratov napak, v katero so vstavljeni takšni parametri, da bo njen rezultat čim manjši.

Gradient boosting Gradient boosting je metoda, katere cilj je, da iz značilk čim bolje klasificira določen primer v nek razred oziroma določi s kakšno verjetnostjo nek primer spada v določen razred. Za to metodologijo je značilno, da uporablja druge preproste klasifikacijske metode, nato pa se v vsaki iteraciji uči iz napak teh preprostih metod.

Cross validation Cross validation je metoda, ki ovrednoti uspešnost našega algoritma. Deluje tako, da so učni podatki razdeljeni na k delov (ponavadi 10), nato pa se metoda izvede k-krat. V vsaki iteraciji je eden izmed k-tih delov izpuščen iz učnih podatkov in uporabljen kot testni podatek. Vsi "testni podatkišo nato postavljeni v skupno matriko in ocenjeni.

Metoda končnih razlik Ta metoda je uporabljena pri preverjanju pravilnosti zastavljenih enačb. Uporablja se tako, da se primerja rezultat odvoda (v gradientu) z vrednostmi, ki se pridobijo kot rezultat iz metode končnih diferenc. Če so skoraj enake, potem so zastavljene enačbe pravilno odvedene, sicer ne.

Logarithmic Loss Logarithmic Loss je ena izmed metod za ovrednotenje rešitve. Ta metoda je uporabljena na tekmovalnem strežniku, zaradi cross validacije pa je implementirana tudi lokalno. Njena ideja je v tem, da sešteva logaritme napovedanih vrednosti in jih množi z 1 ali nič, odvisno od tega, če je nek izdelek v tistem razredu.

3 Rezultati

V nalogi je implementirno preverjanje KL-divergence z numeričnim odvajanjem (to je storjeno na iris podatkih, saj jih je bilo v konkretnem primeru preveč). Za testiranje le te je potrebno odkomentirati dve vrstici v kodi. Vgrajena je tudi neodvisnost algoritma od implementacije šibkega modela in funkcije izgube. Rezultati so preverjeni tudi s pomočjo cross validacije. Ker je podatkov zelo veliko je k enak 5. V nadaljevanju so podrobneje predstavljeni rezultati. Naj omenim še, da je funkcija LogLoss prekopirana iz Oranga, saj sam še nisem posodobil njegove verzije.

Na predavanjih smo delali z odločitvenimi drevesi. Zato sem tudi sam začel s temi. Zaradi počasnosti algoritma pri velikih iteracijah (te so zapisane v parametru n_estimators) sem se odločil, da za prečno preverjanje (parameter fit - za dejansko preverjanje sem vzel rezultate iz strežnika) uporabim malo iteracij. Pri odločitvenih drevesih sem tako spreminjal globino dreves in število iteracij gradient boostinga. Rezultati so zapisani v Tabeli 1. Razvidno je, da so pri večji globini, rezultati boljši. Enako velja tudi za število iteracij (čeprav bi v teoriji pri zelo veliko iteracijah lahko prišlo do overfittinga). Kar v rezultatih ni zapisano pa je čas izvajanja. Večja kot je globina, počasneje se je algoritem izvajal.

Preizkusil sem še določene druge algoritme, od katerih se nobeden ni izkazal za pretirano uporabnega. Tudi ti rezultati so zapisani v Tabeli 1. Kot je razvidno, so rezultati z naključnimi gozdovi le za malenkost boljši od odločitvenih dreves. Zaradi približno petkrat počasnejšega izvajanja se mi je njihova uporabnost zdela zelo majhna (namesto tega raje povečam število iteracij gradient boostinga). Linearna regresija pa je že sama po sebi vrnila precej slabe rezultate in se prav tako izvajala počasi.

Tabela 1: Tabela z rezultati cross validacije.

preprosti algoritem	parametri	rezultat
decision tree regressor	max_depth=2; n_estimatirs=10	1.05462168
decision tree regressor	max_depth=3; n_estimatirs=10	0.94348305
decision tree regressor	max_depth=3; n_estimatirs=30	0.73468678
decision tree regressor	max_depth=4; n_estimatirs=10	0.87644582
decision tree regressor	max_depth=6; n_estimatirs=10	0.79629589
random forest regressor	max_depth=3; n_estimatirs_rf=5; n_estimatirs=10	0.939261
random forest regressor	max_depth=5; n_estimatirs_rf=5; n_estimatirs=10	0.82402652
linear regression	n_estimatirs=10	0.98031033

Zaradi počasnosti algoritma sem le na strežnik oddal rešitve z nekoliko večim iteracijam

gradient boostinga in večji globini odločitvenih dreves (na strežnik sem oddajal samo te, ker sta se mi ostali dve metodi, zaradi prečnega preverjanja, zdeli precej slabši). Rezultati so zapisani v Tabeli 2, v zadnji vrstici pa so zapisani rezultati rešitve, ki je oddana kot končna.

Tabela 2: Tabela z rezultatov iz strežnika.

preprosti algoritem	parametri	rezultat
decision tree regressor	max_depth=3; n_estimatirs=20	0.80297
decision tree regressor	max_depth=3; n_estimatirs=150	0.59749
decision tree regressor	max_depth=6; n_estimatirs=1000	0.53518

4 Izjava o izdelavi domače naloge

Domačo nalogo in pripadajoče programe sem izdelal sam.