Univerza v Ljubljani, Fakulteta za elektrotehniko
Podiplomski magistrski študijski program 2. stopnje Elektrotehnika
Informacijsko komunikacijske tehnologije - Obdelava slik in videa (64238)

NAVODILA

Vaja 12: Razpoznavanje števk z globokim učenjem

Pripravila: Gašper Podobnik & Tomaž Vrtovec

Navodila

Pred začetkom vaje si je potrebno namestiti knjižnici `torch` in `torchvision`. To storite po enakem postopku, ki je za knjižnico `opencv-python` opisan v *Navodilih za namestitev tolmaca Python*, ki se nahajajo v spletni učilnici FE.

Slika 1: Logo knjižnice PyTorch, ki jo bomo uporabljali pri tej vaji

Metode globokega učenja uvrščamo med metode strojnega učenja, ki spadajo med pristope k zasnovi umetnih inteligentnih sistemov. Pridevnik *globoko* se nanaša na arhitekturo nevronskih mrež, ki je pri globokem učenju tipično precej kompleksna. Take nevronske mreže imenujemo tudi *večslojne* nevronske mreže, ki imajo običajno veliko število parametrov, katere želimo v procesu učenja mreže določiti tako, da bo 'natrenirana' nevronska mreža čim bolje opravila svojo nalogo.
Področje strojnega učenja in umetne inteligence je v zadnjih dveh desetletjih raziskovalno zelo aktivno, zato se znanje in metode na tem področju vsako leto izboljšujejo. Namen te vaje je na primeru klasifikacije slik spoznati osnovne principe delovanja tovrstnih metod in spoznati nekatere izzive in pasti na katere je potrebno paziti, ko uporabljamo metode globokega učenja.

1. Pri vaji boste uporabljali podatkovno bazo imenovano MNIST. Gre za bazo sličic ročno napisanih števk od 0 do 9, velikosti $X \times Y = 28 \times 28$. V nadaljevanju dopolnite Python skripto `vajaDL.py`, ki se nahaja v gradivu za vaje.

   S pomočjo knjižnice `torchvision` avtomatsko prenesite podatkovno bazo slik iz spletnega repozitorija. Uporabite funkcijo `torchvision.datasets.MNIST`.
   Podatkovno bazo sličic razdelite na učno in testno množico, dobljeno učno množico pa še nadalje razdelite na *dejansko*[1] učno množico in na validacijsko množico v razmerju 85 : 15. Obe delitvi opravite naključno s funkcijo `torch.utils.data.random_split()`. S pomočjo funkcije `count_samples_in_class()` preverite kakšno je število slik po kategorijah v posamezni množici.

[1] Izraz *dejanska* učna množica ni uveljavljen izraz. V tem primeru je uporabljen zato, da se poudari razliko med učno in validacijsko množico, ki sta dve različni množici. V literaturi lahko sicer včasih zasledimo, da se izraza *testna* in *validacijska* množica uporabljata izmenično - to je sicer nedosledno in nepriporočljivo, uporablja pa se predvsem, ko imamo v podatkovni bazi le malo podatkov.

Pred reševanjem naslednje naloge preverite:

- kolikšno je število slik v posamezni kategoriji (od 0 do 9), pomagajte si s funkcijo `count_samples_in_class()` in
- velikost posamezne slike.

2. Napišite funkcijo za prikaz sličic iz iz prenesene podatkovne baze. Prikaz naj omogoča prikaz večjega števila sličic na enem samem prikaznem oknu:

```
def showMultipleImages(nRows, nCols, iImages, iLabels, iPrediction=
    None):
    # ...
    return fig
```

kjer vhodni argument `nRows` predstavlja število vrstic, `nCols` število stolpcev v katere so razporejene sličice, `iImages` seznam sličic, `iLabels` seznam dejanskih števk, ki jih predstavljajo sličice v seznamu `iImages`, `iPrediction` pa seznam napovedanih števk. Če uporabnik ne poda seznama `iPrediction`, naj se v naslovu sličice izpiše le dejanska števka, ki jo prikazuje slika, v nasprotnem primeru pa naj se izpišeta obe dve števki. Izhodni argument `fig` naj predstavlja objekt prikaznega okna.

V petih vrstah in šestih stolpcih prikažite 30 sličic iz baze MNIST.

3. Definirajte arhitekturo globoke konvolucijske nevronske mreže. Mreža naj bo sestavljena iz:

   (a) prvega bloka, ki je sestavljen iz konvolucijske plasti (velikost konvolucijskega filtra naj bo 5×5, število filtrov 10, uporabite enak (ničelni) odmik (ang. padding)), ki naj mu sledi združevanje po največjem elementu (ang. maxpooling) z velikostjo 2 in korakom 2 in nato plast z ReLU aktivacijsko funkcijo,

   (b) plasti, kjer se zgodi združevanje po največjem elementu (ang. maxpooling) z velikostjo 2 in korakom 2,

   (c) drugega bloka, ki je povsem enak prvemu z razliko, da je število filtrov v konvolucijski plasti 20 ter da se takoj za konvolucijo doda še izpustni sloj (ang. dropout layer) z verjetnostjo izpusta 50 %,

   (d) polno povezanega nivoja, ki naj ima 50 izhodov, ki mu sledi ReLU aktivacijska funkcija,

   (e) plasti z izpustnim slojem, ki mu sledi polno povezani nivo z desetimi izhodi ter *log-softmax* aktivacijska funkcija.

   Določite še nastavitve učenja modela:

   - za optimizacijsko metodo izberite stohastični gradientni spust z momentom pri čemer moment nastavite na $0,5$[2],
   - začetna učna konstanta naj bo $0,01$,
   - število ponovitev (ang. epochs) naj bo 3,
   - uporabite naključno premešanje slik med ponovitvami,
   - določite kako pogosto naj se preveri napaka na validacijski množici (`val_log_epoch_interval`),

[2] Z uporabo momenta pri posodabljanju uteži v trenutni iteraciji upoštevamo tudi odvode iz prejšnje iteracije učenja pomnožene z utežjo w: $0 \leq w \leq 1$

- določite še ostale parametre, ki vam bodo pomagali pri spremljanju učenja modela.

4. Napišite funkcijo za izračun natančnosti razvrščanja:

```
def calculate_accuracy(gt, predicted):
    # ...
    return N_correct, acc
```

   kjer vhodni argument `gt` predstavlja seznam pravilnih števk, `predicted` pa seznam napovedanih števk. Izhodni argument `N_correct` naj predstavlja število pravilno razvrščenih slik, `acc` pa natačnost razvrščanja.

5. Dokončajte funkciji `train` in `inference` tako, da bo prva uporabna za učenje modela, druga pa za izračun napovedi modela.

   Začnite učenje nevronske mreže, izrišite potek vrednosti kriterijske funkcije in določite kako dobro model deluje na testni množici.

Dodatek

Odgovore na sledeče probleme ni potrebno prilagati k poročilu, prispevajo pa naj k boljšemu razumevanju vsebine.

1. Zmanjšujte število slik v učni množici in opazujte kaj se dogaja z učno krivuljo ter natančnostjo klasifikacije.

2. Spremenite arhitekturo nevronske mreže: spremenite lahko na primer število filtrov v posamezni plasti, izločite ali dodajte dodatne plasti v mrežo. Kaj se zgodi, če bistveno povečate število parametrov nevronske mreže?

3. Preizkusite se še na problemu klasifikacije slik iz baze *CIFAR-10*: preverite kako dobro uporabljeni model deluje za ta problem.