**به نام خدا**

# تمرین پیاده سازی 3

# ماشین بردار پشتیبان

**درس هوش مصنوعی**

**احمد زعفرانی97105985**

**تابستان 99**

# راهنمای اجرا

**پیشنیازها:** از کتابخانه های زیر در این کد استفاده شده است:

sklearn – numpy – matplotlib – PIL(pillow)

**بخش اول :** در پوشه بخش اول، 5 فایل وجود دارند که هر کدام نشانگر یکی از مثال های استفاده شده برای این بخش هستند. برای مشاهده توابع استفاده شده در هر مثال، به بخش توضیحات برنامه مراجعه فرمایید. عدد چاپ شده بیانگر درصد دقت دسته بندی الگوریتم می باشد.

**بخش دوم :** در این پوشه یک فایل قرار دارد که برای اجرای آن، باید خطوط 22 و 23 کد تغییر کنند؛ به اینصورت که در خط 22 باید آدرس پوشه train از پایگاه داده USPS داده می شود، و در خط 23 آدرس پوشه test این پایگاه داده مشخص می شود. پس از اجرای برنامه، درصد دقت عملکرد الگوریتم چاپ خواهد شد.

**بخش سوم :** در این پوشه نیز باید برای اجرای درست برنامه، خط 8 برنامه به آدرس درست پوشه ای که پایگاه داده مربوط به تصاویر ارقام پلاک وجود دارند، تغییر کند. پس از اجرای موفقیت آمیز، مانند قسمت های قبل، درصد دقت SVM در خروجی چاپ می شود.

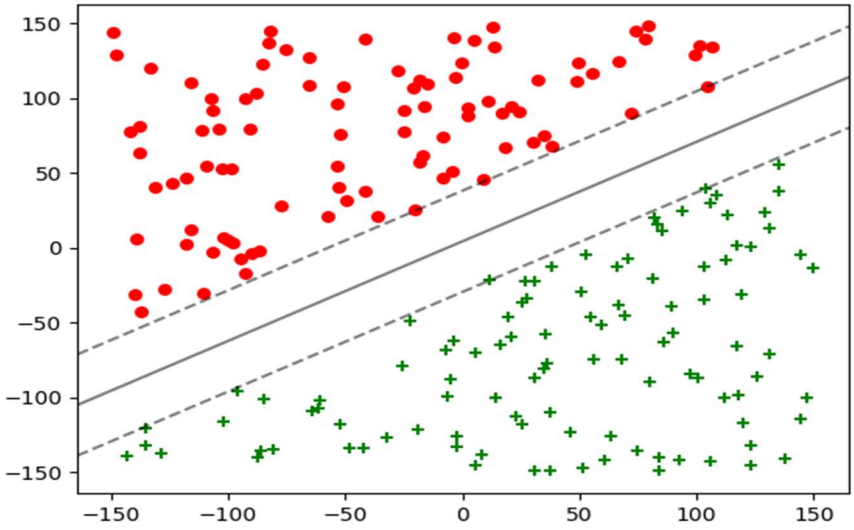
# توضیحات برنامه

### بخش اول

در بخش اول، اولین چالش نرمال سازی داده های آموزشی و آزمایشی است. این پارامتر تاثیر به سزایی در دقت دسته بندی الگوریتم SVM ایفا می کند. جالب آنکه اگر بازه تغییرات مختصات داده های آموزشی و آزمایشی از یک اردر باشد، دقت این الگوریتم به طرز چشمگیری افزایش پیدا می کند.

ابتدا سعی شد تا برای بخش یک، کد را به گونه ای پیاده سازی کنیم که با دریافت تابع دلخواه از کاربر، دسته بندی نقاط را انجام دهد؛ اما تعداد زیاد پارامتر های آزاد و محدودیت نحوه دریافت ورودی (مثلا محدودیت در دریافت توابع چندضابطه ای) ما را وادار کرد تا به آوردن چند مثال برای این بخش بسنده کنیم. در تمامی مثال های زیر، ابتدا 500 نقطه آموزشی بوسیله تابع مشخص شده بصورت تصادفی تولید می کنیم، سپس 200 نقطه آزمایشی دیگر (باز هم بصورت تصادفی) تولید کرده، آنها را بوسیله الگوریتم SVM دسته بندی می کنیم:

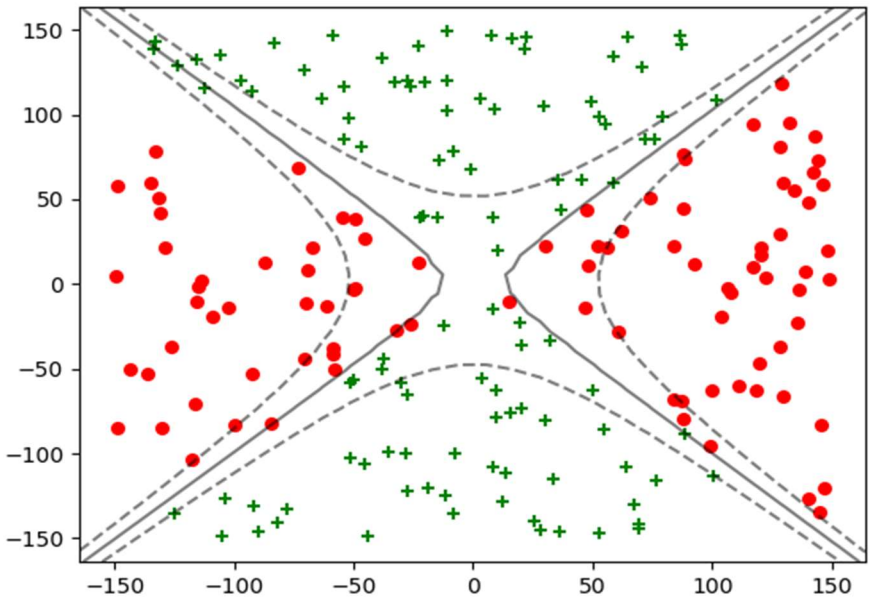
1. تابع : 2*x - 3*y + 13
   دقت: 100٪
   ترفند هسته: خطی




2. تابع : |y| - |x|

دقت : 98٪

ترفند هسته : چند جمله ای – درجه : 2 – بایاس ثابت[1] : 2



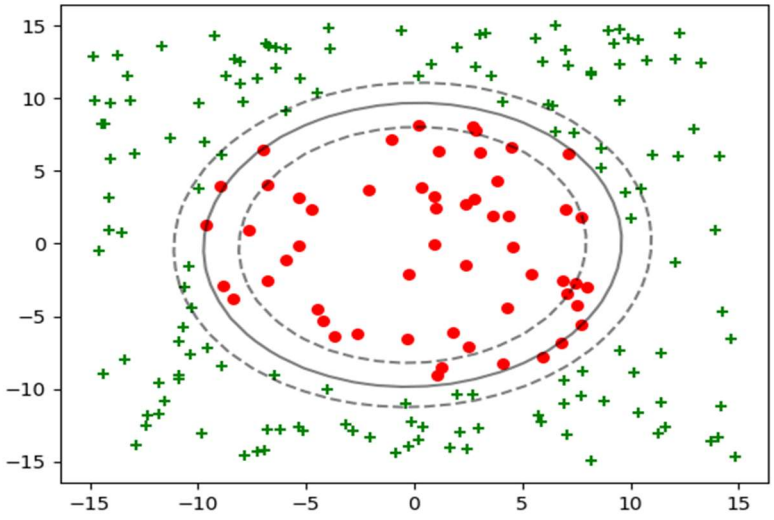

[1] **coef0:** ***float, default=0.0***

Independent term in kernel function. It is only significant in ‘poly’ and ‘sigmoid’.

3. تابع : x**2 + y**2 - 100

دقت : 98٪

ترفند هسته : چند جمله ای – درجه : 2 – بایاس ثابت : 2

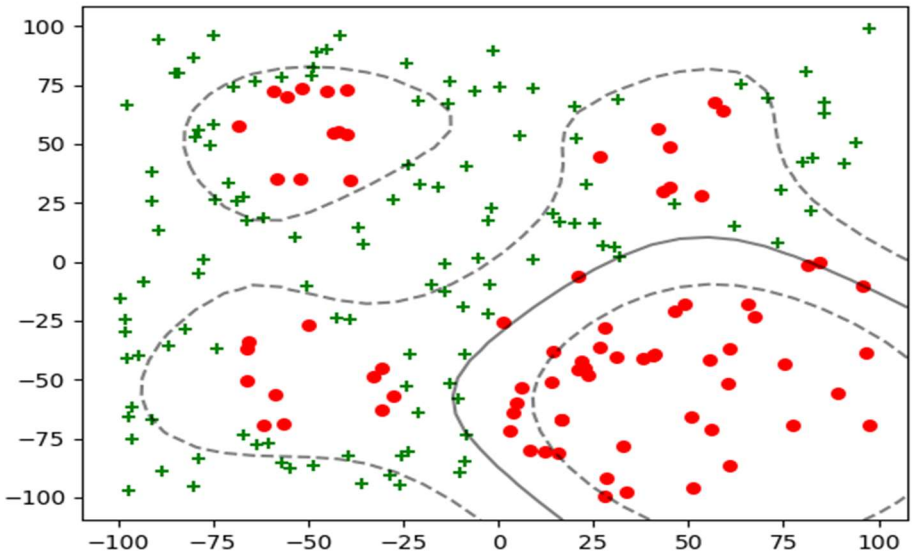



4. تابع :

$$f(x) = \begin{cases} (x-25)^2 + (y-25)^2 - 625 : & x, y > 0 \\ (x+25)^2 + (y-25)^2 - 625 : & x < 0, y > 0 \\ (x+25)^2 + (y+25)^2 - 625 : & x, y < 0 \\ y \quad : & else \end{cases}$$

دقت : 83٪

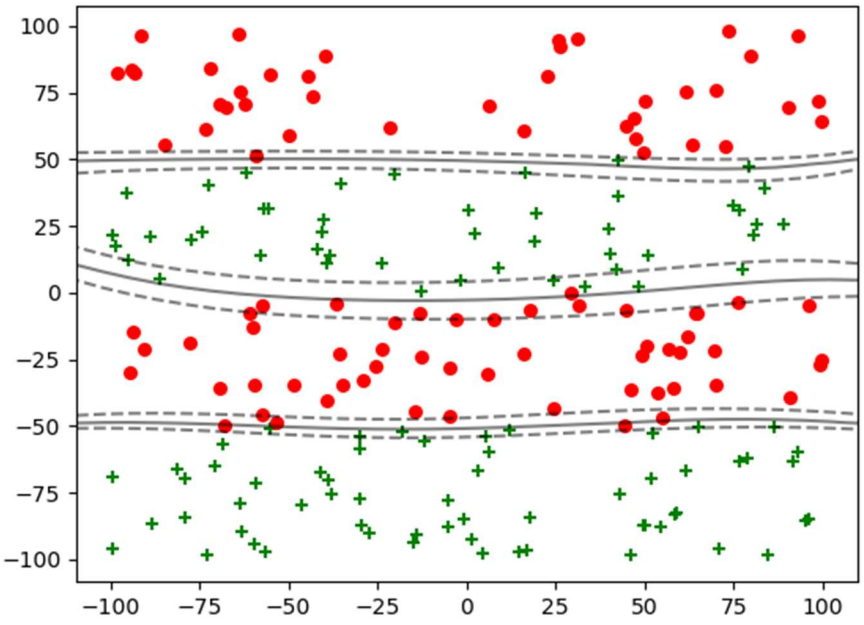
ترفند هسته : گاوسی



5. تابع :
$$f(x)=\begin{cases}1: & y<-50\\ -1: & -50<y<0\\ 1: & 0<y<50\\ -1: & y>50\end{cases}$$

دقت : 98٪

ترفند هسته : چند جمله ای – درجه : 5 – بایاس ثابت : 2




**بخش دوم**

در این بخش از همان پایگاه داده USPS استفاده کردیم و مطابق پاورقی مربوط به این بخش عمل کردیم (یعنی تصاویر را بصورت grayscale لود کردیم و به بردار تبدیل کردیم). طبق تجربه، استفاده از هر سه ترفند هسته خطی، چند جمله ای و گاوسی، دقت پیش بینی های الگوریتم را به بالای 90٪ می رساند(برخلاف ترفند هسته سیگموید که همواره کمتر از 70٪ است)؛ اما بهترین هسته برای این مسئله خاص، هسته گاوسی با دقت 94.818٪ می باشد. همچنین طبق تجربه تغییر پارامتر های C و گاما باعث بدتر شدن دقت می شود.

مقایسه با شبکه عصبی: به نظر می رسد SVM در عین سادگی می تواند هم پای شبکه عصبی در دسته بندی اشیاء پیش بیاید. جالب اینکه بهترین دقت بدست آمده هنگام دسته بندی این پایگاه داده بوسیله شبکه عصبی، حتی مقداری کمتر از SVM بود (94.4%). این موضوع در کنار زمان اجرای نسبتا کمتر SVM و با درنظر گرفتن اینکه شما در SVM مجبور به امتحان کردن پارامترهای آزاد کمتری نسبت به شبکه عصبی هستید، SVM را به گزینه بسیار مناسبی برای دسته بندی اشیاء تبدیل می کند.

**بخش سوم**

در این بخش، مشابه بخش قبل، تصاویر پایگاه داده را لود کردیم. در هر پوشه مربوط به هر کدام از حروف یا ارقام، 300 تصویر وجود دارد که آنها را بصورت تصادفی در بخش داده های آزمایشی و آموزشی دسته بندی کردیم. مشاهده شد که تغییر اندازه دسته داده های آموزشی، تاثیر شگرفی روی دقت پیش بینی های الگوریتم ندارد و حتی اگر SVM را با 10 تصویر آموزش دهیم، دقت بیش از 80٪ باقی می ماند. همچنین برای تعداد بیش از 150 تصویر آموزشی، دقت از 92٪ کمتر نمی شود!
با تنظیم تعداد داده های آموزشی روی 200 تصویر، مشاهده کردیم که تغییر پارامتر های دیگر مانند C یا بایاس ثابت، از دقت الگوریتم می کاهد؛ بنابراین تنظیمات دیگر را تغییر ندادیم؛ اما به نظر می رسد که ترفند هسته چندجمله ای، بهترین پاسخ را برای این داده ها با درجه 3 و میانگین دقت 97٪ داشته باشد؛ هرچند نتیجه آزمایش هسته های گاوسی و خطی نیز، دقتی بیش از 93٪ داشت (البته مانند بخش قبل عملکرد هسته سیگموید بسیار ضعیف بود).

## پایان