Drugi kolokvij

23. svibnja 2023. 19:04

Jednostavne neuronske mreže

```
import tensorflow as tf
from tensorflow import keras
import matplotlib.pyplot as plt
import numpy as np

#KOLOKVIJ morat znati dohvatiti podatke, uzeti npr prvih 50000 itd. !!!!


baza=keras.datasets.fashion_mnist
(X_train,y_train),(X_test,y_test)=baza.load_data()

X_train.shape  # za vidjeti koji input size (60000, 28, 28)

#spremiti nazive klasa u varijablu da pokaže naziv ne broj
class_names=["T-shirt/top","Trouser","Pullover","Ankle Boot"]
y_train[0] #ovo nam kaže oznaku klase u koju je svrstan 0ti element
class_names[y_train[0]]  #ako smo pisali nazive po redu izbaci odgovarajuće!

plt.imshow(X_train[0], cmap='gray') #prikaže sliku iz baze

X_train=X_train/255  #normaliziramo

#prvih 5000 iz training seta uzeti za validaciju a novi train neka bude od 5001.slike do kraja
X_valid=X_train[:5000]
X_train=X_train[5000:]
y_valid=y_train[:5000]
y_train=y_train[5000:]

#prvih 5 iz skupa za valid prikazati
for i in range (0,5):
  plt.subplot(1,5,i+1)
  plt.imshow(X_valid[i], cmap='gray')
  plt.title(class_names[y_valid[i]])
plt.show()
```

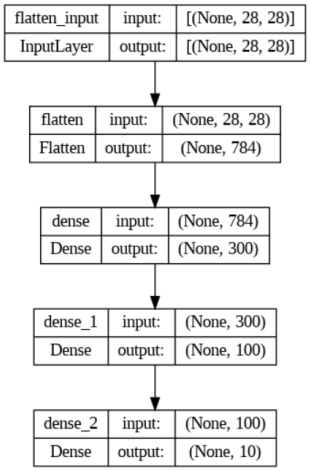
Definiranje modela - aktivacijske fije

```
model=keras.models.Sequential()
#definiramo 1. sloj=ULAZNI sloj, tu nema aktivacijske fije
model.add(keras.layers.Flatten(input_shape=[28,28]))  #flatten sloj poravnava sve, input_shape samo na prvom sloju definirati
#drugi sloj tj.prvi skriveni, potpuno povezani = DENSE sa 300 neurona
model.add(keras.layers.Dense(300, activation="relu"))
#drugi skriveni sloj
model.add(keras.layers.Dense(100, activation="relu"))
#ZADNJI sloj (klasifikacijski) - fija softmax, !!!ima onoliko neurona koliko imamo klasa = 10
model.add(keras.layers.Dense(10, activation="softmax"))

model.layers  #izlista slojeve
```

Prikaz arhitekture stvorenog modela

```
model.summary()  ->
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 300)               235500

 dense_1 (Dense)             (None, 100)               30100

 dense_2 (Dense)             (None, 10)                1010

=================================================================
Total params: 266,610
Trainable params: 266,610
Non-trainable params: 0
_________________________________________________________________
```



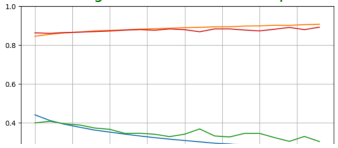

```
keras.utils.plot_model(model, show_shapes=True)  ->

hidden1=model.layers[1]
weights, bias= hidden1.get_weights()
weights  #vrijednosti dodijeljene za svaki neuron
```

Kompajliranje modela

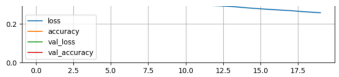
```
#ime modela.compile pa onda fija gubitka, metoda optimizacije i metrika (možemo ih više)
model.compile(loss="sparse_categorical_crossentropy", optimizer="sgd", metrics=["accuracy"])

#npr batch=5 znači da se nakon svakih 5 slika model ažurira, DEFAULT je 32
#broj epocha - broj potpunih prolazaka kroz training set, sadrži više batcha

#history da nam se spremi negdje
history=model.fit(X_train,y_train, batch_size=256, epochs=20, validation_data=(X_valid,y_valid)) #da vć nemamo valid podijeljeno
validation_split=0,2 -> 20% iz traininga uzme
#u loss vidimo kako se pogreška smanjuje sa svakom epohom, a u validaciji vidimo kako se preciznost povećava
```

Vizualiziranje

```
import pandas as pd
pd.DataFrame(history.history).plot(figsize=(8, 5))
plt.grid(True)
```



```
plt.ylim(0,1)
plt.show()   ->
```



Testiranje

```
model.evaluate(X_test,y_test)
predikcije=model.predict(X_test)       #reci su objekt, stupci su klase, gdje je 1 znači da za tu klasu smatra 100% da taj objekt
y_pred=np.argmax(predikcije,axis=-1)   #naći min i max za perdikcije i poredati ih? ugl onda nam ispiše po redu kamo bi svrstao k


#PRVIH 5 iz skupa za test
for i in range (0,5):
  plt.subplot(1,5,i+1)
  plt.imshow(X_test[i], cmap='gray')
  plt.title(class_names[y_pred[i]]) #što je predvidjelo
plt.show()


from sklearn.metrics import classification_report
print(classification_report(y_test, y_pred))
```

Konvolucijske neuronske mreže

skriveni slojevi= konvolucijski + slojevi sažimanja + potpuno povezani?
slojevi sažimanja - da bi se smanjila veličina

```
from keras.datasets import cifar10
(X_train,y_train),(X_test,y_test)=cifar10.load_data()

X_train.shape    (50000, 32, 32, 3) #50000 slika, 32 x 32 px, dubine 3


print('oznaka 1. slike:',y_train[0])
```

One-hot encoding

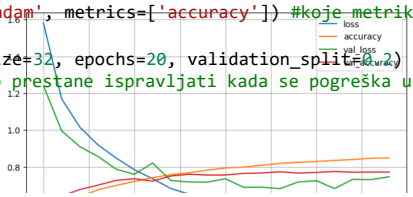
```
y_train_one_hot=keras.utils.to_categorical(y_train, 10)  #što da nam pretvori i koliko klasa je ukupno(10)
y_test_one_hot=keras.utils.to_categorical(y_test,10)


#normalizacija
X_train=X_train/255
X_test=X_test/255
```

Kreiranje konv. neur. mreže

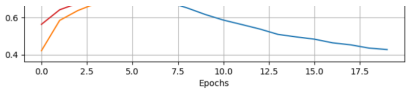
```
model=keras.models.Sequential()

#prvi konv. sloj, 32 neurona, filter 2D- dubine 32 i da bude velik 3x3 px, aktiv.fija relu, u prvom sloju definirati i input_shap
dubina
model.add(keras.layers.Conv2D(32,(3,3), activation='relu',padding='same',input_shape=(32,32,3)))
#još jedan takav, ne treba više input_shape
model.add(keras.layers.Conv2D(32,(3,3), activation='relu',padding='same'))
#maxPooling layer -> sloj sažimanja
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))
#neki postotak neurona odbacimo da bi model bio manje osjetljiv na promjene u ulaznim podacima npr. čovjek je iako mu fali jedna
#npr. ako je 0,25 -> u svakom prolasku 25% neurona se odbaci iz treniranja
#koji je rate za dropanje - 0.25
model.add(keras.layers.Dropout(0.25))

#još dva konvolucijska, jedan sažimanje i dropout (tj.5.-8. sloj našeg modela)
model.add(keras.layers.Conv2D(64,(3,3), activation='relu',padding='same'))
model.add(keras.layers.Conv2D(64,(3,3), activation='relu',padding='same'))
model.add(keras.layers.MaxPooling2D(pool_size=(2,2)))
model.add(keras.layers.Dropout(0.25))


# trebamo jedan sloj za flatten
# jedan potpuno povezani sloj sa 512 neurona
# dropout sa 0.5
#još jeda npotpuno povezani sa brojem klasa -> zadnji
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(512, activation='relu')
model.add(keras.layers.Dropout(0.5))
model.add(keras.layers.Dense(10, activation='softmax'))  #zadnji= aktiv fija softmax, 10 neurona=10 klasa

model.summary()
```

Treniranje

```
model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy']) #koje metrike želimo pratiti

povijest_modela=model.fit(X_train,y_train_one_hot, batch_size=32, epochs=20, validation_split=0.2)
#možemo dodati early_stop onda ne izvrti svih 20 epoha nego prestane ispravljati kada se pogreška ustali; 0.2= 20% podataka uzme


import pandas as pd
pd.DataFrame(povijest_modela.history).plot(figsize=(8,5))
```



```
plt.grid(True)
plt.xlabel('Epochs')
plt.show
```




```
model.save('my_cifar_model.h5')

#testiranje
model.evaluate(x_test,y_test_one_hot)
```

Testiranje na vlastitim slikama

```
!curl -o macka.jpg https://upload.wikimedia.org/wikipedia/commons/thumb/3/3a/Cat03.jpg/300px-Cat03.jpg
slika = plt.imread("/content/macka.jpg")
slika.shape     (300, 300, 3)

from skimage.transform import resize
slika_resized = resize(slika, (32,32))
plt.imshow(slika_resized)

vjerojatnosti = model.predict(np.array( [slika_resized,] ))

broj_u_klase=['avion','automobil','ptica','mačka','jelen','pas','zaba','konj','brod', 'kamion']
index = np.argsort(vjerojatnosti[0,:])
for i in range (9,5,-1):  #prvih nekoliko vjerojatnosti
  print(broj_u_klase[index[i]], ":", vjerojatnosti[0,index[i]])
```

Autoencoderi

```
#broj neurona u skrivenim slojevima se "komprimira" - bottleneck (prije je bilo više) -> na temelju toga rekonstruiramo nešto štc
originalu i istih dimenzija
#npr. ako imamo zrnastu sliku da ju izgladimo, slika veće rezolucije i sl.
#dobri su za anomalije, dobro rade s podacima koji su različiti
#na outputu imamo onoliko neurona koliko imamo na ulazu
# nemamo y jer nemamo klase jer ne predviđamo nego samo od podataka pokušavamo nešto popraviti??
```

To maximize performance, minimize the loss that is, encoders and decoders are typically symmetrical together. Naturally, the input size is ec size of an autoencoder.

Autoencoders always have less input neurons in the middle layer than in the input and output layer. This is called the **bottleneck**. If it weren' bottleneck, the autoencoders could just copy this data over from the input to the output layer without compressing it.

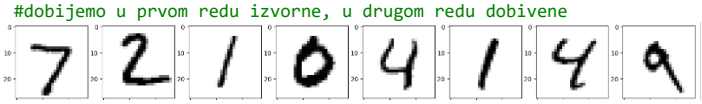
```
#možemo ali ne moramo spremiti y vrijednosti - za autoencoder nam ne trebaju oznake spremljene u y
(x_train,_),(x_test,_)=keras.datasets.mnist.load_data()

#normalizirati između 0 i 1
x_train=x_train/255
x_test=x_test/255
#definirali smo podatke nad kojima radimo
```

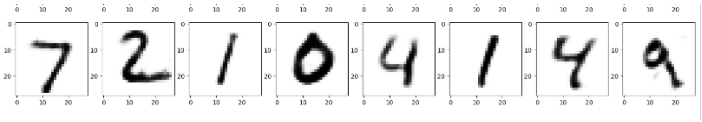
Definiranje modela - encoder + decoder

```
encoder=keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28,28]),   #možemo odmah def slojeve koji nam trebaju; 1.
    keras.layers.Dense(100, activation="relu"),
    keras.layers.Dense(30, activation="relu") ,  #ako tu završavamo onda je to BOTTLENECK - "najuži" , decoder isto samo obrnuto(
> dobijemo Ii..iI
])

decoder=keras.models.Sequential([
    keras.layers.Dense(100, activation="relu", input_shape=[30]),   input shape je x.shape pa vidimo? provjeriti
    keras.layers.Dense(28*28, activation="sigmoid") ,
    keras.layers.Reshape([28,28])      #sloj koji vraća u oblik (u ovom slučaju 28,28)
])


#sada ih spojimo - to je naš model(naziv modela = stacked_autoencoder)
stacked_autoencoder=keras.models.Sequential([encoder,decoder])

#sada isto kao i prošli put, kompajler:
stacked_autoencoder.compile(loss="binary_crossentropy", optimizer='adam', metrics=['accuracy'])  #metrike opcionalno

history=stacked_autoencoder.fit(x_train,x_train,epochs=10, validation_data=[x_test,x_test])
#history-varijabla gdje spremamo podatke o treniranju;; self learning - uči sam od svojih podataka-> umjesto y_train ide x_train
```

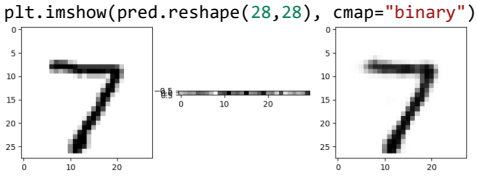
Grafički prikaz

```
#prikazati original i dobiveno treniranjem prvih 8
plt.figure(figsize=(20,5))   #nešto nebitno, da ne ispadne skroz ružno
for i in range(8):
  plt.subplot(2,8,i+1)
  plt.imshow(x_test[i],cmap="binary")  #prvo originale
  plt.subplot(2,8,8+1+i)  #sada dobivene ali da krene od 9 pozicije na dalje
  pred=stacked_autoencoder.predict(x_test[i].reshape(1,28,28))  # 1=jednu sliku na 28x28 (da ih imamo 600 bilo bi (600,28,28) ali
zato samo trenutnu)
  plt.imshow(pred.reshape(28,28),cmap="binary")
  #dobijemo u prvom redu izvorne, u drugom redu dobivene
```



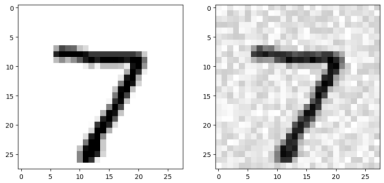


```
#prikazati za prvu original i što smo dobili i što smo reshapaly
plt.figure(figsize=(10,5))  #definiramo veličine, nebitan dio?
plt.subplot(1,3,1)   #1 redak, 3 stupca, 1.pozicija
plt.imshow(x_test[0], cmap="binary")
plt.subplot(1,3,2)  # na drugu poziciju crtaj:
latent_vector=encoder.predict(x_test[0].reshape(1,28,28))    #komprimirani kod je latent vector - vizualno bottleneck, skoro pa c
subplotu
plt.imshow(latent_vector, cmap="binary")
plt.subplot(1,3,3)   #sada ono što je predvidio sa dekoderom na temelju latent vectora, na trećoj poziciji subplota
pred=decoder.predict(latent_vector)
plt.imshow(pred.reshape(28,28), cmap="binary")
```



Dodavanje šuma

```
plt.figure(figsize=(10,5))
plt.subplot(1,2,1)
plt.imshow(x_test[0], cmap="binary")
plt.subplot(1,2,2)
noise=np.random.random((28,28))/4  #random šum da dobijemo neku zrnatiju sliku? /4 da ipak ne bude prevelik
plt.imshow(x_test[0]+noise, cmap="binary")
```



```
#definiranje novog npr. samo dodavanjem dodatnok fully connected layera u encoder i decoder
encoder2=keras.models.Sequential([
    keras.layers.Flatten(input_shape=[28,28]),            #možemo odmah def slojeve koji nam trebaju; 1.
    keras.layers.Dense(100, activation="relu"),
    keras.layers.Dense(100, activation="relu"), #####
    keras.layers.Dense(30, activation="relu") ,  #ako tu završavamo onda je to BOTTLENECK - "najuži" , decoder isto samo obrnuto(
> dobijemo Ii..iI

decoder2=keras.models.Sequential([
    keras.layers.Dense(100, activation="relu", input_shape=[30]),
    keras.layers.Dense(100, activation="relu"), ####
    keras.layers.Dense(28*28, activation="sigmoid") ,
    keras.layers.Reshape([28,28])      #sloj koji vraća u oblik (u ovom slučaju 28,28)
])

stacked_autoencoder2=keras.models.Sequential([encoder2,decoder2])
stacked_autoencoder2.compile(loss="binary_crossentropy", optimizer='adam',metrics=['accuracy'])

x_train_noise=x_train+((np.random.random(x_train.shape))/4) #šum
x_test_noise=x_test+((np.random.random(x_test.shape))/4) #šum

history=stacked_autoencoder2.fit(x_train_noise,x_train,epochs=10, validation_data=[x_test_noise,x_test])  #provjeriti je li dobrc
u fit


plt.figure(figsize=(10,5))
#ista for petlja kao i prije s novim podacima
for i in range(8):
  plt.subplot(2,8,i+1)
  plt.imshow(x_test_noise[i],cmap="binary")
  plt.subplot(2,8,8+1+i)
  pred=stacked_autoencoder2.predict(x_test_noise[i].reshape(1,28,28))
  plt.imshow(pred.reshape(28,28),cmap="binary")
```