بنام خدا

گزارش تمرین کامپیوتری سوم

هوش محاسباتی

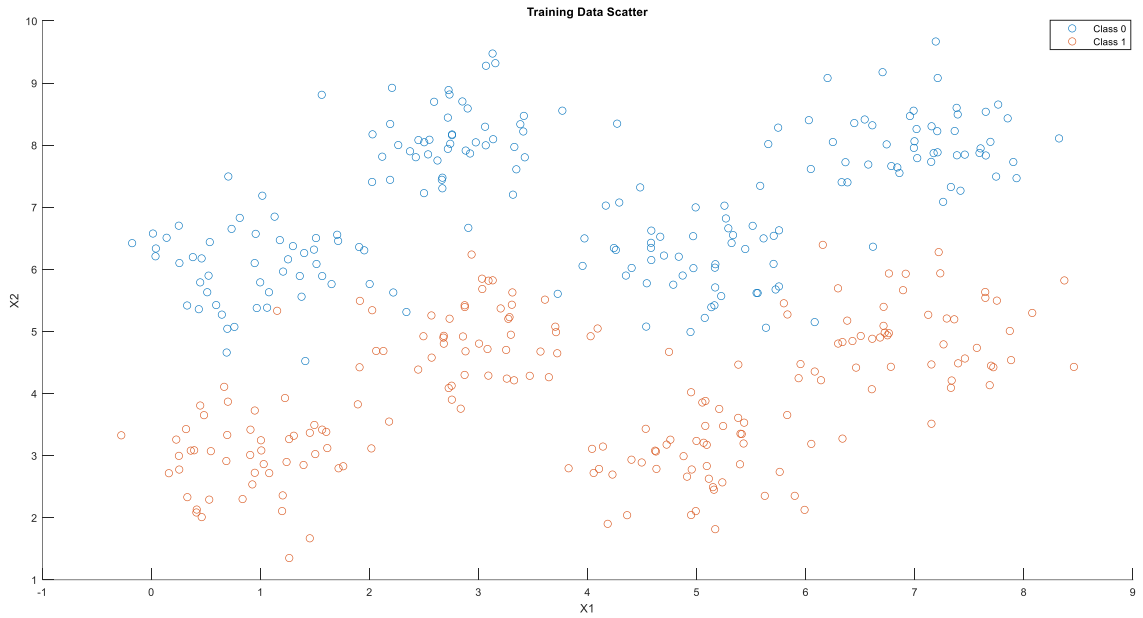
آرشام لؤلؤهری

۹۹۱۰۲۱۵۶

سوال ۱:

الف) (کد در سکشن Part A)

داده کلاس های صفر و یک را در دو ماتریس جداگانه میریزیم، و در یک نمودار رسم میکنیم:



ب) (کد در سکشن Part B)

به صورت رندوم، ۱۲۰ داده بعنوان داده های اعتبارسنجی، و مابقی بعنوان داده آموزشی تعیین میشوند

ج) از شبکه RBF با یک نورون خروجی استفاده میکنیم. در تابع newrb، علاوه بر داده های آموزشی و label های آنها، ترشولد خطای MSE برای توقف الگوریتم (که اینجا صفر گرفته شده تا کمترین خطای ممکن پیدا شود)، شعاع σ برای نورون های

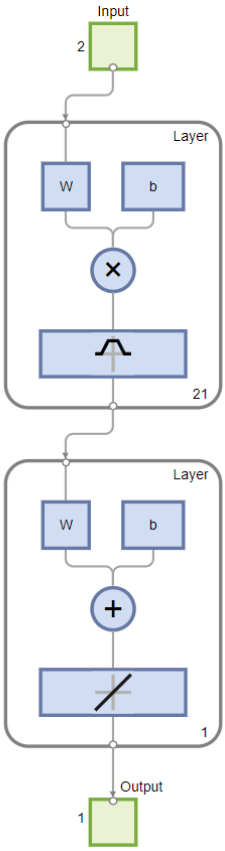
پنهان (با تابع فعالسازی گوسی)، ماکزیمم تعداد نورون های پنهان (که در کل باید کمتر از تعداد داده های آموزشی باشد تا شبکه RBF نرمال باشد)، و گام های افزایش تعداد نورون در هر یادگیری نیز بعنوان ورودی به تابع داده میشود.

برای تعیین بهترین تعداد نورون لایه پنهان و نیز شعاع σ برای این نورون ها، محدوده ای از تغییرات برای هردو کمیت در نظر گرفته و با دوتا حلقه for، هربار به ازای یک جفت مقدار برای این دو پارامتر، شبکه ای را با داده های آموزشی، آموزش داده، خروجی را برای داده های validation بدست آورده و میزان دقت شبکه در تعیین خروجی ها را (accuracy) محاسبه میکنیم. هربار که بهترین دقت خروجی به ازای داده های validation بدست آمد، تعداد نورون (n) و شعاع σ مربوط به آن مرحله را با پسوند نام best ذخیره میکنیم. در نهایت با بهترین تعداد نورون و شعاع، شبکه ای ساخته و خروجی داده های اعتبارسنجی را تعیین میکنیم.

پس از طی این فرایند، به ۲۱ نورون پنهان و شعاع $\sigma = 1$ میرسیم، که دقت 97.5% حاصل کرده است:

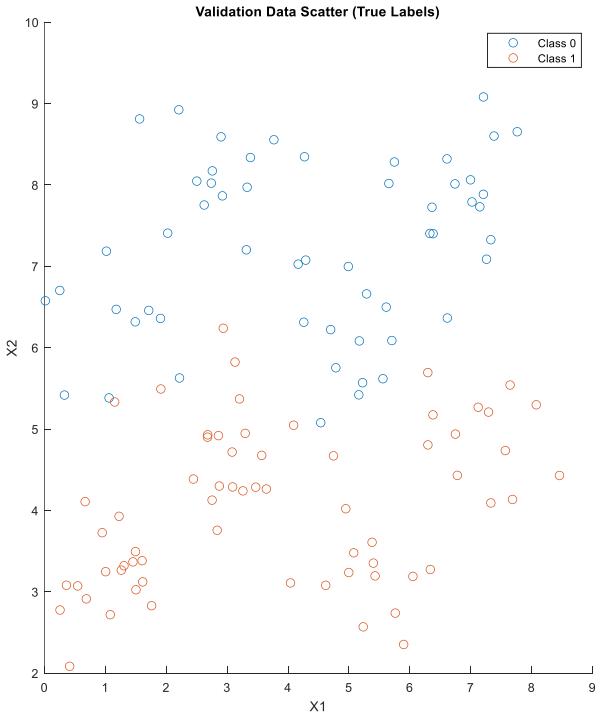
```
Best accuracy: 97.5%
Best number of neurons: 21
Best sigma: 1
```

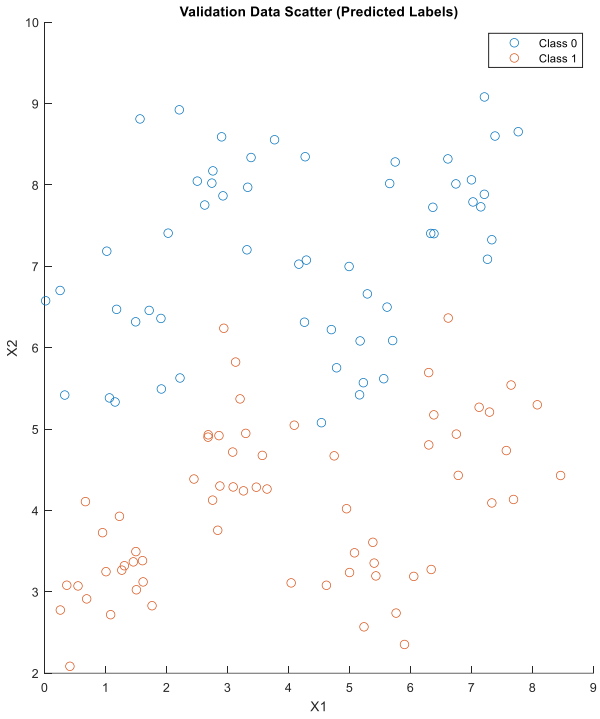
نمای کلی شبکه RBF طراحی شده:



لایه پنهان با تابع فعالسازی گوسی، و لایه خروجی شامل تک نورون خطی است.

این بار تنها داده های validation را، یکبار با لیبل های اصلی و یکبار با لیبل های بدست آمده از شبکه، کلاس بندی کرده و رسم میکنیم:





مشاهده میشود شبکه با دقت خوبی کلاس بندی را به درستی انجام داده است و دقت 97.5% در این دو نمودار مشهود است.

ضمنا خروجی شبکه، باینری نیست. بلکه به صورت یک عدد حقیقی است که اگر به 0 نزدیکتر باشد، به مقدار باینری 0، و اگر به 1 نزدیکتر باشد، به مقدار باینری 1 نظیر میشود.

سوال ۲:

الف) محاسبات الگوریتم k-means در تابع kmeansCal آمده است. این تابع با گرفتن دیتا، مقدار K و مراکز خوشه ی اولیه، مراکز خوشه ی نهایی (clustMat) و بردار شماره خوشه ی مربوط به هریک از نقاط داده (clust_ind) را در خروجی تحویل میدهد.

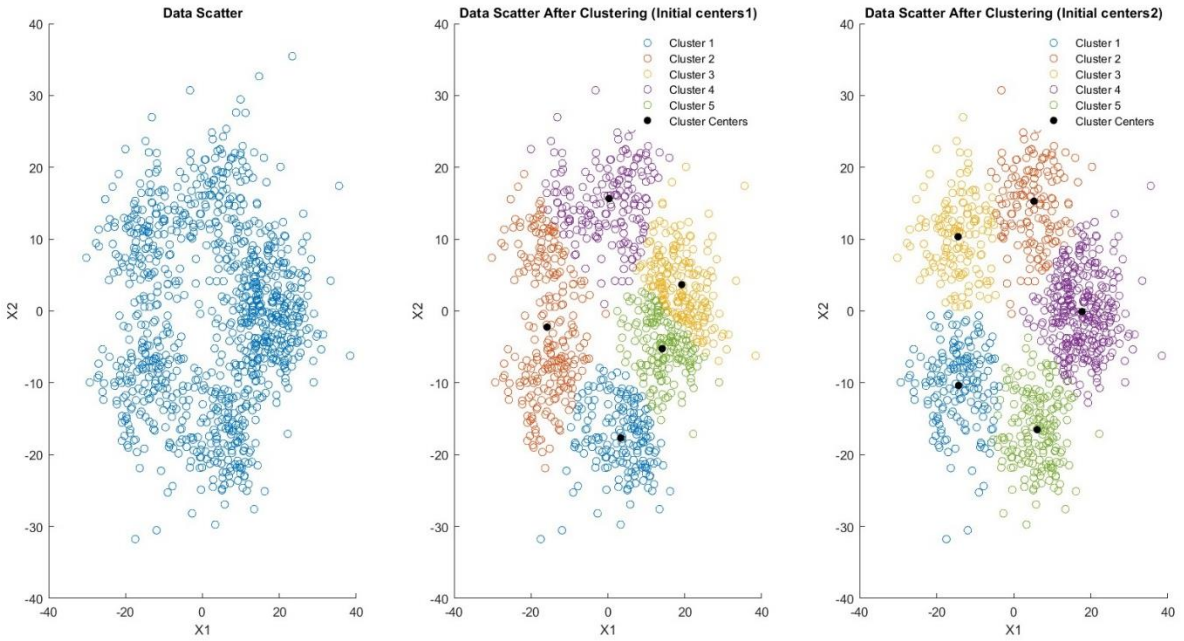
ب) (کد سکشن Part B)

تعداد خوشه را تعیین کرده، و پنج نقطه از نقاط داده را به صورت تصادفی انتخاب میکنیم تا بعنوان مراکز اولیه به الگوریتم kmeansCal بدهیم. نقاط اولیه را از خود نقاط داده انتخاب میکنیم تا الگوریتم حتما به ۵ خوشه منتهی شود.

نقاط اولیه، مطابق خواسته ی سوال، دوبار به صورت رندوم انتخاب میشوند و خروجی نیز دوبار با شروط اولیه ی متفاوت محاسبه میشود.

سپس خروجی های تابع kmeansCal، به تابع دیگری به نام apply_kmeans داده میشود تا رسم scatter ها بر اساس تعداد خوشه ی موردنظر و شروط اولیه تعیین شده، انجام شود.

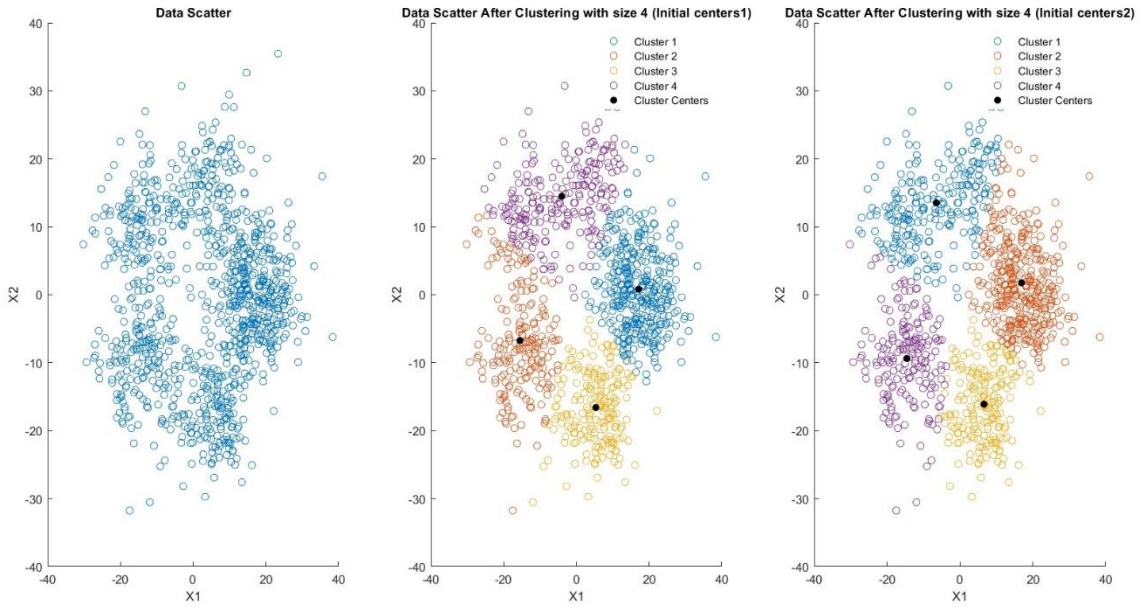
نتایج این بخش به ازای دو شرط اولیه مختلف، به صورت زیر است:



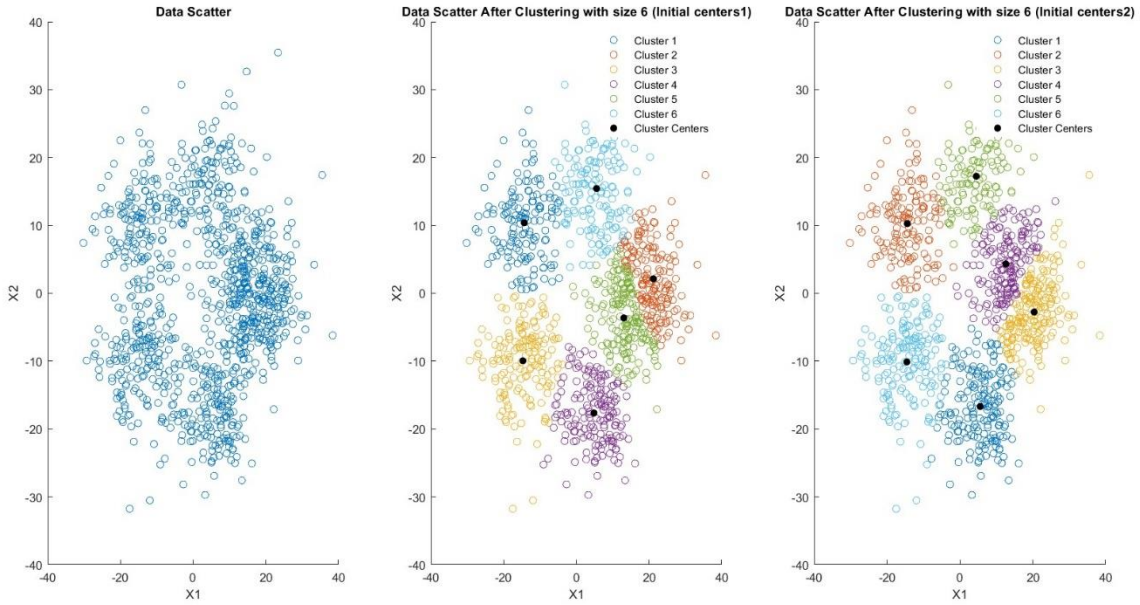
که در هردو نمودار، نقاط به صورتی منطقی و خوب خوشه بندی شده اند و نقاط نزدیک به هم، با تعیین یک مرکز منطقی در میان آن داده ها، در یک خوشه قرار گرفته اند. اما با تغییر مراکز اولیه، خوشه بندی داده ها اندکی تغییر میکند و برخی داده ها از یک خوشه، به دو خوشه متفاوت تبدیل میشوند و یا برعکس.

ج) (کد سکشن Part C)

به ازای clusterNum=4, 6 الگوریتم بخش قبل را تکرار میکنیم. به ازای k=4:



و به ازای k=6:

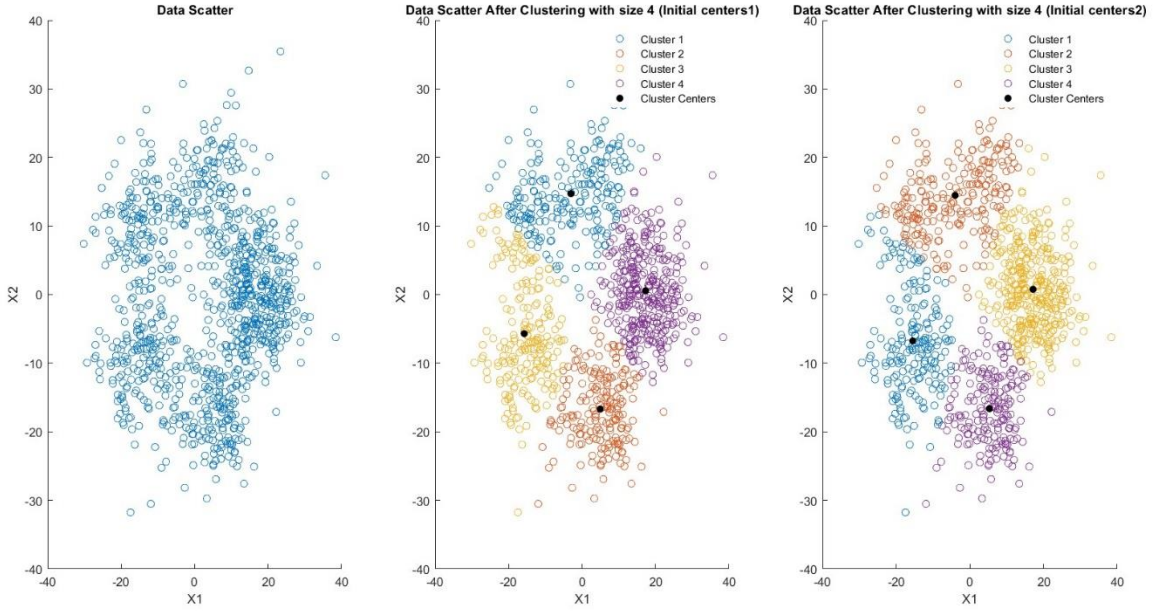


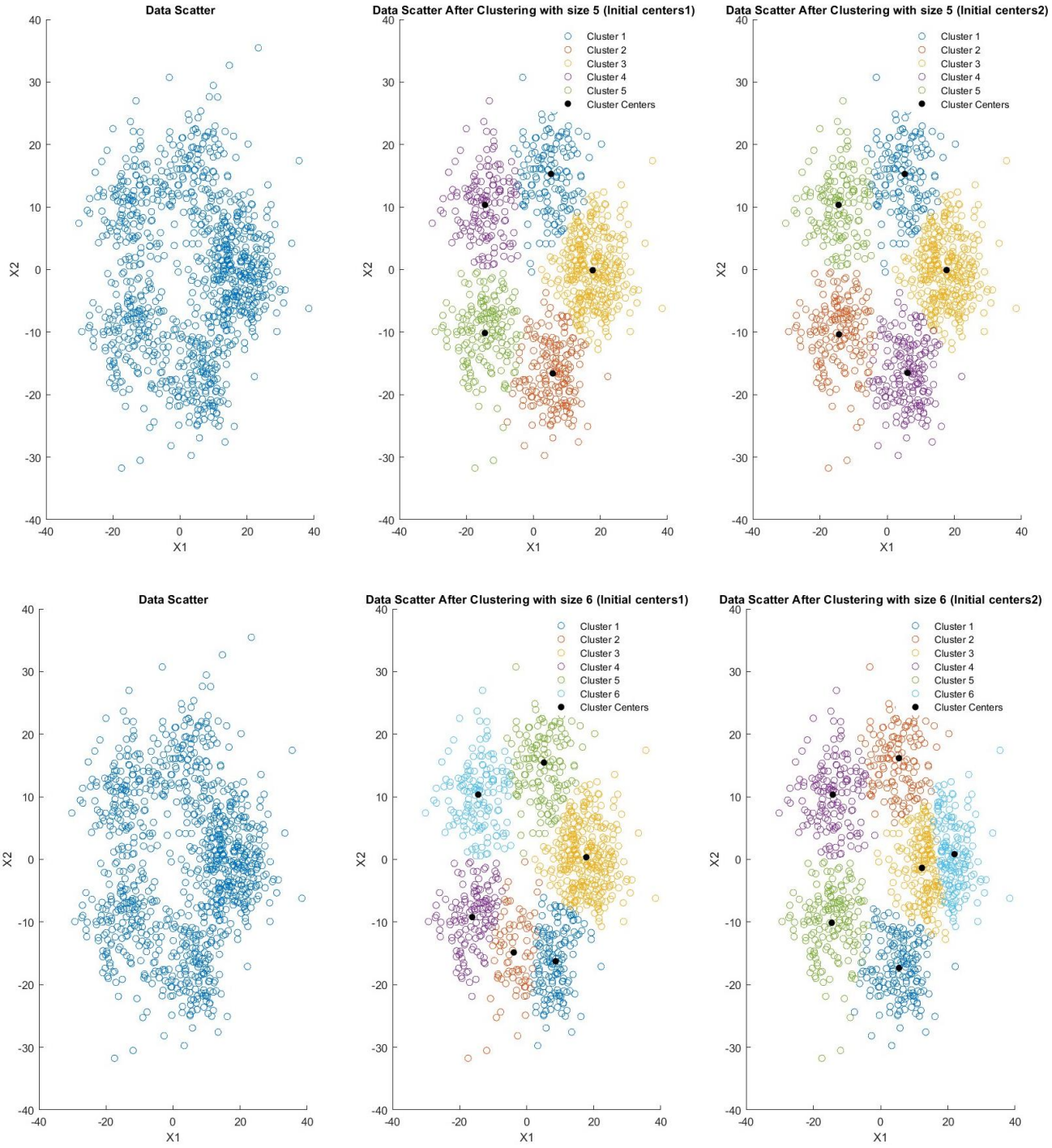
مجددا خوشه بندی به درستی انجام شده و با تغییر شرایط اولیه، مرکز خوشه ها و خود خوشه بندی داده ها اندکی تغییر میکند.

د) (کد سکشن Part D)

اینبار از تابع kmeans استفاده کرده و مجددا نتایج را با تابع apply_kmeans نمایش میدهیم. در داخل تابع kmeans از ورودی sample استفاده میکنیم تا نقاط اولیه به صورت رندوم از داده ورودی برداشته شوند.

نتایج به ازای k=4:





کلیت خوشه بندی مشابه تابع دست نویس است و مراکز خوشه ها تقریبا در همان حوالی قرار گرفته اند. مجددا میبینیم که تغییر تصادفی در شرایط اولیه، اندکی خوشه بندی را تغییر میدهد.

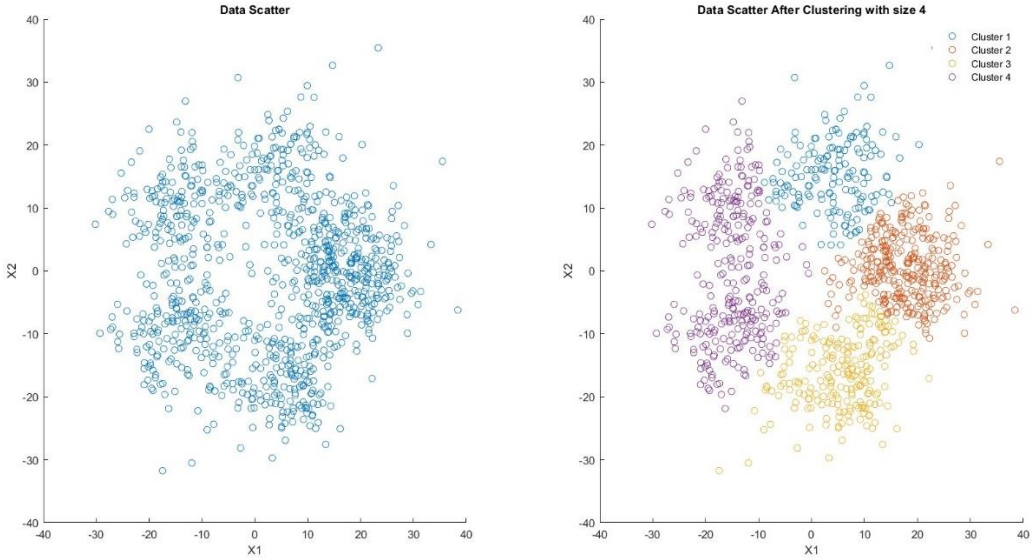
ه) (کد سکشن Part E)

از روش hierarchical استفاده میکنیم. این تکنیک دو شیوه‌ی Agglomerative و divisive دارد که در این کد از روش اول استفاده شده است.

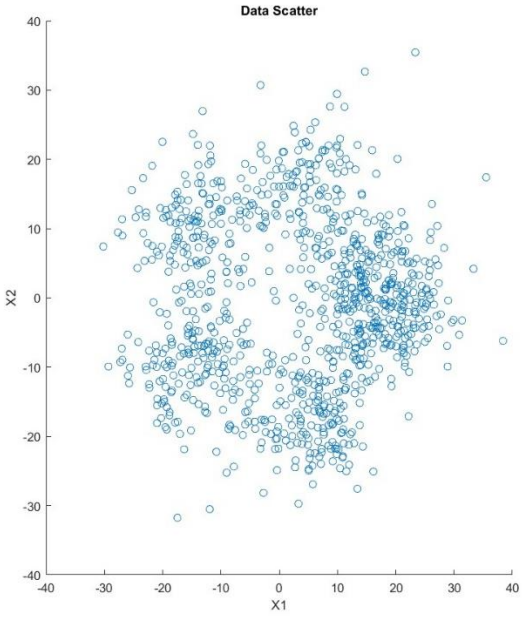
در این روش، در مرحله اول، هر نقطه از دیتا، یک خوشه ی جداگانه حساب میشود. سپس مرحله به مرحله، دو خوشه ای که کمترین فاصله را از یکدیگر داشته باشند، با یکدیگر ادغام میشوند و در نتیجه تعداد خوشه ها، یکی یکی کاهش می یابد تا اینکه به تعداد خوشه مورد نظر برسیم. فاصله بین خوشه ها میتواند با معیارهای مختلفی سنجیده شود که linkage نام دارد. برای مثال در centroid linkage، فاصله اقلیدسی مرکز دو خوشه (میانگین داده های خوشه) بعنوان فاصله حساب میشود. در کد، از ward استفاده شده که مختص زمانی است که بخواهیم واریانس فواصل بین خوشه ها، مینیمال شود.

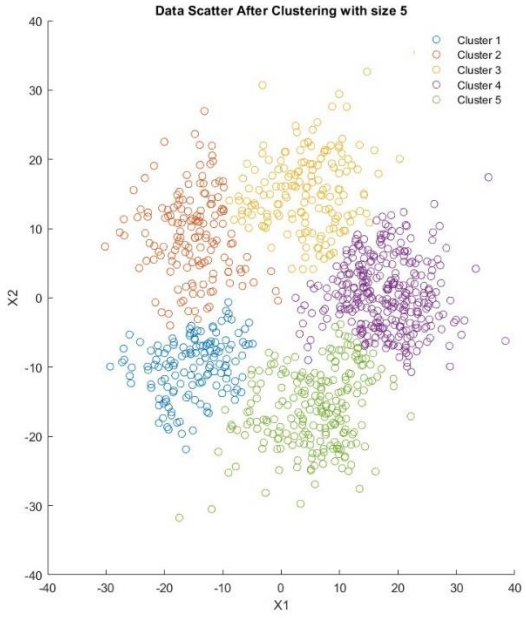
مجددا به ازای تعداد خوشه های مختلف، خوشه بندی را انجام میدهیم:

به ازای k=4:

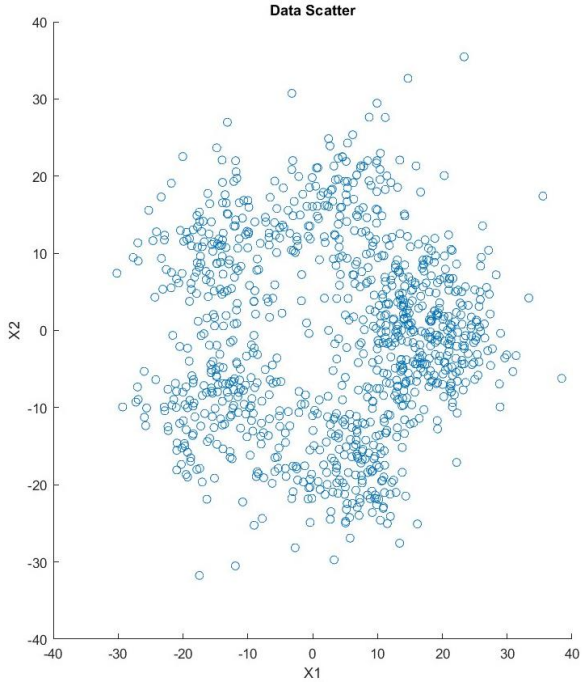


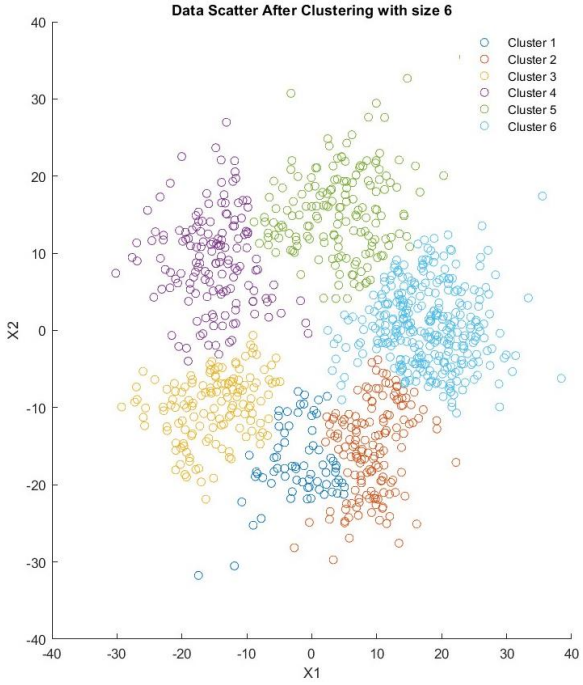
به ازای k=5:





به ازای k=6:





مجددا داده های هر خوشه، به یکدیگر نزدیکتر بوده و خوشه بندی منطقی ای به ازای هر k انجام شده است. ضمنا تاثیر مثبت افزایش تعداد خوشه را نیز در تغییر از k=4 به k=5 میبینیم، که خوشه ی بسیار بزرگ بنفش رنگ، به دو خوشه جدا و منطقی تر تجزیه شده است.