دانشکدگان فنی دانشگاه تهران
دانشکده مهندسی نقشه برداری و اطلاعات مکانی

پروژه درس تشخیص الگو

SVM

دانشجو:
علیرضا براهیمی

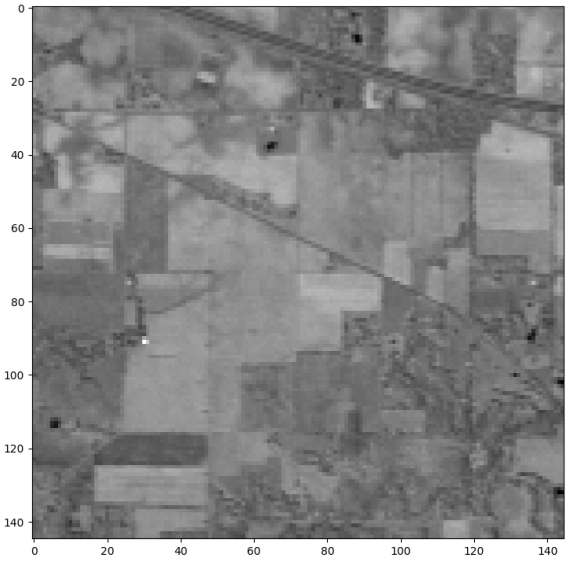
شماره دانشجویی:
810301017

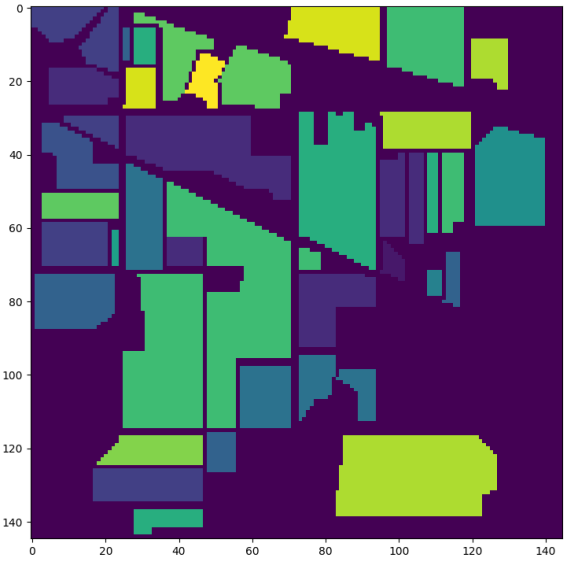
استاد:
دکتر حسنلو

نیمسال اول سال تحصیلی 1401 – 1402

مقدمه :

ماشین بردار پشتیبان (SVM) یک الگوریتم یادگیری ماشین نظارت شده محبوب است که برای طبقه بندی و تحلیل رگرسیون استفاده می شود. ایده اصلی پشت SVM یافتن یک ابر صفحه است که نقاط داده کلاس های مختلف را حداکثر جدا کند. hyperplane که حاشیه بین دو کلاس را به حداکثر می رساند به عنوان hyperplane حداکثر حاشیه شناخته می شود.
در مورد داده های قابل جداسازی خطی، SVM ابرصفحه بهینه را پیدا می کند که دو کلاس را از هم جدا می کند. با این حال، در بسیاری از سناریوهای دنیای واقعی، داده ها به صورت خطی قابل تفکیک نیستند و SVM از یک ترفند هسته برای ترسیم نقاط داده در فضایی با ابعاد بالاتر استفاده می کند، جایی که به صورت خطی قابل جداسازی می شوند. این به SVM اجازه می‌دهد تا ابرصفحه حداکثر حاشیه را که این دو کلاس را از هم جدا می‌کند، پیدا کند.
SVM یک الگوریتم قدرتمند است که به طور گسترده در زمینه های مختلف مانند بینایی کامپیوتر، پردازش زبان طبیعی و بیوانفورماتیک استفاده شده است. توانایی آن در مدیریت داده‌های با ابعاد بالا و استحکام آن در برابر نویز و موارد پرت، آن را به انتخابی محبوب برای بسیاری از مشکلات یادگیری ماشین تبدیل کرده است.
با این حال، SVM می تواند از نظر محاسباتی گران باشد، به خصوص برای مجموعه داده های بزرگ، و انتخاب تابع هسته مناسب و تنظیم پارامترها می تواند چالش برانگیز باشد.
به منظور کاهش بعد داده ، بر روی دیتا PCA اعمال شده و ۲۰ component اول آن به مدل داده می‌شود .





نتایج دقت برای پارامترهای متفاوت به صورت زیر می‌باشد :

params	accuracy
-h 0 -t 1 -d 1 -q	59.5147%
-h 0 -t 1 -d 2 -q	72.1218%
-h 0 -t 2 -q	78.2112%
-h 0 -r 0 -q	78.4015%
-h 0 -r 10 -q	80.0666%
-h 0 -r 20 -q	78.7345%
-h 0 -r 30 -q	78.4967%
-h 0 -r 40 -q	78.687%
-h 0 -r 50 -q	77.3549%
-h 0 -r 60 -q	79.9239%
-h 0 -r 70 -q	79.02%
-h 0 -r 80 -q	78.1637%
-h 0 -r 90 -q	78.7821%

```
import scipy.io
import matplotlib.pyplot as plt
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler
import rasterio as rio
from libsvm.svmutil import  *
import random

DATA = scipy.io.loadmat('/home/alireza/Desktop/seg/Indian_pines_corrected.mat')
label = scipy.io.loadmat('/home/alireza/Desktop/seg/Indian_pines_gt.mat')
DATA = DATA['indian_pines_corrected']
Lbl = label['indian_pines_gt']
lbl = Lbl.flatten()


def UINT8(Data) :
   shape = Data.shape
```

```
    for i in range(shape[2]):
        data = Data[i , : , :]
        data = data / data.max()
        data = 255 * data
        Data[i] = data.astype(np.uint8)
    return Data

def Uint8(img):
    img = img/img.max()
    img = img * 255
    img = img.astype(np.uint8)
    return img

def pca_process(data , n_comp=20):
    image = np.zeros((data.shape[0]*data.shape[1] , 1))
    for i in range(data.shape[2]):
        band = data[: , : , i]
        band = StandardScaler().fit_transform(band)
        band = band.flatten()
        band = band.reshape((-1 , 1))
        image = np.append(image , band , axis=1)
    image = image[: , 1:]
    pca = PCA(n_components=n_comp)
    pca_img = pca.fit_transform(image)
    return pca_img
data = pca_process(DATA)
plt.subplot(121)
plt.imshow(DATA[: , : , 100] , cmap = "gray")
plt.subplot(122)
plt.imshow(Lbl)
plt.show()
rate = 0.1
k = round(rate * data.shape[0])
test_ind = random.sample(range(0 , data.shape[0]) , k)
test = data[test_ind]
test_gt = lbl[test_ind]
train = np.delete(data , test_ind , axis = 0)
train_gt = np.delete(lbl , test_ind , axis = 0)
print(f'data:"total data : {data.shape[0]}/ test : {test.shape[0]} / train :
{train.shape[0]}')

T = [1 , 2 , 3]
kernel = ['polynomial' , 'radial' , 'sigmoid']
for t in T :
    if t == 1 :
        D = np.arange(1 , 3 , 1)
        for d in D:
            test_ind = random.sample(range(0, data.shape[0]), k)
            test = data[test_ind]
            test_gt = lbl[test_ind]
            train = np.delete(data, test_ind, axis=0)
            train_gt = np.delete(lbl, test_ind, axis=0)
```

```
            param = f'-h 0 -t {t} -d {d} -q'
            print('*' * 100)
            print(param)
            print(f'using {kernel[t-1]} kernel with degree of {d}')
            model = svm_train(train_gt, train, param)
            p_labels, p_acc, p_vals = svm_predict(test_gt, test, model)
    if t == 2 :
        test_ind = random.sample(range(0, data.shape[0]), k)
        test = data[test_ind]
        test_gt = lbl[test_ind]
        train = np.delete(data, test_ind, axis=0)
        train_gt = np.delete(lbl, test_ind, axis=0)
        print('*' * 100)
        param = f'-h 0 -t {t} -q'
        print(param)
        print(f'using {kernel[t - 1]} with default params')
        model = svm_train(train_gt, train, param)
        p_labels, p_acc, p_vals = svm_predict(test_gt, test, model)
    if t == 3 :
        Coef = np.arange(0 , 100 , 10)
        for coef in Coef :
            test_ind = random.sample(range(0, data.shape[0]), k)
            test = data[test_ind]
            test_gt = lbl[test_ind]
            train = np.delete(data, test_ind, axis=0)
            train_gt = np.delete(lbl, test_ind, axis=0)
            print('*' * 100)
            param = f'-h 0 -r {coef} -q'
            print(param)
            print(f'using {kernel[t - 1]} kernel with coef of {coef}')
            model = svm_train(train_gt, train, param)
            p_labels, p_acc, p_vals = svm_predict(test_gt, test, model)
```