SVEUČILIŠTE U ZAGREBU

FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA

SEMINAR

Pythonske biblioteke za ostvarivanje računalnog vida i strojnog učenja

Krešimir Kralj

Voditelj: ***Siniša Šegvić***

Zagreb, svibanj 2016.

Sadržaj

1. Uvod

1.1 Uvod u računalni vid

Računalni vid je područje umjetne inteligencije koje obuhvaća metode za dohvaćanje, obradu, analiziranje i razumijevanje slike te, općenito, više dimenzionalnih podataka iz realnog svijeta u cilju dobivanja numeričkih ili simboličkih informacija. Računalni vid je povezan s mnogim disciplinama, a neke od njih su: umjetna inteligencija, neurobiologija, fizika poluvodiča te obrada signala, a ujedno je i sam disciplina s brojnim primjenama. Glavni cilj računalnog vida je modelirati, replicirati te unaprijediti ljudski vid koristeći računalne programe i sklopovlje različitih razina.

Najvažniji zadaci računalnog vida su:

1) Prepoznavanje - utvrđivanje sadrži li slika određeni objekt, značajku ili aktivnost.

2) Analiza pokreta - analiza gdje je niz slika obrađen za procjenu brzine u nekom trenutku na slici ili u 3D sceni, ili čak i od kamere koja proizvodi sliku.

3) Rekonstrukcija događaja i slika - s obzirom slike u sceni ili s obzirom na video, rekonstrukcija događaja ima za cilj izračunavanje 3D modela scene. Cilj restauracije slike je uklanjanje smetnji iz slika.

1.2 Uvod u strojno učenje

Strojno učenje odnosi se na izgradnju računalnih sustava koji automatski poboljšavaju svoje performanse kroz iskustvo. Ideja takvog pristupa temelji se na proučavanju i izgradnji algoritama koji su sposobni zaključivati i donositi pretpostavke s obzirom na dane podatke. Duboko učenje jedna je od grana strojnog učenja, a temelji se na skupu algoritama koji pokušavaju transformirati apstraktne objekte u podatke, korištenjem složenih struktura i transformacija.

Zadaće strojnog učenja najčešće se dijele u tri široke kategorije:

1) Nadzirano učenje - Računalu je dan ulaz i željeni izlaz, a cilj je pronaći pravila koja mapiraju ulaze u izlaze.

2) Nenadzirano učenje - Predani su podaci bez ciljne vrijednosti, treba pronaći pravilnost u podacima.

3) Podržano/ojačano učenje - Učenje optimalne strategije na temelju pokušaja s odgođenom nagradom. Računalni program mora izvršiti određeni cilj bez da mu se sugerira je li došao blizu cilja.

2. Biblioteka NumPy

NumPy je jedan od glavnih znanstvenih paketa implementiranih za programski jezik Python, koji pruža podršku za upravljanje opširnim, više dimenzionalnim nizovima i matricama, te implementira široki spektar matematičkih funkcija za efikasno upravljanje nad tim poljima.

Osnovna funkcionalnost biblioteke Numpy je struktura podataka koja predstavlja n-dimenzionalno, homogeno polje, "ndarray". Zbog toga što je "ndarray" homogena kolekcija sastavljena od objekata istog tipa, svaki blok memorije u polju se interpretira na potpuno jednak način. Ovakav način implementacije strukture podataka jedan je od glavnih razloga vrlo velike efikasnosti algoritama implementiranih u NumPy bibliotekama. Neke od najvažnijih operacija koje se mogu izvršavati nad poljem uključuju: množenje i transponiranje matrica, rješavanje sustava jednadžbi, množenje i normalizacija vektora. Svaka od ovih operacija koristi se pri modeliranju, obradi i razvrstavanju slika. U sklopu računalnog vida, biblioteka NumPy je od presudne važnosti jer se slike prevode u skupove podataka koji se potom spremaju i obrađuju kao NumPy polja. Dakle, neke od najpoznatijih biblioteka koje implementiraju algoritme za obradu slike oslanjaju se na to da se ta slika obrađuje kao "ndarray" polje.

Koristeći biblioteku NumPy, sliku možemo opisati kao višedimenzionalno polje. Pretpostavimo da imamo sliku veličine 452x589 piksela u RGB prostoru boja gdje se svaki piksel sastoji od tri komponente: vrijednosti crvene, zelene i plave boje. Svaka vrijednost boje je u intervalu [0, 255]. Tada imamo matricu 452x589 takvih piksela. NumPy nam omogućava da vrlo efikasno spremimo takvu sliku u niz (452, 589, 3), koji ima 452 retka, 589 stupaca i 3 vrijednosti, po jednu za svaki piksel u RGB prostoru boja.

Efikasnost izvođena te jednostavnost korištenja su glavni razlozi korištenja NumPy biblioteke za obradu slika.

2.1 Stvaranje polja i osnovne operacije

Primjer demonstrira stvaranje polja te neke od osnovnih operacija koje pruža NumPy biblioteka. Polje je moguće stvoriti pozivom *array funkcije,* kojoj se kao argumenti predaju željeni elementi toga polja. Dodatnim argumetnom *dtype* moguće je specifirati tip podataka koji će biti spremljen u novostvorenom polju. Funkcija *arrange* generira polje čiji su elementi u rasponu koji je predan kao argument te funkcije. Pozivom funkcije *linspace* stvara se novo polje koje sadrži *n* jednako udaljenih brojeva. NumPy podržava i aritmetičke operacije nad nizovima, pa je oduzimanje nizova dozvoljena operacija, pod uvjetom da su nizovi međusobno kompatibilni. Funkcija *flatten* pretvara matricu proizvoljnih dimenzija u jednodimenzionalan niz, kopirajući elemente po recima. Funkcijom *repeat* moguće je umnožiti retke i stupce matrice određen broj puta. Ako zadano polje ima puno dimenzija, za iteriranje po takvom polju koristi se funkcija *ndindex* koja služi kao jednostavnija zamjena za ugniježđene petlje. Ulazno izlazne operacije omogućene su funkcijama *toFile* za pisanje te *fromFile* za čitanje zapisanog niza iz datoteke na disku.

Primjer izvođenja osnovnih operacija biblioteke NumPy

```
>>> import numpy as np
>>> x = np.array([127, 2, 3], dtype=np.int8)
>>> y = x.copy().astype(float)
>>> y[0] = 5.123
>>> y
array([ 5.123,  2.  ,  3.  ])
>>> multi = np.array([[1, 2, 3], [4, 5, 9], [7, 8, 9]])
>>> np.argmax(multi)
5
>>> multi.flatten()
array([1, 2, 3, 4, 5, 9, 7, 8, 9])
>>> x = np.array([[1,2],[3,4]])
>>> np.repeat(x, 2)
array([1, 1, 2, 2, 3, 3, 4, 4])
>>> np.repeat(x, 2, axis=0)
array([[1, 2],
       [1, 2],
       [3, 4],
       [3, 4]])
>>> np.repeat(x, 2, axis=1)
array([[1, 1, 2, 2],
       [3, 3, 4, 4]])
```

```
>>> for i in np.ndindex(x.shape):
... x[i] += 5
>>> x
array([[6, 7],
       [8, 9]])
>>> multi[:]=0
>>> multi[1:2,0:1]=5
>>> multi
array([[0, 0, 0],
       [5, 0, 0],
       [0, 0, 0]])
>>> b=multi.copy()
>>> b[(b < 10) & (b > 1)] = 8
>>> b[[0, 1, 0], [1, 2, 0]] = 2
>>> b[1, 1] = 1
>>> b
array([[2, 2, 0],
       [8, 1, 2],
       [0, 0, 0]])
>>> np.arange(10)
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
>>> a = np.array([1, 2, 3, 6])
>>> b = np.linspace(0, 2, 4)
>>> a - b
array([ 1.        ,  1.33333333,  1.66666667,  4.        ])
>>> a**3
array([  1,   8,  27, 216])
>>> dim = np.eye(4)
>>> np.nonzero(dim)
(array([0, 1, 2, 3]), array([0, 1, 2, 3]))
>>> multi.tofile("output")
>>> fname = open("output", "rb")
>>> np.fromfile(fname, dtype=np.int)
array([0, 0, 0, 5, 0, 0, 0, 0, 0])
```

2.2 Linearna algebra

Biblioteka NumPy pruža velik broj funkcija za rad s dvodimenzionalnim poljem, što je od velike koristi pri obradi i analizi slika. Korištenje nekih od osnovnih funkcija linearne algebre nad dvodimenzionalnim poljem je prikazano u nastavku.

```
>>> from numpy.random import rand
>>> from numpy.linalg import solve, inv
>>> a = np.array([[1, 2, 3], [3, 4, 6.7], [5, 9.0, 5]])
>>> a.transpose()
array([[ 1. ,  3. ,  5. ],
       [ 2. ,  4. ,  9. ],
       [ 3. ,  6.7,  5. ]])
>>> inv(a)
array([[-2.27683616,  0.96045198,  0.07909605],
       [ 1.04519774, -0.56497175,  0.1299435 ],
       [ 0.39548023,  0.05649718, -0.11299435]])
>>> b =  np.array([3, 2, 1])
>>> solve(a, b)  # solve the equation ax = b
array([-4.83050847,  2.13559322,  1.18644068])
>>> c = rand(3, 3)  # create a 3x3 random matrix
>>> c
array([[ 3.98732789,   2.47702609,   4.71167924],
       [  9.24410671,   5.5240412 ,  10.6468792 ],
       [ 10.38136661,   8.44968437,  15.17639591]])
>>> np.dot(a, c)  # matrix multiplication
array([[  53.61964114,   38.8741616 ,   71.53462537],
       [ 118.4935668 ,   86.14012835,  158.40440712],
       [ 155.04043289,  104.3499231 ,  195.26228855]])
```

Primjer izvođenja operacija linearne algebre sadržanih u biblioteci NumPy

3. Kolekcija biblioteka SciPy

SciPy je skup biblioteka otvorenog koda, najčešće korištena među znanstvenicima, analitičarima i inženjerima koji se bave računalnim izračunima i obradom informacija. SciPy, između ostalog, sadrži biblioteke za obradu slika i signala. Algoritmi za obradu korišteni u tim bibliotekama temelje se na NumPy polju i njegovoj implementaciji.

Neki od osnovnih biblioteka sadržanih u skupu SciPy su:

1. NumPy biblioteka - osnovni paket korišten za numeričko računanje. Implementira metode za upravljanje i obradu podataka nad numeričkim poljem i matricama.
2. SciPy biblioteka - jedna od osnovnih biblioteka koju čini SciPy skup. Sadrži skup algoritama korištenih za obradu signala, optimizaciju, statistiku te, između ostalog, pruža efikasne metode za numeričko integriranje.
3. Matplotlib biblioteka – omogućuje crtanje dvodimenzionalnih te nekih osnovnih trodimenzionalnih objekata na vrlo jednostavan način.
4. IPython – bogato interaktivno sučelje koje omogućuje brzu obradu i vizualiziranje podataka.
5. SymPy – biblioteka koja pruža podršku za simboličku matematiku i računalnu algebru.
6. Pandas – biblioteka otvorenog koda koja omogućuje korisniku da na vrlo jednostavan način upravlja vrlo efikasnim strukturama podataka te alatima za analizu podataka.

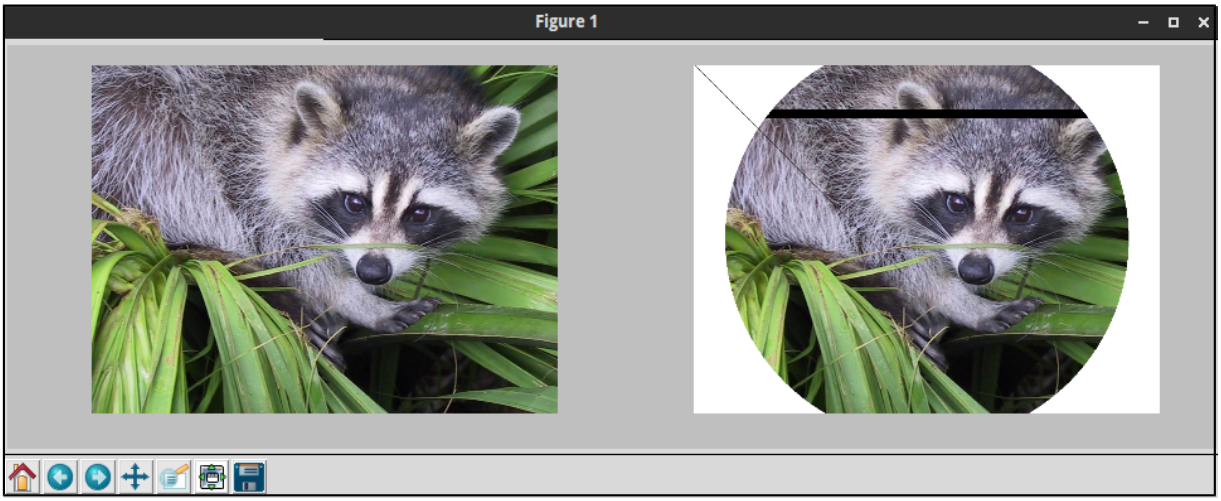
3.1 Manipulacija i obrada slike koristeći SciPy

Cilj ovog primjera je demonstrirati neke od osnovnih funkcija biblioteke SciPy korištenih pri obradi slika.

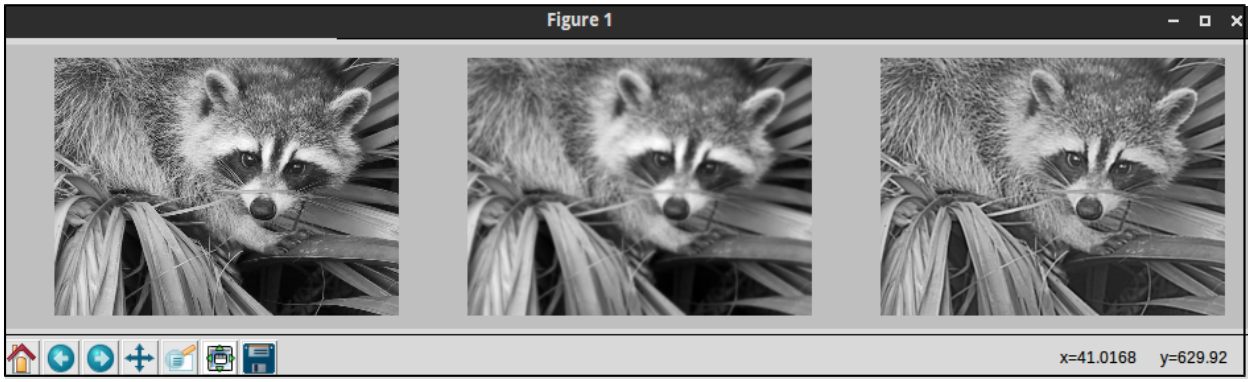
```
1. from scipy import misc, ndimage
2. import numpy as np, scipy
3.
4. face = misc.imread('face.png')
5.
6. face[100:120] = 0
7. lx, ly, lz = face.shape
8. X, Y = np.ogrid[0:lx, 0:ly]
9. mask = (X - lx/2)**2 + (Y - ly/2)**2 > lx*ly/4
10. face[mask] = 255
11. face[range(500), range(500)] = 0
12.
13. face.mean() # 147.932189094
14. face.max(), face.min() # 255, 0
15.
16. #Blurring/smoothing
17. fgray = np.dot(face[...,:3], [0.299, 0.587, 0.114]) # Y' = 0.299 R + 0.587 G + 0.114 B
18. blurred_face = ndimage.gaussian_filter(fgray, sigma=3)
19. very_blurred = ndimage.gaussian_filter(fgray, sigma=5)
20.
21. #Sharpening
22. f = scipy.misc.face(gray=True).astype(float)
23. blurred_f = ndimage.gaussian_filter(f, 3)
24.
25. filter_blurred_f = ndimage.gaussian_filter(blurred_f, 1)
26.
27. alpha = 30
28. sharpened = blurred_f + alpha * (blurred_f - filter_blurred_f)
```

Primjer 1. manipulacija slike

Primjer 1. započinje učitavanjem slike *PNG* formata sa diska računala. Nakon izvršavanja linije *4*, varijabla face sadrži *NumPy* objekt koji reprezentira učitanu sliku pomoću *ndarray* polja. Prva obrada slike kreće modificiranjem polja face u liniji 6. Sve koordinate polja u kojima je redak između linija 100 i 120 su obojene u crnu boju. U linijama 6 – 11 unutar slike se stvara kružnica te se svi pikseli izvan kružnice postavljaju u bijelu boju. Konačno, završetkom linije 11, u modificiranoj se slici iscrtava dijagonala crne boje počevši od koordinate (0, 0), pa sve do koordinate (500, 500). Redci 13 i 14 pokazuju dvije metode NumPy biblioteke. Funkcija mean() računa srednju vrijednost boja piksela cijele slike, dok metode min() i max() vraćaju najmanju i najveću boju piksela koja postoji u traženoj slici. Linije 17 – 19 demonstriraju promjenu zamućenja i zaglađivanja slike koristeći Gaussov filter. U retku 17 trenutna slika se boja u sivo te se potom zamućuje u redcima 18 i 19. Stupanj zamućenja predaje se kao jedan od parametara funkcije za izračun Gaussovog filtra. Konačno, u linijama 22 – 28 zamućenu sliku ponovno izoštravamo.



Izgled slike nakon izvođenja 11. linije.



Prikaz slika u trenutcima izvođenja 11 – 28. linije. Prva slika prikazuje originalnu sliku obojanu u sivo, druga slika je dobivena postupkom zamućivanja, a treća je nastala iz druge postupkom izoštravanja.

4. Skup alata SciKit

Skup alata SciKit (skraćeno od „SciPy Toolkits”), nadograđuje se na skup paketa SciPy. Dva alata iz tog skupa su posebno zanimljiva u sklopu računalnog vida i strojnog učenja: SciKit-image te SciKit-learn.

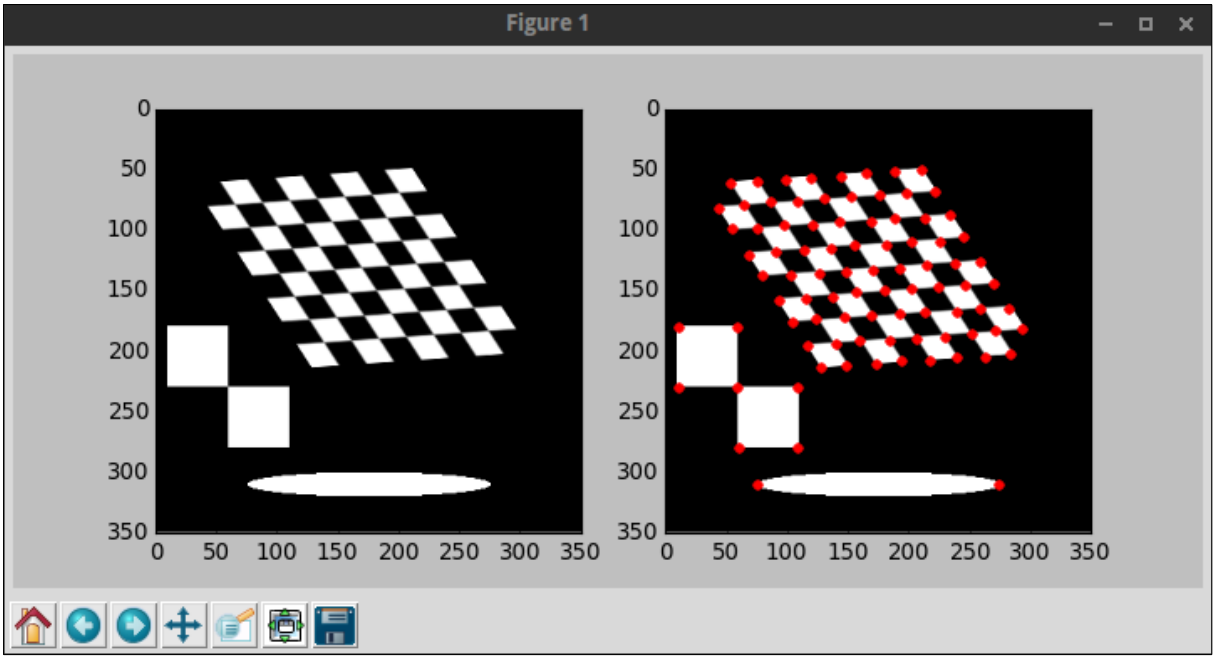
4.1 SciKit-image biblioteka

Biblioteku SciKit-image čini kolekcija algoritama namijenjenih obradi slika.

Neki od algoritama sadržanih u sklopu ove biblioteke su: algoritmi za segmentaciju, geometrijske transformacije, manipuliranje bojama i prostorom, analizu, filtriranje, morfološku analizu, detekciju obilježja i slično.

4.1.1 Detekcija rubova korištenjem SciKit-image biblioteke

*Primjer ilustrira primjenu biblioteke SciK*it-image u detekciji rubova određene slike. Primjer započinje stvaranjem slike koja će se koristiti za provjeru rezultata. U linijama 7. i 8. učitavamo oglednu sliku te ju skaliramo, rotiramo i bojamo u sivo koristeći funkcije warp i AffineTransform iz Skimage biblioteke. Na dobivenu sliku dodajemo elipsu i kvadrate u linijama 10 i 11. Nakon što smo stvorili željenu sliku, u liniji 16. pozivamo Harrisov algoritam za detekciju rubova, koji vraća skup koordinatnih točaka koje su najvjerojatnije rubovi. U linijama 18 – 24 koristimo Matplotlib iz biblioteke SciPy da bi prikazali rezultat.



Rezultat izvođenja Harrisovog algoritma, svi pronađeni rubovi su označeni crvenim točkama.

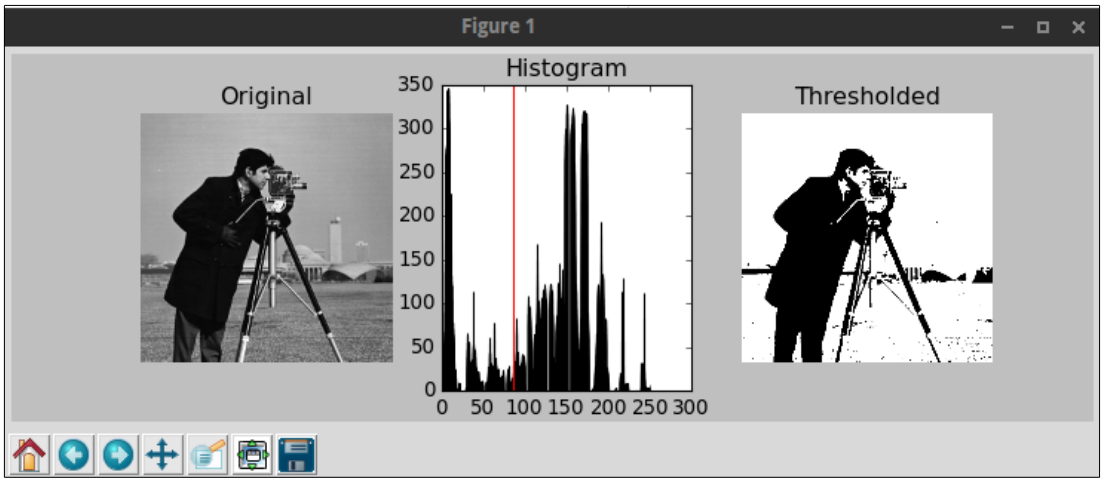
```
1. from matplotlib import pyplot as plt
2. from skimage import data
3. from skimage.feature import corner_harris, corner_peaks
4. from skimage.transform import warp, AffineTransform
5. from skimage.draw import ellipse
6.
7. tform = AffineTransform(scale=(0.9, 0.9), rotation=1, shear=0.5, translation=(210, 50))
8. image = warp(data.checkerboard(), tform.inverse, output_shape=(350, 350), preserve_range=True)
9.
10. rr, cc = ellipse(310, 175, 10, 100)
11. image[rr, cc] = 255
12.
13. image[180:230, 10:60] = 255
14. image[230:280, 60:110] = 255
15.
16. coords = corner_peaks(corner_harris(image), min_distance=5)
17.
18. plt.subplot(121)
19. plt.imshow(image, cmap=plt.cm.gray)
20. plt.subplot(122)
21. plt.imshow(image, cmap=plt.cm.gray)
22. plt.plot(coords[:, 1], coords[:, 0], '.r', markersize=10)
23. plt.axis((0, 350, 350, 0))
24. plt.show()
```

4.1.1 Detekcija rubova korištienjem Harrisovog algoritma

4.1.2 Kreiranje binarne slike korištenjem metode optimalnog praga

Metoda optimalnog praga koristi se za kreiranje binarne slike. Prag određuje koliko se tonalna vrijednost piksela mora razlikovati da bi se smatrao kontrastnim prijelazom i bio uključen u djelovanje filtera. Slika je binarna ako ima samo dvije moguće vrijednosti za svaki piksel. Najčešće su to vrijednosti 0 i 1, jer se tada jedan bit može koristiti za reprezentaciju piksela. U ovom primjeru koristi se Otsuova metoda za računanje optimalnog praga.

```
 1. import matplotlib.pyplot as plt
 2. from skimage.data import camera
 3. from skimage.filters import threshold_otsu
 4.
 5. image = camera()
 6. thresh = threshold_otsu(image)
 7. binary = image > thresh
 8.
 9. fig = plt.figure(figsize=(12, 4))
10. ax1 = plt.subplot(1, 3, 1, adjustable='box-forced')
11. ax2 = plt.subplot(1, 3, 2)
12. ax3 = plt.subplot(1, 3, 3, sharex=ax1, sharey=ax1, adjustable='box-forced')
13.
14. ax1.imshow(image, cmap=plt.cm.gray)
15. ax1.set_title('Original')
16. ax1.axis('off')
17.
18. ax2.hist(image)
19. ax2.set_title('Histogram')
20. ax2.axvline(thresh, color='r')
21.
22. ax3.imshow(binary, cmap=plt.cm.gray)
23. ax3.set_title('Thresholded')
24. ax3.axis('off')
25.
26. plt.show()
```



Rezultat izvođenja Otsuovog algoritma

4.2 SciKit-learn biblioteka

SciKit-learn je biblioteka otvorenog koda koja implementira širok spektar funkcija strojnog učenja u programskom jeziku Python. Biblioteka se zasniva na NumPy, SciPy te Matplotlib bibliotekama.

Najbitniji skupovi operacija koje scikit-learn biblioteka pruža su:

1) Klasifikacija – utvrđivanje kategorije kojoj objekt pripada

2) Regresija – aproksimiranje nepoznate funkcije vezane za neki objekt.

3) Grupiranje – raspoređivanje sličnih objekata u skupove

4) Kompresija podataka – projekcija podataka u niže-dimenzionalne prostore

5) Odabir modela – usporedba, potvrđivanje te izbor parametara i modela

6) Predobrada podataka – organiziranje, čišćenje i filtriranje podataka

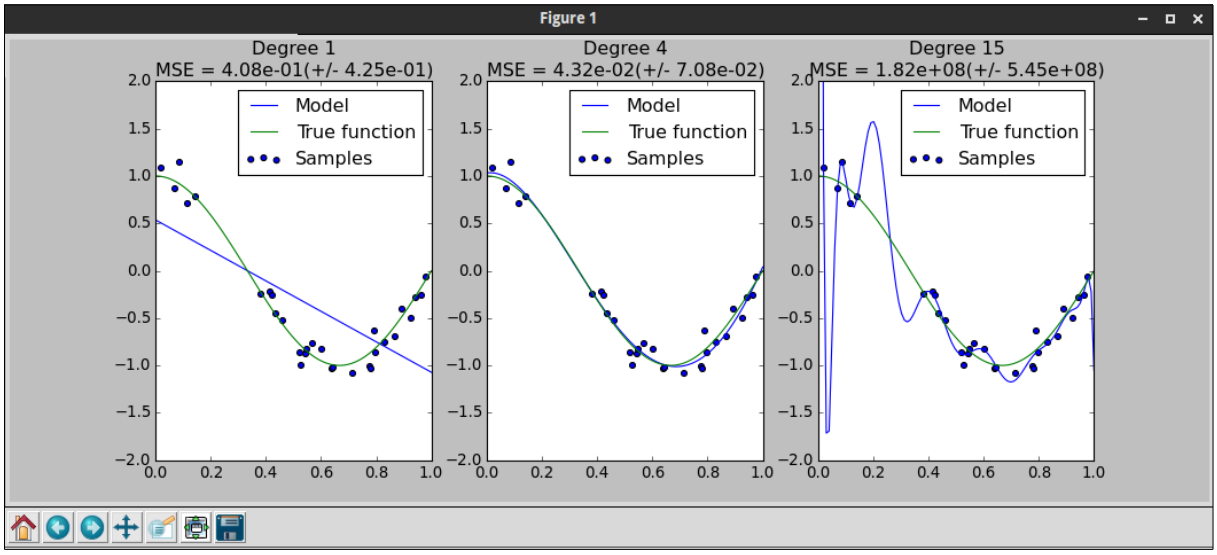
Reprezentacija podataka u biblioteci SciKit-learn

Strojno učenje temelji se na stvaranju modela iz podataka. Iz tog razloga, važno je razumjeti kako se ti podaci prezentiraju da bi ih računalo moglo obrađivati. Većina algoritama strojnog učenja implementirana u biblioteci scikit-learn očekuje da su podaci spremljeni u dvodimenzionalno polje, čije su dimenzije oblika (n_uzoraka, n_karakteristika). Takva dvodimenzionalna polja mogu biti ili NumPy polja ili, u nekim slučajevima, *scipy.sparse* matrice koja može biti memorijski efikasnija.

4.2.1 Problem prenaučenosti i podnaučenosti

Ovaj primjer demonstrira probleme prenaučenosti i podnaučenosti koji nastaju pri aproksimiranju nelinearne funkcije korištenjem metode linearne regresije. Zelenom bojom na grafovima je prikazana funkciju koju želimo aproksimirati, ona je dio funkcije kosinus. Nadalje, plavom bojom su prikazane aproksimacije tražene funkcije koristeći tri različita modela. Modeli imaju svojstva polinoma različitih stupnjeva. Očito

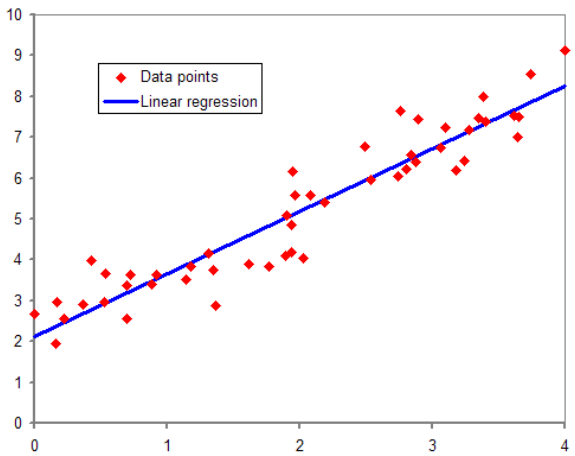
je kako linearna funkcija (polinom prvog stupnja) nije dovoljna da bi pristajala traženoj funkciji. Ovakav polinom nastao je zbog svojstva podnaučenosti aproksimacije. Polinom četvrtog stupnja aproksimira pravu funkciju gotovo savršeno. Polinomi većeg stupnja nastali aproksimacijom se ne preklapaju sa originalnom funkcijom tako dobro kao polinom četvrtog stupnja. Ovakav slučaj naziva se prenaučenost. Optimalno rješenje ovog problema dobiva se korištenjem metode unakrsne provjere. Izvorni kod prikazan je u nastavku.



Rezultat aproksimacije za polinome prvog, četvrtog i petnaestog stupnja (redom s lijeva na desno).

Metoda linearne regresije

Linearna regresija se odnosi na svaki pristup modeliranju relacija između jedne ili više varijabli označene sa Y, te jedne ili više varijabli označene sa X, na način da takav model linearno ovisi o nepoznatim parametrima estimiranih iz podataka. Razlog korištenja linearne regresije je taj da se modeli koji linerano ovise o svojim nepoznatim parametrima lakše modeliraju nego modeli sa nelinearnom ovisnošću o parametrima. U nastavku je prikazan primjer linearne regresije s jednom nezavisnom varijablom.



```
import numpy as np, matplotlib.pyplot as plt
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
from sklearn import cross_validation

np.random.seed(0)
n_samples = 30
degrees = [1, 4, 15]
true_fun = lambda X: np.cos(1.5 * np.pi * X)
X = np.sort(np.random.rand(n_samples))
y = true_fun(X) + np.random.randn(n_samples) * 0.1

plt.figure(figsize=(14, 5))
for i in range(len(degrees)):
    plt.subplot(1, len(degrees), i + 1)

   polynomial_features = PolynomialFeatures(degree=degrees[i],
                                            include_bias=False)
    linear_regression = LinearRegression()
    pipeline = Pipeline([("polynomial_features", polynomial_features),
                         ("linear_regression", linear_regression)])
    pipeline.fit(X[:, np.newaxis], y)

    # Evaluate the models using crossvalidation
    scores = cross_validation.cross_val_score(pipeline,
        X[:, np.newaxis], y, scoring="mean_squared_error", cv=10)
    X_test = np.linspace(0, 1, 100)
    plt.plot(X_test, pipeline.predict(X_test[:, np.newaxis]), label="Model")
    plt.plot(X_test, true_fun(X_test), label="True function")
    plt.scatter(X, y, label="Samples")
    plt.title("Degree {}\nMSE = {:.2e}(+/- {:.2e})".format(
        degrees[i], -scores.mean(), scores.std()))
  plt.show()
```

5. OpenCV biblioteka

OpenCV (Open Source Computer Vision) je biblioteka programskih funkcija namijenjena obradama računalnog vida u stvarnom vremenu. Mnogi događaji u procesu računalnog vida moraju se izvršavati u stvarnom vremenu, što znači da obrada jedne slike smije trajati maksimalno 30-40 milisekundi, što je vrlo izazovan zahtjev. Glavni cilj OpenCV biblioteke je pružanje alata potrebnih za rješavanje problema računalnog vida.

OpenCV sadrži mješavinu funkcija niskog nivoa za obradu slika te složenih algoritama kao što su: algoritam za prepoznavanje lica, prepoznavanje pješaka na ulici, praćenje objekata te prepoznavanje uzoraka. Biblioteka je preuzeta više od sedam milijuna puta i trenutno sadrži više od 2500 optimiziranih algoritama. OpenCV podržava programske jezike: C, C++, Python, Javu i MATLAB te operacijske sustave: Windows, Linux, Android i Mac OS.

5.1 OpenCV i Python

U usporedbi sa jezicima C/C++, izvođenje Python programa je sporije. No, vrlo važno svojstvo jezika Python je njegovo povezivanje sa programskim jezikom C/C++. Omogućeno je pisanje kritičnih dijelova koda u programskom jeziku C/C++ koji se onda, uz pomoć posebnih omotača mogu koristiti u jeziku Python. OpenCV koristi upravo takav način integracije sa jezikom Python. OpenCV-Python biblioteku čini originalna implementacija algoritama OpenCV biblioteke u jeziku C++ omotanih Python omotačem.

6. Zaključak

Strojno učenje te računalni vid i njima bliske znanstvene discipline su vrlo popularna područja koja svakim danom postaju opsežnija. Da bi se omogućilo jednostavnije korištenje algoriama iz takvih područja, u programskom jeziku Python stvorene su biblioteke *SciKit-image* te *SciKit-learn*, koje se nadovezuju na druge postojeće biblioteke tog programskog jezika (*NumPy*, *SciPy*, *SciKit*) te korisniku omogućavaju da vrlo jednostavno koristi takve algoritme kao dio svog programa. Razlog velike popularnosti ovih biblioteka leži u njihovoj opsežnosti te efikasnsti implementacije. Obje biblioteke se oslanjaju na širok skup biblioteka programskog jezika Python, koje su većinom visoko efikasne. Jedan takav primjer je biblioteka NumPy, koju obje biblioteke koriste za spremanje podataka koje obrađuju. Zahvaljujući svojoj opsežnosti te jednostavnosti pisanja programa, programski jezik Python jedan je od idealnih izbora za pisanje programa koji se oslanjaju na podučja računalnog vida te strojnog učenja.

7. Literatura

https://hr.wikipedia.org/wiki/Ra%C4%8Dunalni_vid

http://www.zemris.fer.hr/education/ml/nastava/ag20022003/1-uvod.pdf

https://en.wikipedia.org/wiki/Machine_learning

http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf

https://web.math.pmf.unizg.hr/nastava/su/index.php/download_file/-/view/152/

http://web.mit.edu/dvp/Public/numpybook.pdf

http://fotografija.hr/unsharp-mask-izostravanje/

http://cazencott.info/dotclear/public/lectures/ma2823_2015/scikit-learn.pdf

https://en.wikipedia.org/wiki/Linear_regression

https://queue.acm.org/detail.cfm?id=2206309

8. Sažetak

U ovom seminarskom radu opisane su najpoznatije biblioteke za ostvarivanje strojnog učenja i računalnog vida korištene u programskom jeziku Python.

Da bi se algoritmi za računalnog vida i strojnog učenja mogli izvoditi, prije svega je potrebnan način za pohranu podataka koje će ti algoritmi obrađivati. U tu svrhu, korištena je biblioteka NumPy koja vrlo efikasno implementira metode za obradu višedimenzionalnih nizova. Najpoznatija biblioteka za obradu slika u programskom jeziku Python dolazi is SciKit skupa. Biblioteku SciKit-image čini kolekcija algoritama namijenjenih obradi slika. SciKit-image, u kombinaciji sa skupom biblioteka SciPy čini opsežan skup alata korištenih u području računalnog vida. Osim računalnog vida, u radu se opisuje i strojno učenje. Najpoznatija biblioteka koja sadrži metode korištene u strojnom učenju je SciKit-learn. Biblioteka se zasniva na NumPy, SciPy te Matplotlib bibliotekama. Konačno, kao alternativa SciKit-image biblioteci, u radu se spominje OpenCV, jedna od najpoznatijih biblioteka računalnog vida implementirana u jeziku C++. OpenCV-Python biblioteku čini originalna implementacija algoritama OpenCV biblioteke omotanih Python omotačem.