**به نام خدا**

# گزارش تمرین SVM

**ریحانه آهنی ۹۸۲۳۰۰۹**

**پاییز ۱۴۰۱**

ابتدا کتابخانه های مورد نیاز وارد می کنیم:

```
from sklearn.preprocessing import OrdinalEncoder, MinMaxScaler
from sklearn.model_selection import train_test_split, KFold
from sklearn.metrics import accuracy_score
from sklearn.svm import SVC

import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

سپس با استفاده از کتابخانه pandas داده را بارگزاری می کنیم.

```
dataset = pd.read_csv('Frogs_MFCCs.csv')
```

سپس از داده ها ورودی ها و خروجی را مشخص کرده و مقادیر String را به مقادیر عددی برای تشخیص توسط SVM تبدیل می کنیم ( اینکار با استفاده از OrdinalEncoder صورت می گیرد ) سپس مقادیر را با استفاده از MinMaxScaler به محدوده ۰ تا ۱ تبدیل می کنیم:

```
inputs = ['MFCCs_ 1', 'MFCCs_ 2', 'MFCCs_ 3', 'MFCCs_ 4', 'MFCCs_ 5', 'MFCCs_ 6',
       'MFCCs_ 7', 'MFCCs_ 8', 'MFCCs_ 9', 'MFCCs_10', 'MFCCs_11', 'MFCCs_12',
       'MFCCs_13', 'MFCCs_14', 'MFCCs_15', 'MFCCs_16', 'MFCCs_17', 'MFCCs_18',
       'MFCCs_19', 'MFCCs_20', 'MFCCs_21', 'MFCCs_22', 'Family', 'Genus',
       'Species']
output = ['RecordID']

X = dataset[inputs].to_numpy()
X = OrdinalEncoder().fit_transform(X)
X = MinMaxScaler().fit_transform(X)

Y = dataset[output].to_numpy().ravel()
```

سپس داده ها را به ۷۰ درصد داده یادگیری و ۳۰ درصد داده برای بررسی تقسیم می کنیم ( برای اینکه مدل از همه داده ها بتواند نمونه های داشته باشد آن را به طور تصادفی انتخاب می کنیم ):

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.3)
```

حال برای اجرای بقیه بخش ها آماده هستیم.

## بخش اول) SVM خطی

```
clf = SVC(kernel='linear')
clf.fit(X_train, Y_train)

Y_train_pred = clf.predict(X_train)
print(f'Train accuracy: {accuracy_score(Y_train, Y_train_pred)}')

Y_test_pred = clf.predict(X_test)
print(f'Test accuracy: {accuracy_score(Y_test, Y_test_pred)}')

print(f'Number of support vectors per class: {clf.n_support_}')
```

بخش استفاده از مدل SVM خطی کافی است از کلاس SVC ( که الگوریتم SVM را برای طبقه بندی پیاده سازی می کند ) استفاده کنیم. Kernel خطی را انتخاب می کنیم و بر روی داده Train مدل را fit می کنیم. پس از آن میزان دقت در داده یادگیری و اعتبار را بررسی میکنیم. سپس با استفاده از ویژگی n_support تعداد support vector ها را برای هر کلاس نمایش می دهیم.

نتایج به شکل زیر است:

```
Train accuracy: 0.8733121525019857

Test accuracy: 0.8615099583140343

Number of support vectors per class: [ 30  35  16  45  15  28  12  68  25  45  59  16  51  99 144  56  83 117
 175 171 139 164 173  89  23  10  29  41   4  19  17  21   5  21  25  92
  69  40   8 230 211 203  69   6  22  16  45   3  36  33   8   6  26   1
   1  22  13  23  16   9]
```

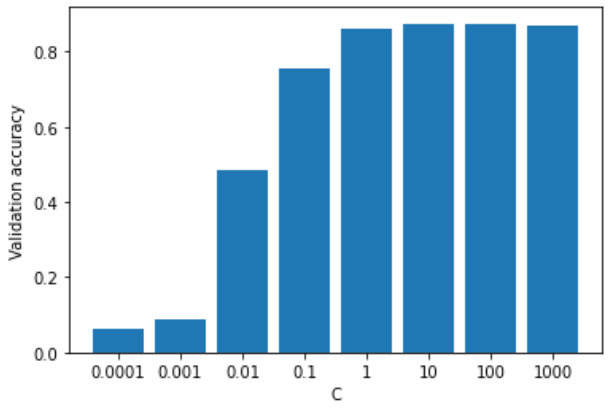
مشاهده می شود که مدل با دقت مشابهی هم داده های یادگیری و هم داده های اعتبار را پیش بینی می کند.

## بخش دوم) SoftSVM خطی

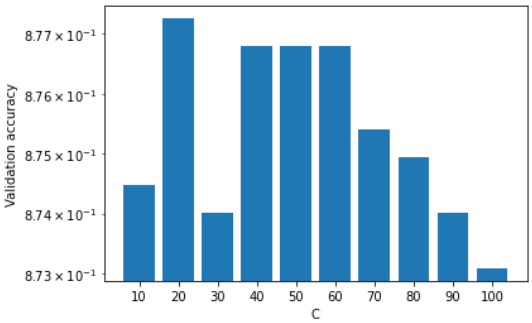
برای این بخش مقادیر مختلف برای پارامتر C که میزان مجاز Miss-Classification را تعیین می کند را تست می کنیم. مقادیر تست شده به شرح زیر است:

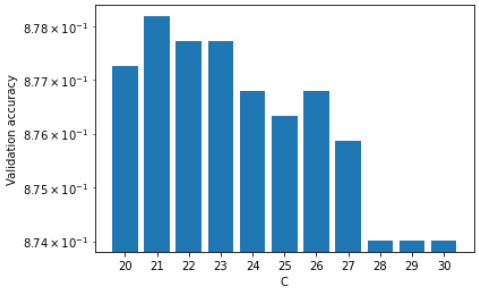
```
0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000
```

سپس دقت هر مدل در داده صحت را بر حسب مقدارC هر مدل نمایش می دهیم:




مشاهده میشود که در مقادیر بسیار کم C، میزان Miss-Classification بسیار بالاست و دقت مدل کم است. با افزایش C دقت افزایش یافته و به سمت Hard Margin حرکت می کنیم، تا زمانی که افزایش بیشتر C موجب کاهش دقت می شود. همانطور که در کلاس توضیح داده شد، حد مناسب C را می توان با استفاده از داده های اعتبار بدست آورد، در واقع با افزایش C به سمت Hard Margin حرکت می کنیم و می دانیم که در حالت Hard Margin اجازه هیچ گونه Miss-Classification را نداریم در نتیجه مدل به سمت غیر جنرال بودن حرکت می کند و به مرور تعمیم پذیری آن کاهش پیدا می کند و بنابراین روی داده های تست نتایج خوبی دریافت نخواهیم کرد. سپس می توان نمودار را برای مقادیر بین ۱۰ تا ۱۰۰ نیز رسم کرد تا بهترین مقادیر برای C پیدا کرد. که برای این مدل در حدود ۲۱ بدست آوردیم.

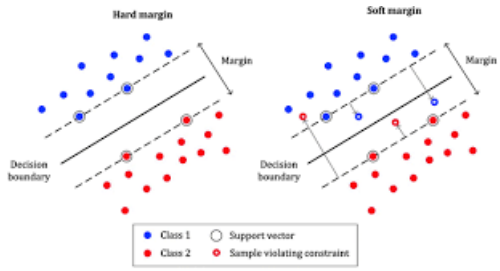



تعداد Support Vector ها در بهترین مدل به شکل زیر است:

```
Number of support vectors per class: [ 24  29  12  32   9  20   8  56  26  26  50  15  48  61 109  44  61  81
 118 164  97 114 136  62  19   6  30  36   6  18  18  18   4  15  23  77
  64  30  10 203 186 180  67  10  18  15  34   3  29  36  10   6  22   2
   1  19  12  23  12   9]
```

مشاهده میشود که تعداد Support Vector ها به طور کلی در همه کلاس ها کمتر از حالت قبلی می باشد. همانطور که در کلاس توضیح داده شد، در حالت Soft، داده هایی که درون حاشیه قرار گرفته اند و باعث Margin Violation شده اند هم به عنوان Support Vector ها نیز در نظر گرفته میشوند، اما در حالت Hard تنهایی داده هایی به عنوان Support Vector هستند که دقیقا بر روی مرز حاشیه قرار گرفته اند و همانطور که می دانیم در Hard Margin ما هیچ داده ای نداریم که ما را دچار Margin Violation بکند.



## بخش سوم) SVM غیرخطی با کرنل های RBF و چند جمله ای ( Poly )

در عمل، الگوریتم SVM با هسته پیاده-سازی می شود که فضای داده ورودی را به فرم مورد نیاز تبدیل می کند. SVM از تکنیکی به نام ترفند هسته استفاده می-کند که در آن هسته فضای ورودی را با تعداد ابعاد کم می گیرد و آن را به فضای بعدی بالاتر تبدیل می کند. به عبارت ساده، هسته مسائل غیر قابل تفکیک را با افزودن ابعاد بیشتر به مسائل قابل تفکیک تبدیل می کند. بنابراین SVM قدرتمندتر، انعطاف پذیر و دقیق تر می شود.

### کرنل چندجمله ای:

این هسته، فضای ورودی منحنی یا غیر خطی را از هم متمایز می-کند. در زیر فرمول هسته چند جمله-ای آورده شده است.

$$k(X, Xi) = 1 + sum(X * X_i)^d$$

در اینجا d درجه چند جمله ای است، که باید به صورت دستی در الگوریتم یادگیری مشخص کنیم.

نتایج کرنل چند جمله ای به شکل زیر است:

```
Train accuracy: 0.886219221604448

Test accuracy: 0.8638258452987494

Number of support vectors per class: [ 27  36  16  49  16  28  11  46  28  47  63  16  45 103 145  59  86 121
 186 176 133 174 170  97  23   9  29  41   4  19  17  21   5  22  25  95
  73  38   8 231 217 200  69   6  22  16  50   3  36  35   8   6  25   1
   1  20  13  24  18  12]
```

### کرنل RBF:

کرنل RBF که بیشتر در طبقه بندی SVM استفاده می شود، فضای ورودی را در فضایی با ابعاد نامشخص ترسیم می کند. فرمول زیر آن را به صورت ریاضی توضیح می دهد:

$$K(x, xi) = exp(-gamma * sum(x - xi^2))$$

در اینجا دامنه گاما از ۰ تا ۱ است. ما باید آن را به صورت دستی در الگوریتم یادگیری مشخص کنیم.

```
Train accuracy: 0.9475774424146147

Test accuracy: 0.8906901343214451

Number of support vectors per class: [ 22  25  13  32   9  23   9  37  26  33  47   9  50  62  86  49  60  89
 123 140  86 118 132  54  19   6  28  39   4  19  17  15   4  15  18  69
  60  31   8 189 184 138  68   6  17  14  31   3  29  30   8   6  24   1
   1  15  10  22  13   7]
```

مشاهده میشود که کرنل Poly برای این دیتاست بهتر عمل می کند و برای بخش بعدی از کرنل Poly استفاده می کنیم. همچنین کرنل Poly تعداد support vector کمتری دارد.

و نسبت به کرنل خطی بهتر عمل می کند.

## بخش چهارم) کرنل Poly و Soft Max

برای این بخش از کرنل Poly و مقدار C برابر با ۱۰ استفاده می کنیم.

```
Train accuracy: 0.9795472597299444

Test accuracy: 0.8994905048633627

Number of support vectors per class: [ 20  23  11  36   9  24  10  35  25  28  53   8  50  50  89  38  48  92

 102 150  81  97 121  50  18   6  25  37   4  18  15  15   4  14  17  70

  48  26   8 168 164 112  74   9  17  14  35   3  29  33   8   7  15   1

   1  16  12  22  12   6]
```

مشاهده میشود که این مدل از تمامی مدل های قبلی بهتر عمل می کند و میزان تعمیم پذیری آن از دیگر مدل های بهتر است.

اما بررسی نتایج مدل ها به این شیوه به درست نخواهد بود. هنگام ارزیابی تنظیمات مختلف برای مدل ها، مانند C که باید به صورت دستی برای یک SVM تنظیم شود، همچنان خطر تطبیق بیش از حد در مجموعه آزمایشی وجود دارد زیرا پارامترها را می توان تا زمانی که برآوردگر عملکرد بهینه انجام دهد، تغییر داد. به این ترتیب، دانش در مورد مجموعه تست می تواند به مدل "نشت" کند و معیارهای ارزیابی دیگر عملکرد تعمیم را گزارش نمی دهند. برای حل این مشکل، بخش دیگری از مجموعه داده را می‌توان به‌عنوان «مجموعه اعتبارسنجی» در نظر گرفت: آموزش در مجموعه آموزشی ادامه می‌یابد، پس از آن ارزیابی روی مجموعه اعتبارسنجی انجام می‌شود، و زمانی که آزمایش موفقیت‌آمیز به نظر می‌رسد، ارزیابی نهایی را می توان روی مجموعه تست انجام داد.

## بخش ۵) پیاده سازی CrossValidation و KFold

KFold تمام نمونه‌ها را به گروه‌هایی از نمونه‌ها تقسیم می‌کند که به آنها folds می‌گویند (اگر این معادل استراتژی Leave One Out است) با اندازه‌های مساوی (در صورت امکان).
مدل با استفاده از فولدها آموخته می‌شود و تا کردن خارج شده برای تست استفاده می‌شود.

در اینجا از 4Fold استفاده می کنیم. نتایج به شکل زیر است:

```
Fold: 0
  Train accuracy: 0.9799851742031134
  Test accuracy: 0.8882712618121178
Fold: 1
  Train accuracy: 0.9797998517420311
  Test accuracy: 0.8943857698721512
Fold: 2
  Train accuracy: 0.9812824314306894
  Test accuracy: 0.8854919399666481
Fold: 3
  Train accuracy: 0.9798035945895868
  Test accuracy: 0.8898776418242491
Mean train: 0.9802177629913551
Mean train: 0.8895066533687915
```

مشاهده میشود که میزان دقت در هر Fold تغییر میکند اما به طور میانگین دقت ۹۸ درصد بر روی داده یادگیری و دقت ۸۹ درصد بر روی داده اعتبارسنجی دریافت می کند.

## نتیجه گیری آخر

در پایان مشاهده میشود که مدل با کرنل Poly و مقدارC برابر با ۱۰ از تمامی مدل ها بهتر عمل می کند.

در این دیتاست مشاهده میشود که Hard Margin بهتر عمل می کند و مدل دقت بیشتر دارد، از جمله علل این ویژگی می تواند نزدیک بودن داده ها بهم و کمبود Outlier ها باشد.