به نام خدا

مهدی فقهی

۴۰۱۷۲۲۱۳۶

گزارش پروژه اول Pattern Recognition

مرحله اول (بارگذاری داده‌ها و بررسی داده‌ها)

ابتدا داده‌ها را از سایت yanna.lecun.com دانلود کردم و هر چهار فایل مربوط را در پوشه‌ای به نام samples در کنار کد اصلی قرار دادم و سپس به کمک library mnist داده‌ها را خواندم و داده‌ها را به دو دسته test , train به کمک خود کتابخانه و براساس همان نمونه‌های اصلی که برای train , test در نظر گرفته شده بود جدا کردم .

```
import matplotlib.pyplot as plt
import numpy as np
from mnist import MNIST
mndata = MNIST('samples')
images_train, labels_train = mndata.load_training()
images_train = np.array(images_train)
labels_train = np.array(labels_train)
images_test, labels_test = mndata.load_testing()
images_test = np.array(images_test)
labels_test = np.array(labels_test)
```

البته سپس در ادامه خواسته شد که parser این قسمت را خودمان بزنیم که بنده از کد در github به آدرس لینک استفاده کردم .

که در اینجا آنچه از کد فهمیدم را توضیح خواهم داد :

```python
import os
import struct


class Image:
    def __init__(self, dir='./'):
        self.train_files = {
                'images': os.path.join(dir, 'train-images-idx3-ubyte'),
                'labels': os.path.join(dir, 'train-labels-idx1-ubyte')
                }
        self.test_files = {
                'images': os.path.join(dir, 't10k-images-idx3-ubyte'),
                'labels': os.path.join(dir, 't10k-labels-idx1-ubyte')
                }

    @property
    def train(self):
        path = self.train_files
        return self._get_dataset(path)

    @property
    def test(self):
        path = self.test_files
        return self._get_dataset(path)

    def _get_dataset(self, path):
        images = self._load_images(path['images'])
        labels = self._load_labels(path['labels'])
        for image, label in zip(images, labels):
            yield image, label
```

یک کلاس تعریف کرده‌است به اسم Image که برای initial کردن باید آدرس محل جایی که داده‌های train و test در آن قرار گرفته است را تعریف کنیم .

سپس همانطور که مشاهده می‌کنید با صدا زدن تابع train یا test یک generator به ما برمی‌گردد که که به کمک آن می‌توانیم آیتم‌های درون فایل را یکی بخوانیم برگردانیم.

```
    def _load_images(self, fname):
        f = open(fname, 'rb')
        header = struct.unpack('>4i', f.read(16))
        magic, size, width, height = header

        if magic != 2051:
            raise RuntimeError("'%s' is not an MNIST image set." % fname)

        chunk = width * height
        for _ in range(size):
            img = struct.unpack('>%dB' % chunk, f.read(chunk))
            yield img, width, height

        f.close()

    def _load_labels(self, fname):
        f = open(fname, 'rb')
        header = struct.unpack('>2i', f.read(8))
        magic, size = header

        if magic != 2049:
            raise RuntimeError("'%s' is not an MNIST label set." % fname)

        for label in struct.unpack('>%dB' % size, f.read()):
            yield label

        f.close()
```

برای خواندن تصویرها تابع load images داریم که به کمک تابع open پایتون فایل مربوط را باز می‌کنیم سپس به کمک تابع unpack شانزده byte اول باینری فایل مربوط را خوانده و چهاربyte چهاربyte آن را جدا می‌کنیم و برابر با یک عدد در نظر می‌گیریم که به ما magic و تعدد عکس و عرض و طول عکس مربوط را می‌دهد .

اگر magic برابر ۲۰۵۱ نباشد خوب این فایل فایل MNIST و دیگر ادامه نمی‌دهم در غیر اینصورت تعداد byte های مربوط به هر عکس با ضرب طول و عرض بدست می‌آوریم و سپس به تعداد size که نشان دهنده تعداد هست یکی یکی عکس‌ها را تبدیل می‌کنیم یعنی به مقدار chunk می‌خوانیم و عکس‌های موجود را بازیابی می‌کنیم که به همراه عکس مقدار طول و عرض آن نیز بدست می‌آید .

در تابع label load هم همینطور است با این تفاوت که چون فقط یک عدد داریم به عنوان label پس فقط در ابتدا به اندازه دو کلمه می‌خوانیم که یکی برابر با تعداد سطرها و دیگری magic برای اینکه ببینیم آیا مربوط به MNIST هست یا نه را خواهیم داشت که سپس به کمک size تمامی label ها را پیدا خواهیم کرد .

```
def return_data_set(name):

        dataset = Image(name)
        img_train_list = []
        label_train_list = []
        for (img, width, height), label in dataset.train:

            img_train_list.append(img)
            label_train_list.append(label)


        img_test_list = []
        label_test_list = []
        for (img, width, height), label in dataset.test:


            img_test_list.append(img)
            label_test_list.append(label)

        return (img_train_list,label_train_list,img_test_list,label_test_list)
```

```
img_train_list,label_train_list,img_test_list,label_test_list = return_data_set('samples')

images_train = np.array(img_train_list)
labels_train = np.array(label_train_list)
images_test = np.array(img_test_list)
labels_test = np.array(label_test_list)
```

به کمک تابع بالا که توسط اینجانب زده شده است کل داده‌های train و test همراه با label شان برگردانده می‌شود .

در ادامه با کدهای زیر سعی در فهمیدن ابعاد داده‌های test ، train همچنین تعداد فیچرها کردم .

```
len(images_train)
```

```
60000
```

```
len(images_test)
```

```
10000
```

```
images_train[0].shape
```

```
(784,)
```

```
images_train.shape
```

```
(60000, 784)
```

```
labels_train.shape
```

```
(60000,)
```

در ادامه برای اینکه ببینم دقیقا با چه نوع داده‌های تصویری سرکار دارم و تعداد کلاس‌های من در این نوع داده چه تعداد است به کمک کد زیر برچسب تمامی کلاس‌ها را پیدا می‌کنم و فراوانی هر کدام را حساب می‌کنم .

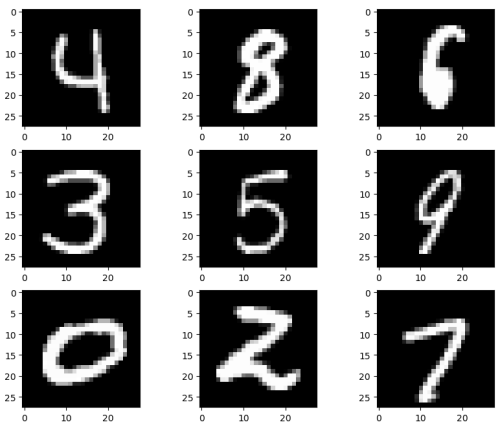
```
unique_elements, counts_elements = np.unique(labels_train, return_counts=True)
print("Frequency of unique values of the said array:")
print(np.asarray((unique_elements, counts_elements)))
```

```
Frequency of unique values of the said array:
[[   0    1    2    3    4    5    6    7    8    9]
 [5923 6742 5958 6131 5842 5421 5918 6265 5851 5949]]
```

سپس به صورت رندوم سعی کردم تمام داده‌های مربوط به یکی از 10 کلاس هستند را به صورت تصویری مشاهده نمایم .

```
from random import randrange

number_see = []
plt.figure(figsize=(10,8))
i = 0
while i < 10:
    index = randrange(0, len(images_train))  # choose an index ;-)
    if labels_train[index] in number_see:
        pass
    else :
        plt.subplot(330 + 1 + i)
        number_see.append(labels_train[index])
        one_date = images_train[index]
        image = one_date.reshape(28,28)
        i+= 1
        plt.imshow(image,cmap=plt.get_cmap('gray'))
plt.show()
```



مرحله دوم (پیاده‌سازی توابع کمکی برای انجام کارها)

در ادامه برای روی داده‌ها train انجام بدهم و به کمک SVM داده‌های مربوط را کلاس بندی کنم از کتابخانه sklearn.svm استفاده کردم .

برای انجام این کار از دوتا از مدل‌های این کتابخانه یعنی SVC , linearSVC استفاده کردم .

از آنجا که برای انجام SVM به صورت linear ، مدل linearSVC سریع‌تر بود از این کتابخانه استفاده کردم .

پروژه های علم داده نیاز به تکرار مکرر دارند. برای مثال، داده‌ها را برای مدل‌سازی با تبدیل به فرمت مناسب تمیز و آماده می‌کنیم، مدل را اجرا می‌کنیم، نتیجه می‌گیریم، مدل را بهبود می‌دهیم .

مدل را تغییر می‌دهیم و روی مهندسی ویژگی کار می‌کنیم، نتایج جدید دریافت می‌کنیم، آنها را با نتایج دیگر مقایسه می‌کنیم و غیره. انجام هر مرحله دوباره و دوباره آسان و هوشمندانه نیست. برای حل این مشکل، می‌توانیم از یک خط لوله برای ادغام مراحل گردش کار یادگیری ماشین استفاده کنیم.

pipeline برای تبدیل و آموزش سریع داده ها بسیار مفید هستند. علاوه بر این، می‌توانیم مدل‌های مختلف را با هم مقایسه کنیم و با ادغام جستجوی شبکه‌ای در pipeline خود، هایپر پارامترها را تنظیم کنیم.

به همین دلیل از کتابخانه sklearn.pipeline برای train کردن مدل خودم استفاده کردم که در آن مدلی که باید براساس آن آموزش می‌دادم را به تابع make_pipeline این کتابخانه می‌دادم براساس آنکه داده موردنظر براساس standard Scaler به شکل مناسبی داده‌ها scale شوند (بین صفر تا یک) و سپس براساس مدل مربوط به شکل pipeline آموزش ببینند .

```python
from sklearn.svm import SVC,LinearSVC
from sklearn.preprocessing import StandardScaler
from sklearn.pipeline import make_pipeline
def scv_learning_by_C_gamma_func(c,gamma,function,x,y):

    if function == 'linear':
        svm = make_pipeline(StandardScaler(),LinearSVC(C=c))
    else:
        svm = make_pipeline(StandardScaler(),SVC(kernel = function, probability=True, gamma=gamma, C=c))
    svm.fit(x,y)
    return svm
```

البته از آن‌جا که مدل بعضا وقتی Dual=true بود با warning اینکه مدل converge نکرده بود مواجه می‌شدم به شکل زیر function را پیاده سازی کردم .

```
((svm = make_pipeline(StandardScaler(),LinearSVC(C=c,dual=False
```

در ادامه سه تابع وجود دارد که براساس داده‌های داده شده بهترین gamma و C را پیدا می‌کنند .
در تابع اول تمامی ترکیبات ممکن را بررسی می‌نماید در تابع دوم متناسب با مقدار ثابت C بهترین gamma را پیدا می‌کند و در تابع سوم نیز بر اساس مقدار ثابت gamma بهترین C ممکن را پیدا می‌کند .
که نمونه جامع که تابع اول است را به عنوان نمونه تابع اول را نشان می‌دهیم .

```python
def find_beest_C_gamma(x_train,y_train,function,x_test,y_test):

    data = []

    best_accuracy = 0
    best_c_gamma = (0,0)
    c_exm = [0.001,0.01,1,10,100]
    gamma = [0.1,1,10]

    for c in c_exm:
        for g in gamma:
            model = scv_learning_by_C_gamma_func(c,g,function, x_train, y_train)
            train_classifier = model.predict(x_train)
            train_accuracy = metrics.accuracy_score(train_classifier,y_train)

            test_classifier = model.predict(x_test)
            test_accuracy = metrics.accuracy_score(test_classifier,y_test)
            data.append({"id":f"C = {c},Gamma = {g}", "training":train_accuracy,"test":test_accuracy})
            print(f"C = {c},Gamma = {g}", f"training {train_accuracy} test: {test_accuracy}")
            if best_accuracy < (train_accuracy+test_accuracy)/2:
                    best_accuracy = (train_accuracy+test_accuracy)/2
                    best_c_gamma = (c,g)


    return [best_c_gamma,best_accuracy,data]
```

همانطور که مشاهده می‌کنید بعد از ساخت هر model براساس داده‌های train، داده‌های train و test را به کمک model بدست آمده train می‌کنم و دقت بدست آمده را یک لیست به صورت یک دیکشنری ذخیره می‌کنم که نشان می‌دهد با توجه به c , gamma داده شده دقت مدل بر روی داده train , test برابر چه مقداری است و بهترین مدل مدلی است از نظر من که میانگین دقت آن بر روی داده‌های train , test کمترین مقدار باشد ولی با این وجود برای بوجود آوردن جدولی که تفاوت دقت هر مدل براساس C , gamma موجود نشان دهد داده data که همان لیست از دیکشنری‌ها نیز هست به عنوان خروجی برمی‌گردانم علاوه بر بهترین C, gamma و بهترین accuracy براساس این C,gamma .

سپس به کمک تابع make accuracy function یک جدول از accuracy training و accuracy test براساس gamma , C مختلف .

```
def make_accuracy_table(dic_data):

    data = [[item['training'],item['test']] for item in dic_data]
    print(data)
    columns = ('training', 'test')
    rows = [item['id'] for item in dic_data]

    plt.rcParams["figure.figsize"] = [7.00, 3.50]
    plt.rcParams["figure.autolayout"] = True
    fig, axs = plt.subplots(1, 1)

    axs.axis('tight')
    axs.axis('off')
    the_table = axs.table(cellText=data, colLabels=columns, rowLabels=rows,loc='center')
    plt.show()
```

یکی از توابع پیاده سازی شده توسط این‌جانب که در این پروژه بسیار کمک کننده بود،
تابع get details of model بود که به کمک آن معیارهایی که در ادامه توضیح میدهم را بررسی می کردم

```
from sklearn import metrics
from yellowbrick.classifier import ROCAUC

def get_details_of_model(c,gamma,function,x_train,y_train,x_test,y_test,name,model=None):
    if model == None:
        svm = scv_learning_by_C_gamma_func(c,gamma,function,x_train,y_train)
    else :
        svm = model
    train_classifier = svm.predict(x_train)
    train_accuracy = metrics.accuracy_score(train_classifier,y_train)

    test_classifier = svm.predict(x_test)
    test_accuracy = metrics.accuracy_score(test_classifier,y_test)

    predicted = svm.predict(x_test)
    print(f"train accuracy {name} = {train_accuracy}")
    print(f"test accuracy {name} = {test_accuracy}")
    print(
    f"Classification report for classifier {svm}:\n"
    f"{metrics.classification_report(y_test, predicted)}\n")
    disp = metrics.ConfusionMatrixDisplay.from_predictions(y_test, predicted)
    disp.figure_.suptitle(f"Confusion Matrix {name}")
    print(f"Confusion matrix {name}:\n{disp.confusion_matrix}")
    plt.show()
    visualizer = ROCAUC(model, encoder={0: 'zero',
                                        1: 'one',
                                        2: 'two',
                                        3: 'three',
                                        4: 'four',
                                        5: 'five',
                                        6: 'six',
                                        7: 'seven',
                                        8: 'eight',
                                        9: 'nine',
                                        10: 'ten'})
    visualizer.fit(x_train, y_train)
    visualizer.score(x_test, y_test)
    visualizer.show()
```

این تابع ورودی C, gamma,function ,training data,training label,test data ,test label,name,model را به عنوان ورودی می‌گیرد و از آیتم‌های function, C ,gamma برای ساخت مدل استفاده می‌کند که در صورتی که model برابر با none باشد در صورتی که بعنوان یک ورودی مدل را به تابع بدهیم دیگر مدل را نمی‌سازد.سپس به کمک مدل دقت را براساس مدل train , test بدست می‌آورد سپس براساس داده‌های تست یک confusion matrix رسم می‌کند و precision ,recall ,f1-score ,support را برای هر label حساب می‌کند و در ادامه یک نمودار برای ROC , میزان AUC را حساب می‌کنم این کار به کمک کتاب‌خانه `yellowbrick.classifier` انجام می‌شود به کمک پارامتر ROCAUC مدل را به همراه نام کلاس و labeli که هر کدام دارند به تابع می‌دهیم و این محاسبات را برای ما انجام می‌دهد .

مرحله سوم : (قسمت‌های خواسته شده پروژه)

همانطور که مشاهده می‌کنید با یکسان در نظر گرفتن C , function و با توجه به اینکه تمامی توابع ما linear بود نیازی به gamma نداشتیم نتیجه که حاصل شد به این گونه بود .

برای انجام اینکار ابتدا داده را به صورت PCA 100 , PCA 30 دسته‌بندی می‌کنم .

```
from sklearn.decomposition import PCA
```

```
pca_30 = PCA(n_components=30)
all_data = np.concatenate((images_train,images_test))
all_data_pac_30 = pca_30.fit_transform(all_data)

pac_30_feature_training_x = all_data_pac_30[:images_train.shape[0]]
pac_30_feature_test_x = all_data_pac_30[images_train.shape[0]:]
print(pac_30_feature_training_x.shape,pac_30_feature_test_x.shape)
```

```
(60000, 30) (10000, 30)
```

```
pac_100_new = PCA(n_components=100)
all_data = np.concatenate((images_train,images_test))
all_data_pac_100 = pac_100_new.fit_transform(all_data)
pac_100_new_training = all_data_pac_100[:images_train.shape[0]]
pac_100_test_test = all_data_pac_100[images_train.shape[0]:]
print(pac_100_new_training.shape,pac_100_test_test.shape)
```

```
(60000, 100) (10000, 100)
```

در مرحله بعد براساس تابع خطی مقدار با کلیه ویژگی‌ها ویژگی‌های مدل را بررسی میکنم به کمک تابع یادشده (get details of model).

وحاصل به شکل زیر خواهد بود برای Accuracy :

Without PCA :

train accuracy linear without pca with c 1 = 0.9205166666666666

test accuracy linear without pca with c 1 = 0.9122

PCA 100 :

train accuracy linear with pca 100 and c 1 OVR = 0.9093333333333333

test accuracy linear with pca 100 and c 1 OVR = 0.9129

PCA 30 :

train accuracy linear with pca with c 1 = 0.8766166666666667

test accuracy linear with pca with c 1 = 0.8837

نتیجه :

دقت مدل با ۱۰۰ عدد feature به مدل کامل نزدیک بود ولی با کاهش feature به ۳۰ عدد مدل کاهش پیدا می کرد

برای بررسی کردن f1-score , Precision , Recall :

Without PCA :

	precision	recall	f1-score	support
0	0.95	0.98	0.96	980
1	0.95	0.98	0.97	1135
2	0.92	0.88	0.90	1032
3	0.90	0.90	0.90	1010
4	0.91	0.93	0.92	982
5	0.87	0.86	0.87	892
6	0.93	0.95	0.94	958
7	0.91	0.91	0.91	1028
8	0.87	0.85	0.86	974
9	0.91	0.86	0.89	1009
accuracy			0.91	10000
macro avg	0.91	0.91	0.91	10000
weighted avg	0.91	0.91	0.91	10000

PCA 100 :

	precision	recall	f1-score	support
0	0.94	0.98	0.96	980
1	0.96	0.98	0.97	1135
2	0.93	0.89	0.91	1032
3	0.90	0.90	0.90	1010
4	0.90	0.92	0.91	982
5	0.89	0.85	0.87	892
6	0.92	0.96	0.94	958
7	0.92	0.92	0.92	1028
8	0.87	0.85	0.86	974
9	0.88	0.87	0.88	1009
accuracy			0.91	10000
macro avg	0.91	0.91	0.91	10000
weighted avg	0.91	0.91	0.91	10000

PCA 30 :

	precision	recall	f1-score	support
0	0.91	0.98	0.95	980
1	0.95	0.98	0.96	1135
2	0.89	0.84	0.87	1032
3	0.86	0.87	0.86	1010
4	0.86	0.90	0.88	982
5	0.83	0.77	0.80	892
6	0.91	0.93	0.92	958
7	0.90	0.90	0.90	1028
8	0.85	0.83	0.84	974
9	0.85	0.81	0.83	1009
accuracy			0.88	10000
macro avg	0.88	0.88	0.88	10000
weighted avg	0.88	0.88	0.88	10000

Precision , Recall :

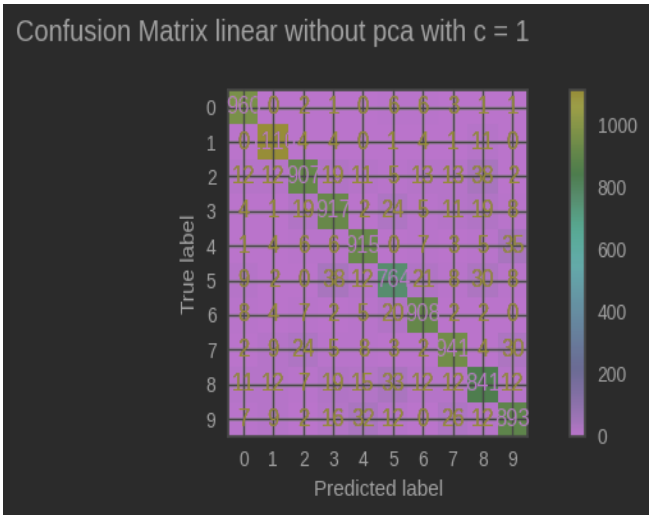
مدل با ۱۰۰ عدد feature به مدل کامل نزدیک بود ولی با کاهش feature به ۳۰ عدد مدل برای هر کدام یک از کلاس‌ها کاهش پیدا می کرد .

f1-score :

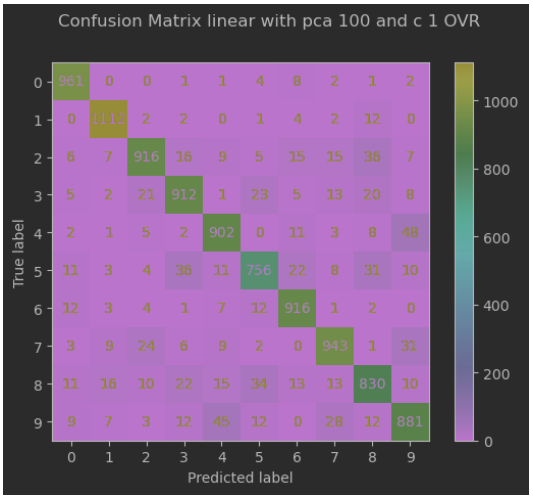
با اینکه هر سه بهم نزدیک هستند ولی با این وجود pca 30 از بقیه در تمامی موارد ۲ الی ۴ واحد کمتر است.

برای بررسی کردن Confusion Matrix :

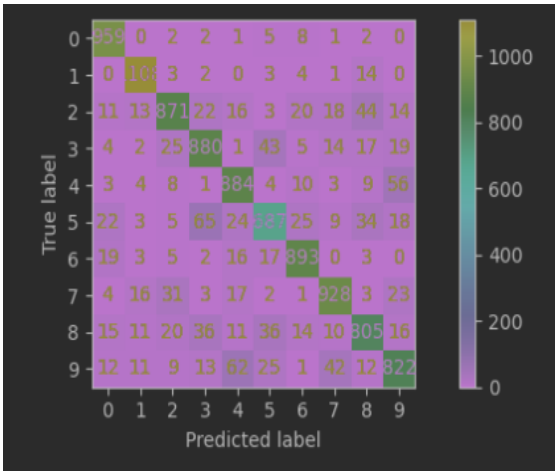
Without PCA :



PCA 100 :



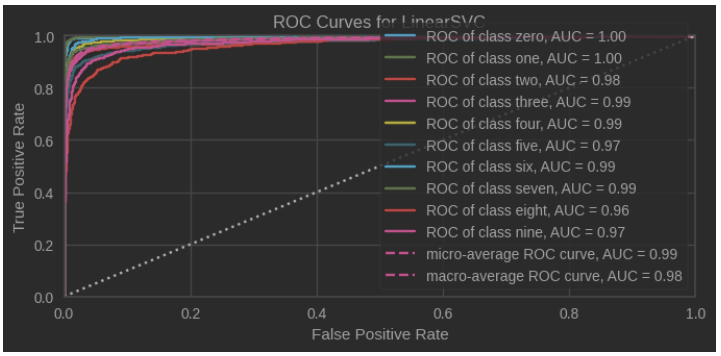
PCA 30 :



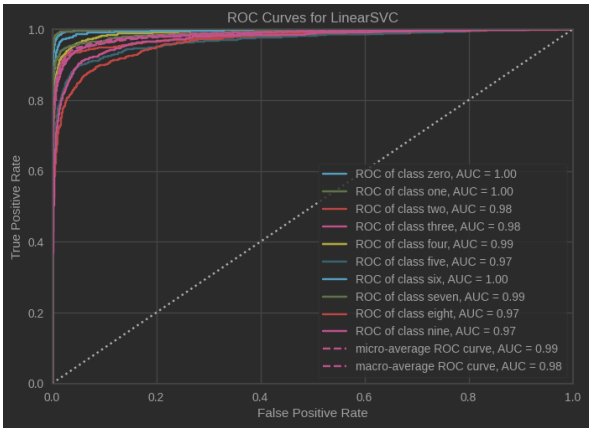
همانطور که می‌بینید با مشاهد confusion matrix شاهد هستیم که در اکثر موارد مدل کامل از هر دو مدل دیگه بهتر پیش‌بینی کرده البته pca 100 به مدل کامل خیلی نزدیک است با این وجود مدل pca 30 باز نیز در این شکل نسبت به دو مدل دیگر در جایگاهی پایین‌تر قرار می‌گیرد .

:AUC and RUCE

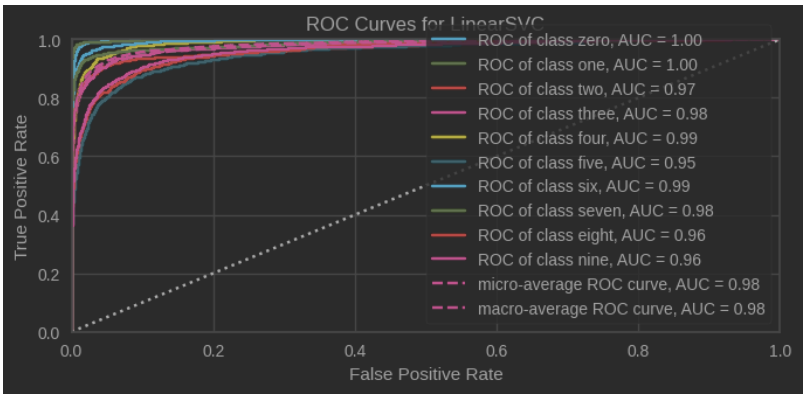
Without PCA :



PCA 100 :



PCA 30 :



با دیدن نسبت (True Positive /False positive) که همان نمودار RUC به ازای تمامی کلاس‌ها صفر تا ۹ است و بدست آوردن مساحت زیر نمودار که حاصل AUC را برای ما بدست می‌آورد و اگر از اختلاف جزئی هر نمودار صرف نظر کنیم می‌بینیم که هر سه روش با AUC بالای ۹۵ درصد به ازای تمامی کلاس‌ها در حالت مناسبی قرار دارند .

قسمت دوم پروژه :

قسمت الف)

از ما خواسته شده بود که به کمک کرنل خطی براساس C های متفاوت جدول مربوط را پر کنیم .

داده‌ها با 30 pca هستند .

	training	test
C = 0.001,Gamma = 1	0.8637166666666667	0.8735
C = 0.01,Gamma = 1	0.87405	0.8823
C = 1,Gamma = 1	0.8764166666666666	0.884
C = 10,Gamma = 1	0.8764666666666666	0.884
C = 100,Gamma = 1	0.8764833333333333	0.884

همانطور که مشاهده می‌کنیم ابتدا با افزایش C مقدار دقت افزایش پیدا می‌کند ولی از یک جایی به بعد دقت ثابت می‌ماند و افزایش پیدا نمی‌کند .

قسمت ب)

در حالت بعد خواسته شده بود که با تغییر c =1 , gamma برای یک مدل با kernel rbf جدول مدل را پر کنیم که حاصل برابر شد با :

	training	test
C = 1,Gamma = 0.1	0.9983	0.9831
C = 1,Gamma = 1	1.0	0.314
C = 1,Gamma = 10	1.0	0.1135

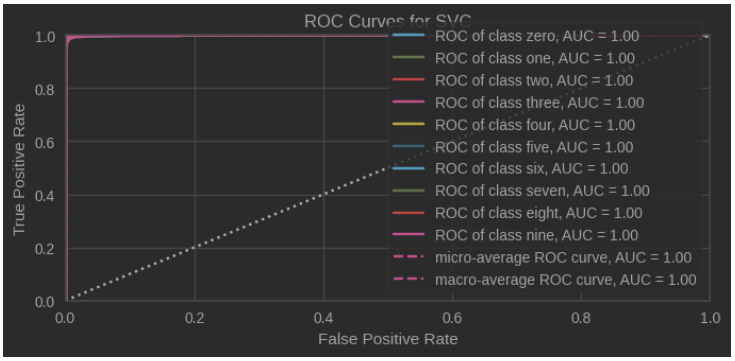
نتیجه گیری :

با افزایش gamma مدل به شدت کاری خود را از دست داده و دقت آن کاهش پیدا کرده است پس باید برای پیدا کردن بهترین hyperparameter ها حتما مقادیر مختلف را چک کنیم .

با متود get details مشخصات PCA 30 با gamma = 0.1 , C = 1 صدا می‌زنیم .

	precision	recall	f1-score	support
0	0.98	0.99	0.99	980
1	0.99	0.99	0.99	1135
2	0.97	0.98	0.98	1032
3	0.98	0.98	0.98	1010
4	0.99	0.98	0.98	982
5	0.98	0.99	0.98	892
6	1.00	0.99	0.99	958
7	0.99	0.98	0.98	1028
8	0.97	0.98	0.98	974
9	0.98	0.96	0.97	1009
accuracy			0.98	10000
macro avg	0.98	0.98	0.98	10000
weighted avg	0.98	0.98	0.98	10000

```
[[ 975    0    1    0    0    0    0    1    3    0]
 [   0 1128    3    1    0    0    0    0    2    1]
 [   4    2 1013    3    0    0    1    3    6    0]
 [   1    0    2  993    0    6    0    5    3    0]
 [   1    0    4    0  964    0    1    0    2   10]
 [   2    0    0    6    1  880    1    0    2    0]
 [   5    1    2    0    2    2  945    0    1    0]
 [   0    5    8    0    0    1    0 1008    1    5]
 [   3    0    2    9    1    2    0    2  954    1]
 [   3    4    4    5    9    4    0    2    7  971]]
```



تا کنون PCA 30 با C = 1 و gamma = 0.1 با kernel = rbf بهترین نتیجه را برای ما در دقت و بقیه موارد داشته است .

قسمت سوم پروژه :

در این قسمت از ما خواسته شده بود که با در نظر گرفتن PCA 100 و با LinearSVC و با C ثابت حالت One vs All و LinearSVC با متد crammer signer را با یکدیگر مقایسه کنید .

Accuracy One vs All :

train accuracy linear without pca with c = 1 = 0.9275166666666667

test accuracy linear without pca with c = 1 = 0.9156

Accuracy with Crammer :

train accuracy linear with pac 100 and c 1 with crammer = 0.9216333333333333

test accuracy linear with pac 100 and c 1 with crammer = 0.9239

برای بررسی کردن Recall , Precision , f1-score :

One vs All :

	precision	recall	f1-score	support
0	0.94	0.98	0.96	980
1	0.96	0.98	0.97	1135
2	0.93	0.89	0.91	1032
3	0.90	0.90	0.90	1010
4	0.91	0.92	0.91	982
5	0.89	0.84	0.87	892
6	0.92	0.96	0.94	958
7	0.92	0.92	0.92	1028
8	0.87	0.85	0.86	974
9	0.89	0.88	0.88	1009
accuracy			0.91	10000
macro avg	0.91	0.91	0.91	10000
weighted avg	0.91	0.91	0.91	10000

Crammer:

	precision	recall	f1-score	support
0	0.94	0.98	0.96	980
1	0.96	0.98	0.97	1135
2	0.92	0.90	0.91	1032
3	0.91	0.91	0.91	1010
4	0.92	0.92	0.92	982
5	0.90	0.86	0.88	892
6	0.93	0.95	0.94	958
7	0.94	0.93	0.93	1028
8	0.88	0.89	0.89	974
9	0.91	0.90	0.91	1009
accuracy			0.92	10000
macro avg	0.92	0.92	0.92	10000
weighted avg	0.92	0.92	0.92	10000

بررسی کردن Confusion Matrix:

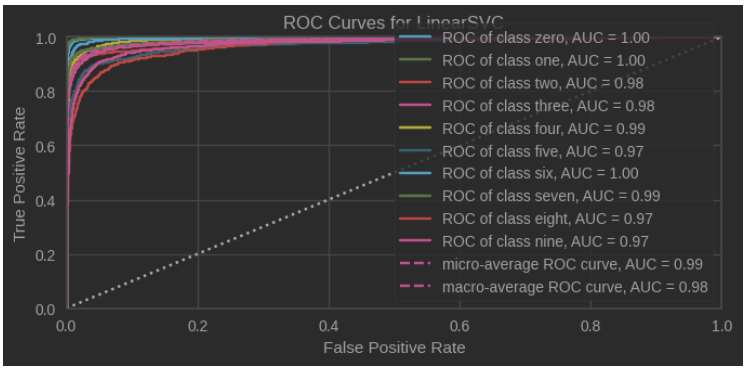
One vs All :

```
Confusion matrix linear with pca 100 and c 1 OVR:
[[ 961    0    0    1    1    3    9    2    1    2]
 [   0 1111    1    3    0    2    4    2   12    0]
 [   6    7  918   15    9    5   15   15   37    5]
 [   5    2   21  913    1   22    4   13   19   10]
 [   2    2    5    1  904    0   10    3    8   47]
 [  12    3    3   39   12  753   23    6   31   10]
 [  13    3    4    1    7   12  915    1    2    0]
 [   3    8   24    6    7    2    0  944    1   33]
 [  12   15    9   23   15   36   13   14  829    8]
 [  10    7    3   11   38   11    0   31   10  888]]
```

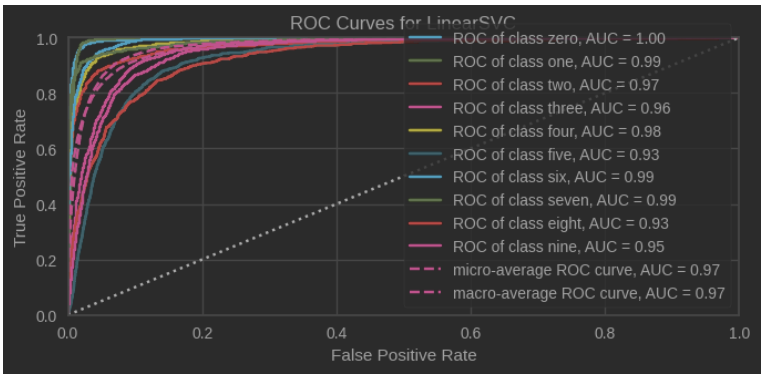
Crammer :

بررسی کردن AUC , ROC :

One Vs All :



Crammer :



خوب در اکثر موارد Crammer بهتر از One Vs All بود هرچند در قسمت AUC,ROC نسبت به One Vs All افت کیفیت داشت ولی آنقدر زیاد نبود و به خوبی می‌توان دید درنهایت به بهترین حالت خودش توانایی رسیدن دارد.

توضیح مقاله :

فرض کنید :

$S = \{(\bar{x}1, y1), \ldots, (\bar{x}m, ym)\}$

مجموعه ای از نمونه های آموزشی m باشد. ما فرض میکنیم که هر کدام مثال $\bar{x}i$ از یک دامنه $X \subseteq n$ کشیده شده است و هر برچسب yi یک عدد صحیح از مجموعه

$Y = \{1، \ldots ، k\}$

یک طبقه بندی کننده (چند کلاسه) یک تابع $H : X \to Y$ است که یک نمونه را ترسیم می کند و هرعنصر $\bar{x}$ به عنصر y از Y انتصاب داده می‌شود .

که در این جا یک ماتریس M داریم که در جزوه استاد W نشان داده شده که از ۱ تا k سطر دارد که به تعداد کلاس‌های است . وقتی می‌خواهیم ببینیم که یک نمونه متعلق به یک کلاس است یا نه در این ماتریس ضرب می کنیم و یک بردار در یافت می کنیم که بزرگترین مقدار به ما می گوید که نمونه ما در کدام کلاس قرار می گیرید و بزرگی اینجا نشان دهنده بزرگی شباهت است .

پس براساس این تعریف خطای ما برابر خواهد بود با :

$$\epsilon_S(M) = \frac{1}{m} \sum_{i=1}^{m} [\![H_{\mathbf{M}}(x_i) \neq y_i]\!]$$

هدف ما این است که یک M پیدا کنیم که empirical error اون کمترین میزان ممکن باشد .

در ادامه مقاله برای misclassification error شکل جدیدتری به شکل زیر ترسیم می‌نماید.

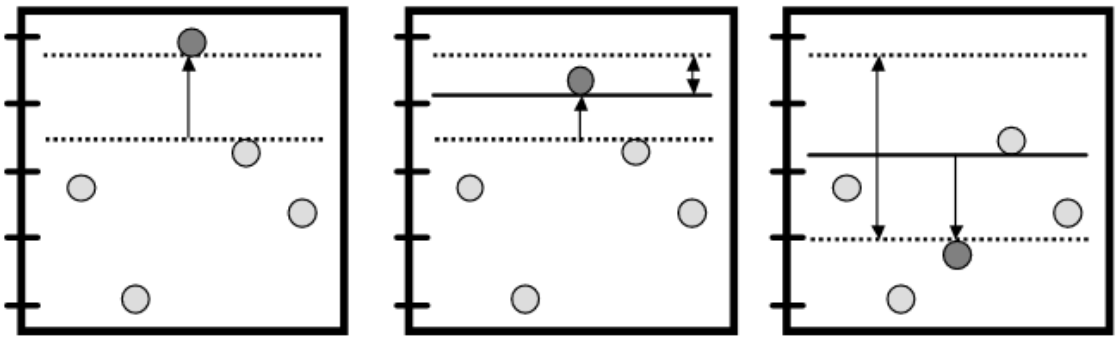
$$\max_r \{\bar{M}_r \cdot \bar{x} + 1 - \delta_{y,r}\} - \bar{M}_y \cdot \bar{x} \ ,$$

که در واقع فهم این فرمول به شکل زیر که استاد تعریف کرده بسیار راحت‌تر است .

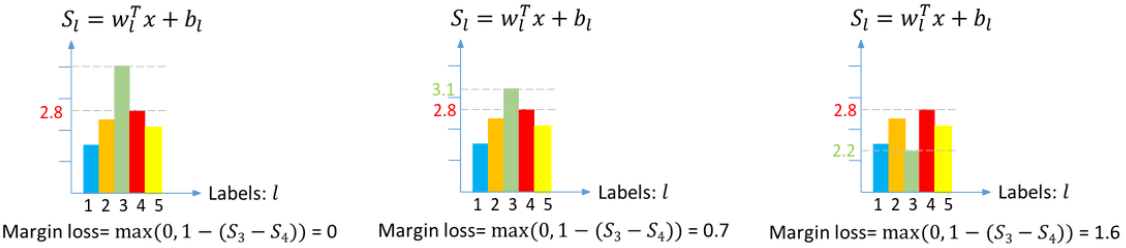
$\text{Margin loss} = \max(0,1 - (Si - Sl))$

در اینجا Si میزان بزرگی label درست براساس Mi است و Sl نیز بزرگی بزرگترین کلاس به غیر از Si است که اگر اختلاف شان بزرگتر از یک باشد در نتیجه margin loss برابر با صفر می‌شود .

در فرمول بالا تر نیز دقیقا به همین شکل می‌خواهیم طبقه بندی ما بزرگترین margin ممکن را داشته باشد .
که شکل زیر تعریف دقیق تری از این موضوع است .



که نمود بصری آن به شکل پایین بسیار شبیه است .



که برای مجموعه train خواهیم داشت :

ما می گوییم که یک نمونه S به صورت خطی توسط یک ماشین چند کلاسه قابل تفکیک است اگر ماتریس M وجود داشته باشد به طوری که loss فوق برای همه مثال های S برابر با صفر باشد.

$$\forall i \quad \max_r \{\bar{M}_r \cdot \bar{x}_i + 1 - \delta_{y_i,r}\} - \bar{M}_{y_i} \cdot \bar{x}_i = 0 \ .$$

همچنین باید محدودیت زیر را نیز در نظر بگیرد .

$$\forall i, r \quad \bar{M}_{y_i} \cdot \bar{x}_i + \delta_{y_i,r} - \bar{M}_r \cdot \bar{x}_i \geq 1 \quad . \qquad (4$$

همین فرمول بالا به یک شکل دیگر :

$$S_i - S_l \geq 1 \equiv (1 - (S_i - S_l)) \leq 0$$

براساس همین اگر هر چقدر norm ماتریس M کوچکتر باشد در نتیجه شرط بالا احتمال موفقیتش بیشتر است
پس :

$$\begin{aligned} \min_{M} \quad & \frac{1}{2}\|M\|_2^2 \\ \text{subject to :} \quad & \forall i, r \quad \bar{M}_{y_i} \cdot \bar{x}_i + \delta_{y_i,r} - \bar{M}_r \cdot \bar{x}_i \geq 1 \end{aligned}$$

فرمول بالا به شکل تعریف شده در جزوه استاد :

$$\min_{(w_1;\, b_1),\ldots,\,(w_k;\, b_k)} \frac{1}{2}\sum_{l=1}^{k}\|\boldsymbol{w}\|_2^2$$

regularizer

Subject to: $S_i - S_l \geq 1,$
$l \in \{1, \ldots, k\}, l \neq i$
$\forall i \in [m],$

Score constraint: Score for true label is higher than score for ***any*** other label by 1

البته مدل بالا یک مدل غیرواقعی هست در حالت کلی، نمونه S ممکن است قابل جداسازی خطی توسط یک ماشین چند کلاسه نباشد. پس ما مدل را با خطا در نظر می گیریم در نتیجه خواهیم داشت :

$$\begin{aligned} \min_{M,\xi} \quad & \frac{1}{2}\beta\|M\|_2^2 + \sum_{i=1}^{m}\xi_i \\ \text{subject to :} \quad & \forall i, r \quad \bar{M}_{y_i} \cdot \bar{x}_i + \delta_{y_i,r} - \bar{M}_r \cdot \bar{x}_i \geq 1 - \xi_i \end{aligned}$$

که به شکل زیر در جزوه استاد تعریف شده است :

$$\min_{(w_1;\, b_1),\ldots,\,(w_k;\, b_k)} \frac{1}{2}\sum_{l=1}^{k}\|w_l\|^2 + C\sum_{i=1}^{m}\xi_i^p$$

Subject to: $S_i - S_l \geq 1 - \xi_i \wedge \xi_i \geq 0$
$l \in \{1, \ldots, k\}, l \neq i$
$\forall i \in [m]$

در اینجا B در واقع همان (۱/c) است .

پس dual مسئله بالا به شکل زیر خواهد بود :

$$\begin{aligned} \mathcal{L}(M, \xi, \eta) \quad = \quad & \frac{1}{2}\beta\sum_{r}\|\bar{M}_r\|_2^2 + \sum_{i=1}^{m}\xi_i \\ & + \sum_{i,r}\eta_{i,r}\left[\bar{M}_r \cdot \bar{x}_i - \bar{M}_{y_i} \cdot \bar{x}_i - \delta_{y_i,r} + 1 - \xi_i\right] \\ \text{subject to :} \quad & \forall i, r \quad \eta_{i,r} \geq 0 \quad . \end{aligned}$$

نسبت به متغیرها مشتق می گیریم و برابر با صفر قرار می‌دهیم در نتیجه خواهیم داشت :

$$\frac{\partial}{\partial \xi_i}\mathcal{L} = 1 - \sum_r \eta_{i,r} = 0 \quad \Rightarrow \quad \sum_r \eta_{i,r} = 1 \ .$$

$$\begin{aligned}\frac{\partial}{\partial \bar{M}_r}\mathcal{L} &= \sum_i \eta_{i,r}\bar{x}_i - \sum_{i,y_i=r} \underbrace{\left(\sum_q \eta_{i,q}\right)}_{=1} \bar{x}_i + \beta \bar{M}_r \\ &= \sum_i \eta_{i,r}\bar{x}_i - \sum_i \delta_{y_i r}\bar{x}_i + \beta \bar{M}_r = 0 \ ,\end{aligned}$$

که در نتیجه :

$$\bar{M}_r = \beta^{-1}\left[\sum_{i:y_i=r}(1-\eta_{i,r})\bar{x}_i + \sum_{i:y_i\neq r}(-\eta_{i,r})\bar{x}_i\right]$$

خواهیم داشت :

$$\mathcal{Q}(\eta) \ = \ -\frac{1}{2}\beta^{-1}\sum_{i,j}(\bar{x}_i\cdot\bar{x}_j)\left[\sum_r(\delta_{y_i,r}-\eta_{i,r})(\delta_{y_j,r}-\eta_{j,r})\right] - \sum_{i,r}\eta_{i,r}\delta_{y_i,r} \ .$$

در نظر بگیریم که 1ī یک بردار است که همه درایه آن صفر است به غیر از درایه iام آن .

و 1 را یک بردار در نظر بگیرید که تمامی درآیه آن یک است .براساس این دو می‌توانیم dual را به شکل زیر بنویسیم .

$$\begin{aligned}\max_{\eta} \quad & \mathcal{Q}(\eta) = -\frac{1}{2}\beta^{-1}\sum_{i,j}(\bar{x}_i\cdot\bar{x}_j)\left[(\bar{1}_{y_i}-\bar{\eta}_i)\cdot(\bar{1}_{y_j}-\bar{\eta}_j)\right] - \sum_i \bar{\eta}_i\cdot\bar{1}_{y_i} \\ \text{subject to :} \quad & \forall i \ : \ \bar{\eta}_i \geq 0 \quad \text{and} \quad \bar{\eta}_i\cdot\bar{1} = 1 \ .\end{aligned}$$

بر همین اساس τi =¯ 1yi −¯ ηi در نظر میگیریم در نتیجه تمامی معادلات را به شکل زیر بازنویسی میکنیم .

$$\bar{M}_r = \beta^{-1}\sum_i \tau_{i,r}\bar{x}_i \ .$$

$$\begin{aligned}\max_{\tau} \quad & \mathcal{Q}(\tau) = -\frac{1}{2}\sum_{i,j}(\bar{x}_i\cdot\bar{x}_j)(\bar{\tau}_i\cdot\bar{\tau}_j) + \beta\sum_i \bar{\tau}_i\cdot\bar{1}_{y_i} \\ \text{subject to :} \quad & \forall i \quad \bar{\tau}_i \leq \bar{1}_{y_i} \quad \text{and} \quad \bar{\tau}_i\cdot\bar{1} = 0 \ .\end{aligned}$$

که در جزوه استاد به شکل زیر نوشته شده است :

$$\max_{A} \mathcal{L} = C\sum_{i=1}^{m} \bar{A}_i \cdot \bar{1}_{y_i} - \frac{1}{2}\sum_{i,j=1}^{m} (\bar{A}_i \cdot \bar{A}_j)(x_i \cdot x_j)$$

$$\text{Subject to: } \forall i : \bar{A}_i \le \bar{1}_{y_i} \text{ and } \bar{A}_i \cdot \bar{1} = 0$$

- Where $\bar{A}_i = \bar{1}_{y_i} - \bar{\alpha}_i$

همچنین اگر فانکشن kernel داشته باشیم به شکل زیر در خواهد آمد .

$$\max_{\tau} \quad Q(\tau) = -\frac{1}{2}\sum_{i,j} K(\bar{x}_i, \bar{x}_j)\ (\bar{\tau}_i \cdot \bar{\tau}_j) + \beta \sum_{i} \bar{\tau}_i \cdot \bar{1}_{y_i}$$

$$\text{subject to :} \quad \forall i \quad \bar{\tau}_i \le \bar{1}_{y_i} \quad \text{and} \quad \bar{\tau}_i \cdot \bar{1} = 0 \ ,$$

$$\max_{A} \mathcal{L} = C\sum_{i=1}^{m} \bar{A}_i \cdot \bar{1}_{y_i} - \frac{1}{2}\sum_{i,j=1}^{m} (\bar{A}_i \cdot \bar{A}_j) K(x_i \cdot x_j)$$

$$\text{Subject to: } \forall i : \bar{A}_i \le \bar{1}_{y_i} \text{ and } \bar{A}_i \cdot \bar{1} = 0$$

پس hypotheses ما به شکل زیر خواهد شد :

$$H(\bar{x}) = \arg\max_{r=1}^{k} \left\{ \sum_{i} \tau_{i,r} K(\bar{x}, \bar{x}_i) \right\} .$$

$$h(x) = \arg max_{l=1}^{k} \left\{ \sum_{i=1}^{m} A_{i,l} K(x, x_j) + b_l \right\}$$

از اینجا به بعد درباره حل مسئله بهینه سازی صحبت می کنیم :

مسئله درجه دوم دوگانه ارائه شده توسط معادله را می توان با استفاده از تکنیک های برنامه نویسی (QP) استاندارد حل کرد . با این حال، از آنجایی که تعداد متغیرهای mk حالت است، تبدیل آن معادله دوگانه ارائه شده از یک ماتریس با اندازه mk × mk استفاده می کند که واضح است که ذخیره یک ماتریس با آن اندازه بزرگ سخت و مسئله به این شکل غیرقابل حل است.

تجزیه آن به مسائل کوچک یک الگوریتم ساده و کارآمد حافظه برای حل مسئله بهینه سازی درجه دوم ارائه شده است .

ایده اصلی الگوریتم ما بر اساس جداسازی قیود معادله است آخربه m مجموعه های منفصل به شکل زیر است :

$$\{\bar{\tau}_i | \bar{\tau}_i \leq \bar{1}_{y_i},\ \bar{\tau}_i \cdot \bar{1} = 0\}_{i=1}^{m}$$

و در یک حلقه الگوریتم به شکل زیر کار می کند :

در هر دور الگوریتم یک الگوی p را انتخاب می کند و مقدار هدف را بهبود می بخشد براساس محدودیت‌هایی که در مسئله داریم .

Input $\{(\bar{x}_1, y_1), \ldots, (\bar{x}_m, y_m)\}$.
Initialize $\bar{\tau}_1 = \bar{0}, \ldots, \bar{\tau}_m = \bar{0}$.
Loop:

1. Choose an example p.
2. Calculate the constants for the reduced problem:
   - $A_p = K(\bar{x}_p, \bar{x}_p)$
   - $\bar{B}_p = \sum_{i \neq p} K(\bar{x}_i, \bar{x}_p)\bar{\tau}_i - \beta \bar{1}_{y_p}$
3. Set $\bar{\tau}_p$ to be the solution of the reduced problem :

$$\min_{\tau_p} \quad Q(\bar{\tau}_p) = \frac{1}{2} A_p(\bar{\tau}_p \cdot \bar{\tau}_p) + \bar{B}_p \cdot \bar{\tau}_p$$
$$\text{subject to :} \quad \bar{\tau}_p \leq \bar{1}_{y_p} \quad \text{and} \quad \bar{\tau}_p \cdot \bar{1} = 0$$

Output : $H(\bar{x}) = \arg\max_{r=1}^{k} \left\{ \sum_i \tau_{i,r} K(\bar{x}, \bar{x}_i) \right\}$.

به تعداد تمامی ورودی‌هایمان τ_i داریم که در ابتدا کار برابر با بردار صفر همه آن‌ها را در نظر میگیریم.
یک نمونه مانند نمونه p را انتخاب می‌کنیم . kernel مقدار ورودی p را حساب می‌کنیم و حاصل به دست آمده را در خودش ضرب می‌کنیم .

در مرحله بعد سعی می‌کنیم $\bar{B}_p$ را پیدا کنیم . برای محاسبه آن ضرب kernel ورودی به ازای تمامی ورودی را در kernel ماتریس p بدست می‌آوریم به جز خود آیتم P و در τ_i ضرب می‌کنیم و سپس حاصل بدست آمده را از ضرب وارن C در ماتریس تماما صفر به غیر از درآیه y_p آن کم می‌کنیم که در نهایت از جمع تمامی این بردار ها بردار $\bar{B}_p$ بدست می‌آید که به تعداد کلاس‌های ما درایه دارد .

سپس در قسمت بعد سعی می‌کنیم مسئله optimization زیر را برای پیدا کردن $\bar{\tau}_p)$ جدید حساب کنیم که به مسئله ما با انتخاب $\bar{\tau}_p)$ که شرایط را برقرار کنند کمترین مقدار ممکن را پیدا کند و اینگونه مقدار $\bar{\tau}_p)$ **جدید** بدست می آید و آپدیت می‌شود .
خروجی ما باید یک ماتریس k * m باشد که m تعداد feature هاست .

برای تکمیل جزئیات الگوریتم باید در مورد مسائل زیر بحث کنیم.
اول، ما نیاز به یک معیار توقف برای حلقه داریم .
دوم آنکه، ما به یک طرح برای انتخاب p در هر دور نیاز داریم .
در این مقاله ما طرحی که برای انتخاب نمونه p شرح می دهیم به شیوه ای حریصانه است .

خوب از الگوریتم مسئله قبل مراحلی باقی ماند مثل :

$$\min_{\tau} \qquad \mathcal{Q}(\tau) = \frac{1}{2}\sum_{i,j} K_{i,j}\ (\bar{\tau}_i \cdot \bar{\tau}_j) - \beta \sum_i \bar{\tau}_i \cdot \bar{1}_{y_i}$$

برای حل آن از Karush-Kuhn-Tucker theorem استفاده می‌کنیم .برای اینکه τ پیدا کنیم که مسئله ما optimum شود .

تابع Lagrangian آن برابراست با :

$$\begin{aligned}\mathcal{L}(\tau,u,v) &= \frac{1}{2}\sum_{i,j}K_{i,j}\sum_{r}\tau_{i,r}\tau_{j,r}-\beta\sum_{i,r}\tau_{i,r}\delta_{y_i,r}\\ &\quad+\sum_{i,r}u_{i,r}(\tau_{i,r}-\delta_{y_i,r})-\sum_{i}v_i\sum_{r}\tau_{i,r}\\ \text{subject to :}\quad &\forall i,r\quad u_{i,r}\geq 0\ .\end{aligned}$$

در نتیجه با مشتق گیری از آن خواهیم داشت :

$$\frac{\partial}{\partial\tau_{i,r}}\mathcal{L} = \sum_{j}K_{i,j}\tau_{j,r}-\beta\delta_{y_i,r}+u_{i,r}-v_i=0\ .$$

حال اجازه دهید مجموعه متغیرهای کمکی زیر را تعریف کنیم،

$$F_{i,r} = \sum_{j}K_{i,j}\tau_{j,r}-\beta\delta_{y_i,r}$$

برای هر نمونه $\vec{x}i$، مقدار Fi,r نشان دهنده اطمینان در تخصیص برچسب r به xi است.

مقدار β از اطمینان برچسب صحیح کم می شود تا margin حداقل β به دست آید.

در نتیجه خواهیم داشت :

$$F_{p,r}=B_{p,r}+k_{p,p}\tau_{p,r}\ .$$

ما از این رابطه بین متغیرهای F و B در بخش بعدی استفاده خواهیم کرد که ما یک راه حل کارآمد برای مسئله درجه دوم را مورد بحث قرار می دهیم.

باید این موارد زیر را در نظر گرفت :

$$\begin{aligned}&\forall i,r\qquad F_{i,r}+u_{i,r}=v_i\ ,\\ &\forall i,r\qquad u_{i,r}(\tau_{i,r}-\delta_{y_i,r})=0\ ,\\ &\forall i,r\qquad u_{i,r}\geq 0\ .\end{aligned}$$

با انجام یکسری محاسبات از شرط‌های بالا در نهایت به رابطه مهم زیر می‌رسیم .

$$\psi_i=\max_r F_{i,r}-\min_{r\ :\ \tau_{i,r}<\delta_{y_i,r}}F_{i,r}\ .$$

در اجرای عددی واقعی کافی است $\bar{\tau}_i$ را پیدا کنیم که شرط $\psi_i \leq \epsilon$ برای آن برقرار باشد که در اینجا اپسیلون در واقعی دقت از پیش تعریف شده پارامترها را مشخص می‌کند .

پس ما حلقه اصلی در الگوریتم قبل را تا زمانی که به ازای یک مثال (xi, ȳi) کمتر از مقدار اپسیلون نشده است ادامه می‌دهیم .

متغیرهای ψi نیز به عنوان ابزاری برای انتخاب نمونه ای برای به روز رسانی عمل می کنند .

شاخص مثال p را نیز به گونه‌ای انتخاب می کنیم که ψp برای آن حداکثر است .

در قسمت بعد ما یک الگوریتم fix point کارآمد را توصیف می‌کنیم که یک راه حل تقریبی برای معادله زیر پیدا می کند.

$$\begin{aligned} \min_{\tau} \quad & \mathcal{Q}(\bar{\tau}) = \frac{1}{2} A_p(\bar{\tau}_p \cdot \bar{\tau}_p) + \bar{B}_p \cdot \bar{\tau}_p \\ \text{subject to :} \quad & \bar{\tau}_p \leq \bar{1}_{y_p} \text{ and } \bar{\tau}_p \cdot \bar{1} = 0 \ . \end{aligned}$$

مسئله را به شکل زیر دوباره بازنویسی می‌کنیم .

$$\begin{aligned} \mathcal{Q}(\bar{\tau}) &= -\frac{1}{2} A(\bar{\tau} \cdot \bar{\tau}) - \bar{B} \cdot \bar{\tau} \\ &= -\frac{1}{2} A[(\bar{\tau} + \frac{\bar{B}}{A}) \cdot (\bar{\tau} + \frac{\bar{B}}{A})] + \frac{\bar{B} \cdot \bar{B}}{2A} \ . \end{aligned}$$

تغییر متغیر زیر را می‌دهیم .

$$\bar{\nu} = \bar{\tau} + \frac{\bar{B}}{A} \qquad \bar{D} = \frac{\bar{B}}{A} + \bar{1}_y$$

از آنجا که در جواب optimal ما وجود یا عدم وجود ضریب A تاثیر گذار نیست آن را حذف می‌کنیم تا به معادله زیر برسیم .

$$\begin{aligned} \min_{\nu} \quad & \mathcal{Q}(\bar{\nu}) = \|\bar{\nu}\|^2 \\ \text{subject to :} \quad & \bar{\nu} \leq \bar{D} \text{ and } \bar{\nu} \cdot \bar{1} = \bar{D} \cdot \bar{1} - 1 \end{aligned}$$

میدانیم که :

$$F_{i,r} = B_{i,r} + A_i \tau_{i,r}$$

ما می توانیم ψi را از Bi محاسبه کنیم و بنابراین باید Bi,r یا Fi,r را ذخیره کنیم.

اجازه دهید با θ و αi متغیرهای مسئله دوگانه معادله را نشان دهیم را نشان دهیم .

$$\forall r \quad \nu_r \leq D_r \;\; ; \;\; \alpha_r(\nu_r - D_r) = 0 \;\; ; \;\; \nu_r + \alpha_r - \theta = 0 \quad .$$

که ما را به نتیجه زیر می‌رساند .

$$\nu_r \leq \min\{\theta, D_r\}$$

که براساس یک قضیه که اثبات کرده به نتیجه زیر می‌رسیم .

$$\sum_{r=1}^{k} \min_r\{\theta, D_r\} = \sum_{r=1}^{k} D_r - 1$$

و با تعمیم آن خواهیم داشت قضیه زیر را :

Theorem 2 *Let θ^* be the fixed point of Eq. (40) ($\theta^* = F(\theta^*)$). Assume that $\theta_1 \leq \max_r D_r$ and let $\theta_{l+1} = F(\theta_l)$. Then for $l \geq 1$*

$$\frac{|\theta_{l+1} - \theta^*|}{|\theta_l - \theta^*|} \leq 1 - \frac{1}{k} \;,$$

where k is the number of classes.

حال با توجه به موارد بالا می‌توانیم تابع $\texttt{FixedPointAlgorithm}(\bar{D}, \theta, \epsilon)$ را پیاده سازی کنیم .

$\texttt{FixedPointAlgorithm}(\bar{D}, \theta, \epsilon)$

Input $\bar{D}$, θ_1, ϵ.

Initialize $l = 0$.

Repeat

- $l \leftarrow l + 1$.
- $\theta_{l+1} \leftarrow \frac{1}{k}\left[\sum_{r=1}^{k} \max\{\theta_l, D_r\}\right] - \frac{1}{k}$.

Until $\left|\frac{\theta_l - \theta_{l+1}}{\theta_l}\right| \leq \epsilon$.

Assign for $r = 1, \ldots, k$: $\nu_r = \min\{\theta_{l+1}, D_r\}$

Return: $\bar{\tau} = \bar{\nu} - \frac{\bar{B}}{A}$.

در این تابع بردار D را طبق رابطه ریاضی بالا حساب و به تابع می‌دهیم و همچنین θ اولیه را به عنوان ورودی می‌گیریم و اپسیلون که میزان دقت را برای ما مشخص می‌کند .

در هر مرحله مقدار θ را متناسب با فرمول بالا حساب می‌کنیم و تا زمانی مقدار قدرمطلق θ پیشین از جدید تقسیم بر میزان θ جدید کمتر از اپسیلون نشده باشد یا به دقت مورد نظر ما نزدیک نشده باشد این مرحله را تکرار می‌کنیم .

در نهایت بردار V را می‌سازیم و به کمک آن همانند فرمول موجود در قسمت return مقدار τ آپدیت شده را حساب می‌کنیم و برمی‌گردانیم .

بعد از دیدن این روند دیگر می‌توانیم الگوریتم کلی را ببینیم که به شکل زیر خواهد بود در این الگوریتم باید توجه داشت که مقدار دقت و β به عنوان ورودی میگیریم .

Input $\{(\bar{x}_1, y_1), \ldots, (\bar{x}_m, y_m)\}$.
Initialize for $i = 1, \ldots, m$:

- $\bar{\tau}_i = \bar{0}$
- $F_{i,r} = -\beta\delta_{r,y_i}$ $(r = 1 \ldots k)$
- $A_i = K(\bar{x}_i, \bar{x}_i)$

Repeat:

- Calculate for $i = 1 \ldots m$: $\psi_i = \max_r F_{i,r} - \min_{r\ :\ \tau_{i,r} < \delta_{y_i,r}} F_{i,r}$
- Set: $p = \arg\max\{\psi_i\}$
- Set for $r = 1 \ldots k$: $D_r = \frac{F_{p,r}}{A_p} - \tau_{p,r} + \delta_{r,y_p}$ and $\theta = \frac{1}{k}\left(\sum_{r=1}^{k} D_r\right) - \frac{1}{k}$
- Call: $\bar{\tau}_p^\star =$`FixedPointAlgorithm`$(\bar{D}, \theta, \epsilon/2)$. (See Figure 3)
- Set: $\Delta\bar{\tau}_p = \bar{\tau}_p^\star - \bar{\tau}_p$
- Update for $i = 1 \ldots m$ and $r = 1 \ldots k$: $F_{i,r} \leftarrow F_{i,r} + \Delta\tau_{p,r}\, K\left(\bar{x}_p, \bar{x}_i\right)$
- Update: $\bar{\tau}_p \leftarrow \bar{\tau}_p^\star$

Until $\psi_p < \epsilon\beta$

Output : $H(\bar{x}) = \arg\max_r \left\{ \sum_i \tau_{i,r} K\left(\bar{x}, \bar{x}_i\right) \right\}$.

همانند الگوریتم اول ابتدا مقدار اولیه تمامی $\bar{\tau}_i$ برابر با صفر در نظر می‌گیریم در این الگوریتم مقدار Fi,r نیز داریم که مطابق ضرب بالا مقدار اولیه آن را حساب می‌کنیم و مقدار Ai هم که مانند قبل بدست می‌آوریم .
مقدار ψi برای تمامی نمونه‌ها حساب می‌کنیم در درون while true خودمون و سپس ماکس ψi را پیدا می‌کنیم و اندیس آن را برمیداریم و سپس مقدار بردار Dr را حساب و از روی آن θ اولیه را حساب می‌کنیم .
در انتها برحسب بردار D , θ به کمک تابع fixedPioint که در بالا شرح دادیم روند کار را مقدار آپدیت شده $\bar{\tau}_i$ را حساب می‌کنیم . سپس اختلاف مقدار بدست آمده از مقدار قبلی $\bar{\tau}_i$ را حساب می‌کنیم و از حاصل بدست آمده مقادیر Fi.r را آپدیت می‌نماییم .
و تا زمانی که ماکس ψ کمتر از اپسیلون در β نشده مواردی که در حلقه ادامه پیدا می‌کند.

خروجی فانکشن زیر $\bar{\tau}_i$ که اگر در فرمول اولیه یعنی

$$\max_{\tau} \quad Q(\tau) = -\frac{1}{2}\sum_{i,j} K\left(\bar{x}_i, \bar{x}_j\right)\ \left(\bar{\tau}_i \cdot \bar{\tau}_j\right) + \beta \sum_i \bar{\tau}_i \cdot \bar{1}_{y_i}$$

$$\text{subject to :} \quad \forall i \quad \bar{\tau}_i \leq \bar{1}_{y_i} \quad \text{and} \quad \bar{\tau}_i \cdot \bar{1} = 0 \ ,$$

قرار بگیرد حاصل بیشترین مقدار ممکن می‌شود .

در ادامه مقاله درباره روش‌هایی و تکنیک هایی که می‌تواند این پیاده سازی را سریع‌تر و بهتر کند توضیح داده مانند

Maintaining an active set

Caching

و ...

در انتها نیز درباره نتایج آزمایش‌هایی را که به منظور ارزیابی انجام داده شده به کمک این الگوریتم صحبت کرده که نشان دهنده قدرت این الگوریتم است .

CRAMMER AND SINGER

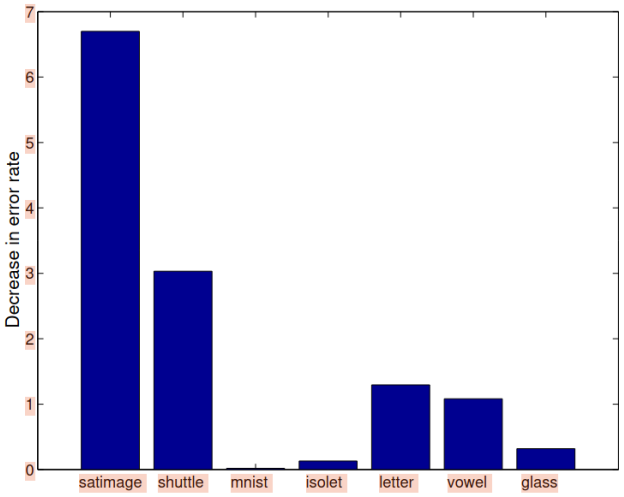


Figure 8: Comparison of a multiclass SVM build using the one-against-rest approach with the multiclass support vector machines studied in this paper.

لینک‌هایی که برای پیاده سازی این پروژه از آن‌ها استفاده کردم :

https://pawanccs19.medium.com/handwritten-digit-recognition-in-python-using-scikit-learn-fd7147e01149

https://ishika-tailor.medium.com/handwritten-digit-recognition-on-mnist-dataset-61b8d6a884b8

https://towardsdatascience.com/dimensionality-reduction-for-data-visualization-pca-vs-tsne-vs-umap-be4aa7b1cb29

https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

https://towardsdatascience.com/multi-class-classification-one-vs-all-one-vs-one-94daed32a87b

https://vitalflux.com/svm-rbf-kernel-parameters-code-sample/

https://scikit-learn.org/stable/auto_examples/classification/plot_digits_classification.html#sphx-glr-auto-examples-classification-plot-digits-classification-py

https://scikit-learn.org/stable/modules/svm.html

https://towardsdatascience.com/building-a-machine-learning-pipeline-3bba20c2352b

https://www.w3resource.com/python-exercises/numpy/python-numpy-exercise-94.php

https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html

https://medium.com/swlh/how-to-create-an-auc-roc-plot-for-a-multiclass-model-9e13838dd3de

https://gist.github.com/matsub/206a1dac75093d74d8ae2ab9c5a2ae35