به نام خدا

پروژه SVM

ابوالفضل ارشیا

بخش اول:

در این بخش مسئله های دو کلاسه در فضای دو بعدی ایجاد کرده و با استفاده از SVM آن ها را دسته بندی میکنیم.
برای این کار ابتدا با یکسری نقاط به دو مرکز ایجاد میکنیم تا به دسته بندی آن ها بپردازیم. برای ایجاد داده هایی با حالت دایره ای (برای کرنل rbf) از تابع make_circles استفاده میکنیم.

مقادیر موجود در آرگومان تابع make_blobs به ترتیب برای تعداد نقاط و تعداد کلاس و همچنین انتخاب رندوم نقاط استفاده می شوند و با استفاده از آن می توانیم نقاط متفاوتی درنظر بگیریم.

بعد از ساخت نقاط با تابع svm.SVC مقادیر و نوع کرنل را برای دسته بندی تعیین میکنیم و بعد از آن با تابع fit نقاطی که در ایتدا تولید کردیم را دسته بندی میکنیم.

در انتها نقاط و support vector ها را بر روی نمودار نمایش می دهیم.

تصاویر نمودار ها با کرنل های استفاده شده در پوشه این بخش قرار دارند.

بخش دوم:

در این برای دیتاست از دیتای آماده تشخیص اعداد در پکیج sklearn استفاده کرده ایم.

در ابتدای برنامه تعدادی از این عکس ها را با مقدار label یا target نمایش میدهیم. سپس با استفاده از تابع reshape هر عکس را از حالت یک آرایه دوبعدی به یک آرایه یک بعدی تبدیل میکنیم(طبق توضیحات داک پروژه)

سپس کرنل و پارامتر های موجود در SVMرا ست کرده و دیتای عکس ها را بر اساس پارمتر test_size به دو دسته برای داده تست و آموزشی تقسیم میکنیم.

پارامتر های C , gamma در عملکرد svm تاثیر زیادی دارند و باید مقادیر آن ها با آزمایش و چک کردن دقت به دست بیایند.

در انتها نیز داده ها را روی SVM فیت میکنیم و خروجی مقادیر تست را با مقدار واقعی با تابع accuracy_score چک کرده و با استفاده از تابع classification_report نیز برای هر عدد گزارش کاملی از دقت پیش بینی آن ارائه میکنیم.

دقت نهایی برای C=1000, gamma = 0.001 و 80 درصد داده آزمایشی و 20 درصد داده تستی

```
number of images : 1797
Accurarcy : 0.9638888888888889
Classification report for classifier SVC(C=1000, gamma=0.001):
              precision    recall  f1-score   support

           0       1.00      0.97      0.99        35
           1       0.97      1.00      0.99        36
           2       1.00      1.00      1.00        35
           3       1.00      0.84      0.91        37
           4       0.97      0.92      0.94        37
           5       0.93      1.00      0.96        37
           6       1.00      1.00      1.00        37
           7       1.00      1.00      1.00        36
           8       0.86      0.97      0.91        33
           9       0.92      0.95      0.93        37

    accuracy                           0.96       360
   macro avg       0.97      0.96      0.96       360
weighted avg       0.97      0.96      0.96       360
```

بررسی تجربی کرنل ها:

کرنل linear برای داده هایی که از هم فاصله بیشتری داند مناسب تر است و از بیش برازش جلوگیری میکند.

کرنل poly به جای خط صاف یک خط منحنی رسم میکند و در صورتی که داده ها درهم تنیدگی داشته باشند مناسب تر است

کرنل rbf برای داده هایی که به صورت دایره ای یا مرکزی درون هم قرار دارند مناسب تر است.

چالش ها:

ساخت داده هایی دوکلاسه که هر کلاس حول یک نقطه تجمع داشته باشند.

برای تست کردن کرنل rbf باید داده های بصورت دو دایره تو در تو باشند که کرنل به خوبی عمل کند. برای این مطلب باید تابعی برای ایجاد اینگونه نقاط استفاده کرد.

برای استفاده از دیتاست MNIST با مشکل مواجه شدم چون مسیر ایمپورت کردن این این دیتاست از Tensorflow1 تغییر کرده است و برای دیتاست بخش دوم به جای خواندن از فایل عکس ها و لیبل گزاری برای آنها از دیتاست آماده عکس هایی از اعداد که در لایبری sklearn موجود است استفاده کردم.