2. domača naloga pri predmetu ITAP

Anamarija Potokar

11. maj 2025

1. naloga: Izbira modelov

a)

Za vsako izmed šestih podatkovnih množic sem izbrala enega izmed možnih modelov in svojo odločitev tudi utemeljila.

b)

Za vsakega izmed podanih štirih scenarijev sem predlagala izbiro ustreznega modela.

2. naloga: SVM

a)

V tej podtočki sem implementirala funkcijo, ki za podan primer x, uteži w in predsodek b napove razred. Odločitev temelji na vrednosti izraza $w^T x + b$. Če je ta vrednost ≥ 0 (ležimo nad oz. na odločitveni krivulji), potem napove razred 1 oz. pozitiven razred, če pa primer leži pod odločitveno krivuljo, pa razred -1 oz. negativen razred. Testni primeri so potrjeni z *assert* in kažejo, da funkcija pravilno deluje za različne kombinacije vhodov.

b)

Najprej sem približno shranila podatke s slike v numpy array X in jim priredila istoležne ciljne vrednosti razredov, prav tako shranjene v numpy array-u y. Iz vaj sem si sposodila in še nekoliko modificirala funkcijo *vizualize_decision_boundary*, ki vizualizira odločitveno krivuljo in po novem še izpiše enačbo krivulje oz. v tem primeru premice. Nato sem na naboru podatkov X in y naučila model SVM, s pomočjo dobljenih uteži w in b definirala nove točke, ki ležijo nad, na in pod odločitveno krivuljo, preverila delovanje funkcije *napovej* na teh točkah in vizualizirala stare točke, nove točke in odločitveno krivuljo. Tako klasifikacija kot vizualizacija sta se izkazali za uspešni.

c)

Pri tej podnalogi sem po navodilih napisala kodo za algoritem, ki je približek linearni klasifikaciji, ki se zgleduje po principih metode podpornih vektorjev, a se jo uči z bolj preprostim iterativnim postopkom. Ideja je, da če je napoved napačna oz. preblizu

odločitveni meji, okrepimo vpliv tega primera in premaknemo mejo, če je pa primer ustrezno klasificiran in daleč od meje, pa le rahlo zmanjšamo uteži, da ohranimo stabilnost modela. Prav tako sem implementirala tudi funkcijo za iskanje podpornih vektorjev, tj. točk, kjer je $y_i(w^T x_i + b) < C$.

d)

Funkcijo iz prejšnje podtočke sem zdaj testirala na podani množici podatkov in jih skupaj s podpornimi vektorji ter odločitveno krivuljo vizulizirala. Model ni bil uspešen, saj poskuša podatke, ki niso linearno ločljivi (generirani so s pomočjo *make_circles* in zato seveda nobena ravna črta ne mora uspešno ločevati primerov). Ker linearni PSVM ni primeren za nelinearne podatke, je potrebna razširitev modela na različna jedra.

e)

Namesto uteži w v novem modelu, v katerem lahko uporabljamo tudi nelinearna jedra, uporabljamo koeficiente α, kjer vsak primer prispeva k odločitveni funkciji prek jedra. Pri napovedovanju uporabimo le podporne vektorje, da pospešimo računanje.

f)

V tej podtočki sem definirala polinomsko in RBF jedro in nato model z novimi jedri ponovno testirala na istih nelinearnih podatkih. Seveda sta bila oba modela z nelinearnim jedrom veliko uspešnejša pri delu z nelinearno ločljivimi podatki, kot je bilo tudi pričakovano.

3. naloga: Nevronske mreže

Moja naloga pri 3. točki je bila naučiti model, kako napovedati čustveno stanje človeka na podlagi slike njegovega izraza na obrazu. Za začetek sem definirala pot do učnih in testnih podatkov. Nato sem ustvarila ločeno verzijo *train_set*-a brez normalizacije, da sem preverila, kakšna sta povprečje in standardni odklon, da sem potem lahko karseda natančno normalizirala podatke pred definicijo in učenjem nevronske mreže. Poleg normalizacije sem izvedla še druge transformacije, kot so bile pretvorba v črnobele (da imam na vhodu res samo en kanal namesto 3), prilagoditev velikosti in transformacija slik v tenzorje.
Nato sem se lotila sestavljanja konvolucijske nevronske mreže. Ker imamo možnih 7 čustev, model prejema slike oblike [1, 48, 48] in napoveduje verjetnost za vsako izmed 7 čustev oz. razredov. Potem sem inicializirala model, kjer sem za funkcijo izgube uporabila križno entropijo, optimizator Adam, ker je bolj robusten od navadnega SGD, in learning rate 0.001 (kot privzeto), kar je dovolj malo, da ne preskoči optimalne točke, ampak dovolj veliko, da učenje ne bo prepočasno. Uporabim še *torch.manual_seed* za reproducibilnost rezultatov. Nato sem natrenirala model in ga testirala na testnih slikah. Model je pred treniranjem pravilno napovedal okoli 830 od skupno 7178 primerov, po treniranju pa je število pravilnih napovedi naraslo na okrog 4000, kar je že bistveno boljši rezultat, a še vedno dokaj malo, zato sem ga poskusila izboljšati. Najprej sem preverila, ali je kateri izmed razredov (bistveno) bolj zastopan v učnih podatkih kot ostali. Prišla sem do ugotovitve, da je zelo redek razred *disgust* s 438 primeri, zelo pogost pa razred *happy* s 7215

primeri. Lahko se zgodi, da se model nauči favoritizirati pogostejše razrede pred ostalimi in ignorira redke razrede (ker je v povprečju, če se ne nauči teh nekaj primerov redkejših razredov, napoved lahko še vedno dokaj točna). To sem poskusila popraviti tako, da sem utežila redkejše razrede (strožja kazen za napačne napovedi zanje), za pogostejše razrede pa vpeljala milejše kazni. Še vedno sem uporabljala isto funkcijo izgube in optimizator. Po treningu je model napovedal manj pravilnih čustev, ampak to je lahko zato, ker mora žrtvovati nekaj pravilnih napovedi večinskih razredov, zato skupna točnost lahko pade, ampak se pa nauči bolje prepoznavati redkejše razrede. Da sem bolje videla, kaj se dogaja z napovedmi posameznih razredov, sem vizualizirala confusion matrix pred in po uporabi uteži. To je potrdilo sum, namreč model je z uporabo uteži dejansko najredkejši razred *disgust* napovedoval veliko bolje, vendar pa je bolj zastopane razrede napovedoval slabše, predvsem pa je mešal med seboj *happy* z *neutral* in *angry* ter *fear, sad, neutral*. Torej je uporaba uteži pomagala pri boljšem prepoznavanju redkih razredov, ampak še vedno so ostala izziv čustva, ki imajo med seboj podobno obrazno mimiko.
V naslednjem koraku sem poskusila izboljšati model tako, da sem poleg uteži dodala še nove transformacije učnih slik, kot je na primer naključna rotacija slike. To bi lahko prišlo prav, ker model vidi še "druge vidike"istih izrazov in bi se morda naučil bolje razločevati med podobnimi čustvi. Vendar pa je bil tudi ta model spet slabši, kot prej. Zato sem se odločila, da odstranim uteži in poskusim napovedovati samo z uporabo novih transformacij. Po izpisu confusion matrix-a sem videla, da je bil ta model boljši pri napovedovanju čustev iz razredov z dovolj podatki, medtem ko za najmanj zastopan razred *disgust* ni naredil niti ene pravilne napovedi oz. se ga sploh ni naučil in ga nikoli ni napovedal, prav tako ima model še vedno težave ločevati podobne izraze.
Ker sem še vedno želela, da bi model vsaj bolje prepoznaval *disgust*, sem poskusila uporabiti *WeightedRandomSampler*, da bi model pogosteje videl primere tega razreda in imel s tem več priložnosti, da se ta razred nauči. Pri tem nisem uporabljala dodatnih transformacij, saj lahko smer pogleda, simetrija ipd. nosijo pomembne informacije. Po treningu in testiranju modela ter izpisu confusion matrix-a sem ugotovila, da je imel *disgust* največ pravilnih napovedi do sedaj, vendar je še vedno pogosto bil zamešan z *angry*, prav tako je še vedno prihajalo do mešanja med podobnimi čustvi.
Na koncu mi je že malo zmanjkalo idej in časa, kako bi se še lahko lotila izboljšave modela. Zagotovo pa se mi ne zdi to, da ni prepoznaval nekaterih čustev oz. jih je med seboj mešal, nič presenetljivega, saj imamo že ljudje sami pogosto težave s prepoznavanjem čustev drugih, pa tudi ko sem si pogledala slike iz učne in testne množice, so se zdele nekatere res zelo podobne in nerazločne, zato se mi končno število pravilnih napovedi, v originalnem modelu brez dodatnih prilagoditev, okoli 4000 niti ne zdi tako zelo slabo.