روش‌های استخراج ویژگی و خوشه‌بندی برای افتراق سه‌گانه

|  |
| --- |
| تهیه کننده گزارش: سید مصطفی غضنفری |
| شماره گزارش: |
| تاریخ تهیه گزارش: 14/07/1403 |
| تعداد صفحات: صفحه |
| بر اساس داده‌های : |
| نام فایل کد مربوطه : |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| جدول Revision History | | | | |
| Rev | **شرح تغییرات** | **تهیه کننده گزارش** | **تأییدکنندگان** | **تاریخ** |
| 1 |  |  |  |  |
| 2 |  |  |  |  |
| 3 |  |  |  |  |
| 4 |  |  |  |  |
| 5 |  |  |  |  |
| 6 |  |  |  |  |
| 7 |  |  |  |  |
| 8 |  |  |  |  |
| 9 |  |  |  |  |
| 10 |  |  |  |  |

Table of Contents

[1 مقدمه 3](#_Toc180249987)

[2 استخراج ویژگی 3](#_Toc180249988)

[3 روش‌های مبتنی بر هوش مصنوعی برای استخراج ویژگی 3](#_Toc180249989)

[3.1 Autoencoder 3](#_Toc180249990)

[3.2 Self-supervised learning 5](#_Toc180249991)

[3.2.1 A Simple Framework for Contrastive Learning of Visual Representation (SimCLR) 5](#_Toc180249992)

[3.2.2 Simple Siamese Representation Learning(simsiam) 7](#_Toc180249993)

[4 خوشه‌بندی 8](#_Toc180249994)

[4.1 ویژگی‌های الگوریتم‌های خوشه‌بندی 8](#_Toc180249995)

[4.1.1 Scalability 8](#_Toc180249996)

[4.1.2 Flat 8](#_Toc180249997)

[4.1.3 Inductive 8](#_Toc180249998)

[4.2 الگوریتم‌های خوشه‌بندی 8](#_Toc180249999)

[4.2.1 K\_means 9](#_Toc180250000)

[4.2.2 Spectral clustering 12](#_Toc180250001)

[4.2.3 Hierarchical clustering 14](#_Toc180250002)

[4.2.4 Gaussian mixture 20](#_Toc180250003)

[4.3 پیاده ‌سازی و مقایسه روش‌ها روی داده‌های ما 26](#_Toc180250004)

[5 منابع 28](#_Toc180250005)

# مقدمه

در این گزارش به بررسی روند انجام افتراق سه‌گانه می‌پردازیم. این کار پس از جدا کردن گلبول در تصاویر ، در دو مرحله استخراج ویژگی و خوشه‌بندی انجام می‌پذیرد.

# استخراج ویژگی

استخراج ویژگی برای کار ما به دو روش می‌تواند انجام بگیرد. یکی اینکه با نوجه به مقالات ویژگی‌هایی را که مناسب این افتراق است استخراج کرده و بعد خوشه‌بندی با آن ها صورت گیرد. این کار معمولا یک محاسبه ریاضی ساده است.

روش دوم استخراج ویژگی با شبکه‌های عصبی است. در این حالت هدف استخراج مهم‌ترین ویژگی‌های سازنده تصویر است. این ویژگی‌ها یک سری عدد هستند که لزوما ما معنی آن ها را نمی‌دانیم و توسط شبکه به دست آمده‌اند. البته ممکن است شبکه هم همان ویژگی‌های معناداری که ما هم تشخیص داده ایم را استخراج کند. برای این کار شبکه‌ها و روش های مختلف وجود دارد.

# روش‌های مبتنی بر هوش مصنوعی برای استخراج ویژگی

در این روش ها باید یک شبکه برای استخراج ویژگی آموزش ببیند. مشکلی که وجود دارد این است که داده‌های ما لیبل ندارند. می‌دانیم در شبکه‌های عصبی برای آموزش به داده لیبل زده نیاز داریم. با توجه به این مشکل راه‌های مختلفی وجود دارد. یک راه این است که از شبکه‌های از پیش آموزش داده شده برای استخراج ویژگی استفاده کنیم. راه دیگر استفاده از شبکه‌هایی است که نیاز به لیبل قطعی داده‌های ما ندارند! این روش ها به طرق مختلف task جایگزینی ایجاد می‌کنند و پس از آن آموزش صورت گرفته و بعد از این شبکه برای استخراج ویژگی استفاده می‌شود. در این گزارش به چند مورد از این روش ها می‌پردازیم.

## Autoencoder

خودرمزگذار (Autoencoder) نوعی معماری شبکه عصبی است که برای فشرده‌سازی داده‌های ورودی به ویژگی‌های اصلی آن طراحی شده و سپس بازسازی ورودی اصلی از این ویژگی‌های فشرده ‌شده را انجام می‌دهد.

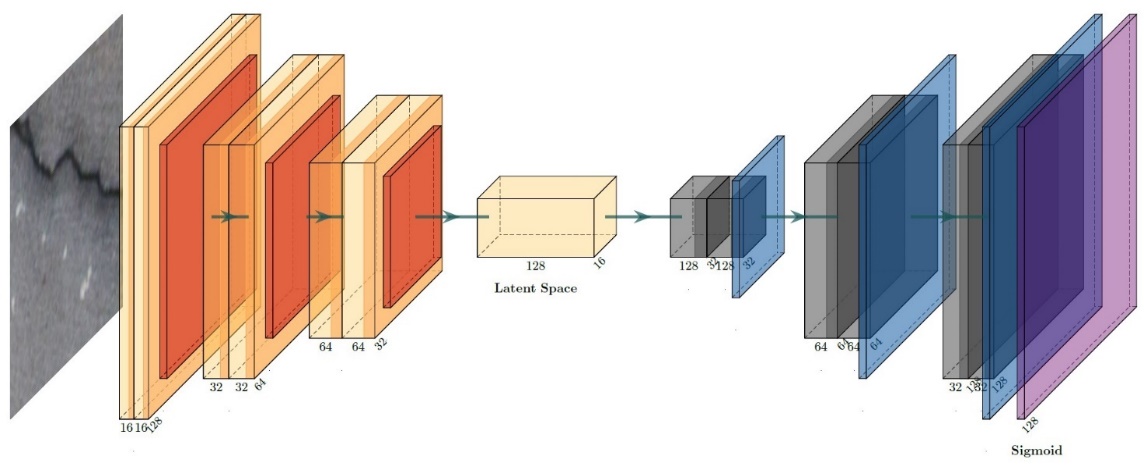
خودرمزگذارها آموزش می‌بینند تا متغیرهای پنهان داده‌های ورودی را کشف کنند ، متغیرهای پنهانی که علیرغم اینکه به طور مستقیم قابل مشاهده نیستند، به طور اساسی نحوه توزیع داده‌ها را مشخص می‌کنند. در طول فرآیند آموزش، خودرمزگذار یاد می‌گیرد که کدام متغیرهای پنهان می‌توانند برای بازسازی دقیق‌تر داده‌های اصلی مورد استفاده قرار گیرند ، این ویژگی ها مهم‌ترین اطلاعات موجود در تصویر است.

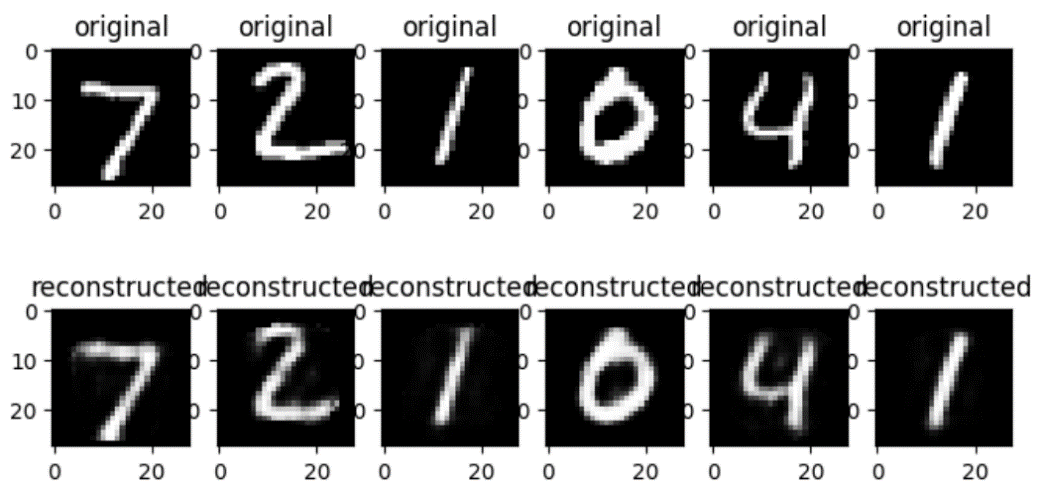
شبکه‌های خودرمزگذار جزو شبکه‌های رمزگذار-رمزگشا (encoder-decoder) هستند. شبکه‌های رمزگذار-رمزگشا، که در آن یک شبکه رمزگذار ویژگی‌های کلیدی داده‌های ورودی را استخراج می‌کند و شبکه رمزگشا این داده‌های استخراج‌ شده را به‌عنوان ورودی خود می‌گیرد، در انواع مختلفی از مدل‌های یادگیری عمیق استفاده می‌شوند. در اکثر کاربردهای این شبکه ها خروجی شبکه عصبی با ورودی آن متفاوت است. برای مثال، در U-Net، شبکه رمزگذار ویژگی‌ها را از تصویر ورودی استخراج می‌کند تا طبقه‌بندی پیکسل‌های مختلف را تعیین کند؛ با استفاده از نقشه ویژگی ، شبکه رمزگشا سپس لیبل هر پیکسل را معلوم می‌کند. این مدل‌ها از طریق یادگیری نظارت‌شده (supervised learning) آموزش می‌بینند.

خودرمزگذارها از این جهت که مثل بقیه task ها به صورت مستقیم لیبل ندارند جزو یادگیری بدون نظارت (unsupervised learning) هستند اما برخلاف مثال‌های یادگیری بدون نظارت و شبیه به مدل‌های یادگیری نظارت‌ شده خودرمزگذارها یک لیبل قراردادی دارند تا خروجی خود را در برابر آن اندازه‌گیری کنند به همین دلیل جزو دسته self-supervised learning قرار می‌گیرند.

این شبکه‌ها از سه بخش encoder ، bottleneck و decoder تشکیل شده اند. Encoder وظیفه دارد ویژگی ها را از ورودی استخراج کند. این قسمت (در حوزه تصویر) یک شبکه کانولوشنی است که به مرور ابعاد ویژگی‌ها را کمتر کرده و ویژگی‌های غنی‌تر را استخراج می‌کند. لایه اخر که بین encoder و decoder قرار دارد همان bottleneck است. بعد هم decoder قرار دارد که قرار است از این ویژگی‌های استخراج شده تصویر را بازسازی کند. بنابراین در این شبکه‌ها لیبل قرار دادی ما خود تصویر است.

یکی از مزایای اصلی استفاده از خودرمزگذارها نسبت به سایر تکنیک‌های کاهش ابعاد مانند PCA این است که خودرمزگذارها می‌توانند همبستگی‌های پیچیده غیرخطی را نیز کشف کنند. به همین دلیل، تابع‌های فعال‌سازی مورد استفاده در خودرمزگذارها معمولاً تابع‌های غیرخطی هستند.





## Self-supervised learning

همان طور که در بخش autoencoder ها توضیح داده شد self-supervised learning به نوعی یادگیری گفته می‌شود که در task اصلی ، داده لیبل‌دار وجود نداشته اما ما با ایجاد یک لیبل قراردادی شبکه را آموزش می‌دهیم. در این نوع روش انواع task های مختلف تعریف می‌شود تا شبکه با آن آموزش ببیند و سپس ویژگی‌های مهم که توسط این شبکه آموزش دیده به دست می‌آید مورد استفاده قرار می‌گیرد.

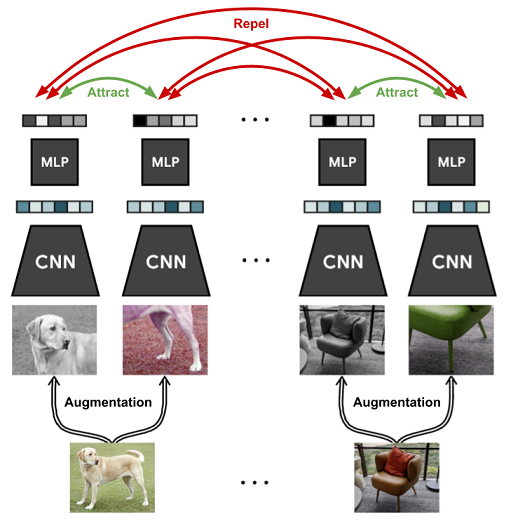
### A Simple Framework for Contrastive Learning of Visual Representation (SimCLR)

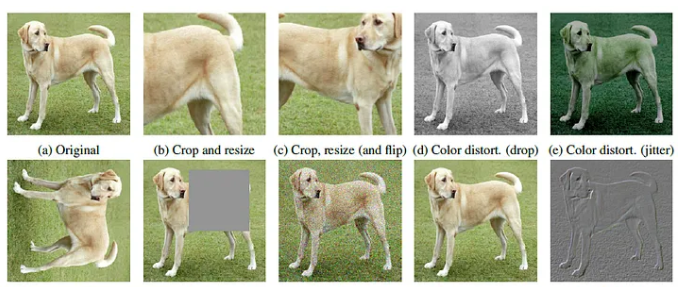
ایده اصلی این روش این است که augment های مختلف یک تصویر مشخص باید ویژگی‌های یکسان و نمایش یکسان داشته باشند.

1. دو دسته تصویر داریم که به اندازه batch size ما هستند. انواع augment های مختلف مانند تبدیلات رنگی و هندسی روی تصاویر این دو دسته اعمال می‌شود. چون اعمال augmentation احتمالی است این دو دسته یکسان نخواهند بود.
2. تصاویر دو دسته به یک شبکه backbone که شبکه CNN می‌باشد داده می‌شوند تا استخراج ویژگی انجام شود.
3. سپس ویژگی ها به یک شبکه MLP داده می‌شود تا آماده بررسی loss کرده و همچنین ویژگی ها را ارزیابی کرده و بهترین ویژگی ها را انتخاب کند.
4. در انتها loss حساب می‌شود. loss به این صورت عمل می‌کند که اگر تصاویر augment شده از یک جسم باشند باید فاصله ویژگی ها از هم کم باشد و اگر تصاویر augment شده از یک جسم نباشند باید loss زیاد شود.

آموزش به همین نحو انجام شده و وزن ها طوری تغییر می‌کنند که ویژگی‌های مهم را استخراج کنند تا loss برای تصاویر augment شده از یک جسم کمینه و در غیر این صورت زیاد شود. نکته مهم در این روش استفاده از batch size های بزرگ است چون باعث می‌شود تعداد negative pair ها زیاد شود. به عنوان مثال اگر batch size برابر با n باشد در محاسبه loss ما n زوج مثبت و

n\*(n-1) زوج منفی داریم که این تعداد زیاد به آموزش ما کمک خواهد کرد.

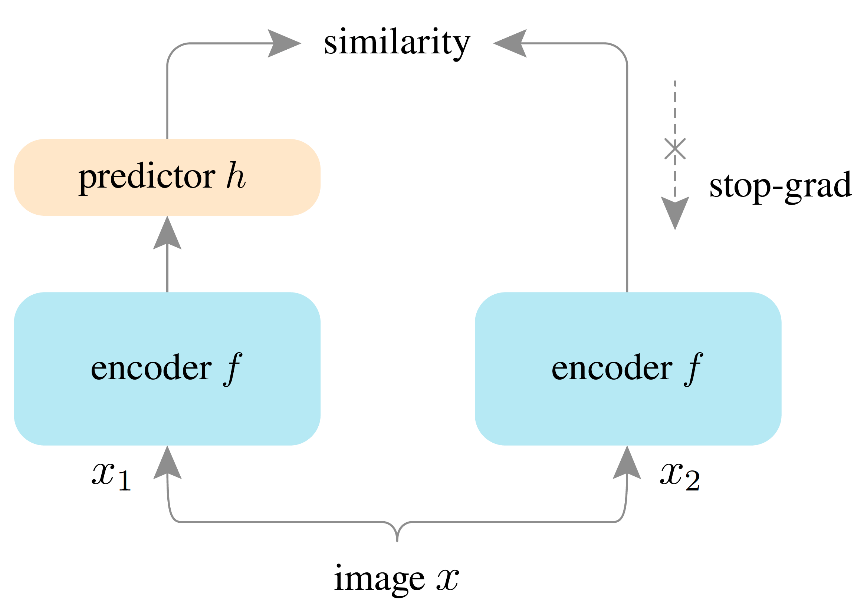




### Simple Siamese Representation Learning(simsiam)

ایده اصلی این روش هم این است که augment های مختلف یک تصویر مشخص باید ویژگی‌های یکسان و نمایش یکسان داشته باشند.

1. دو دسته تصویر داریم که به اندازه batch size ما هستند. انواع augment های مختلف مانند تبدیلات رنگی و هندسی روی تصاویر این دو دسته اعمال می‌شود. چون اعمال augmentation احتمالی است این دو دسته یکسان نیستند.
2. تصاویر دو دسته به یک شبکه backbone که شبکه CNN می‌باشد داده می‌شوند تا استخراج ویژگی انجام شود.
3. سپس ویژگی ها به یک شبکه MLP داده می‌شود تا بهترین ویژگی ها را انتخاب کند.
4. سپس یکی از دو دسته خروجی به یک شبکه MLP به نام predictor داده می‌شود تا این ویژگی‌ها به ویژگی‌های دسته دیگر نگاشت شوند. بعد همین کار برای دسته دوم هم انجام می‌شود.
5. در نهایت با یک معیار شباهت دو کاری که در مرحله 4 انجام شده بررسی شده و loss محاسبه می‌شود.
6. توجه شود که در این حالت فقط روی زوج‌های مثبت loss محاسبه می‌شود و به خاطر همین به batch size بزرگ نیازی نیست.



# خوشه‌بندی

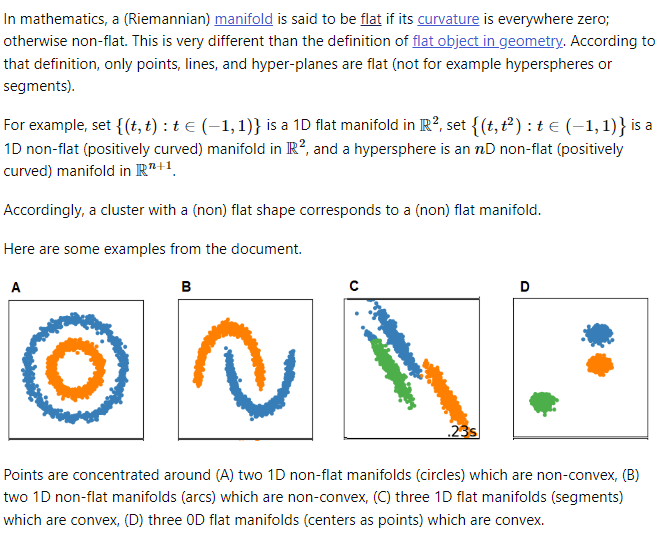
## ویژگی‌های الگوریتم‌های خوشه‌بندی

### Scalability

یعنی پارامتر مشخص شده اگر x برابر شود زمان اجرای الگوریتم هم تقریبا x برابر شود. مثلا نسبت به تعداد داده و یا تعداد خوشه یعنی اگر تعداد داده‌ها یا خوشه x برابر شد زمان اجرای الگوریتم هم x برابر شود.

### Flat

فضا و یا سطحی را flat میگویند که curvature یا همان خمیدگی آن در همه جا صفر باشد.



### Inductive

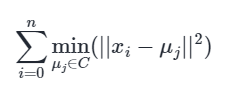
inductive که در مقابل transductive قرار می‌گیرد به این معناست که آیا الگوریتم مدلی می‌سازد تا بتوان بعدا آن را روی نمونه‌های جدید اعمال کرد یا نه.

## الگوریتم‌های خوشه‌بندی

نکته مهم در بررسی الگوریتم‌های مختلف خوشه‌بندی این است که الگوریتم‌هایی که پارامتر ورودی تعداد خوشه ندارند و خودشان بر اساس معیاری که دارند داده را به n خوشه تقسیم می‌کنند چون برای کار ما مناسب نیستند در این گزارش بررسی نشدند.

### K\_means

در این الگوریتم می‌خواهیم مجموع فاصله نقاط هر خوشه تا مرکز خوشه را کمینه کنیم که معادل است با :

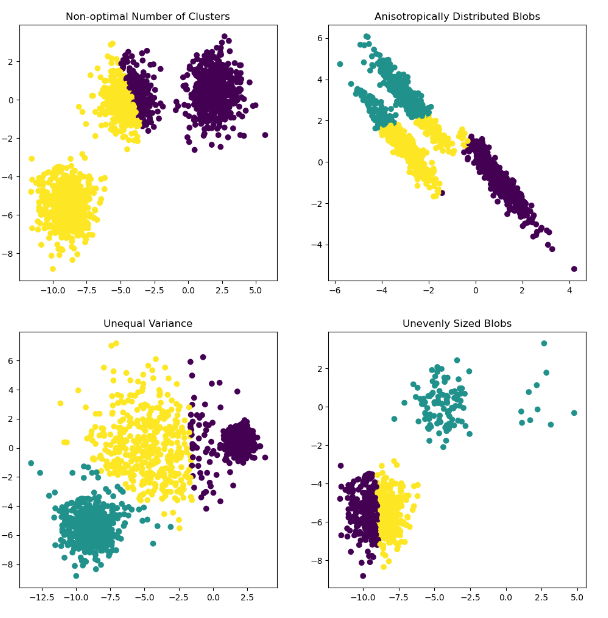


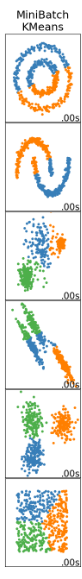
این روش مشکلاتی متعددی را دارد که عبارت اند از :

1. در این روش فرض شده که حتما همه خوشه ها convex هستند. همچنین روی شکل‌های elongated ( کشیده و باریک) این روش خوب جواب نمی‌دهد.



1. اگر تعداد ویژگی ها زیاد باشد چون معیار این روش بر اساس فاصله اقلیدسی کار می کند مشکل curse of dimensionality رخ می‌دهد و تاثیر ویژگی ها از بین می‌رود بنابراین نیاز است از روش‌های کاهش ابعاد استفاده شود.





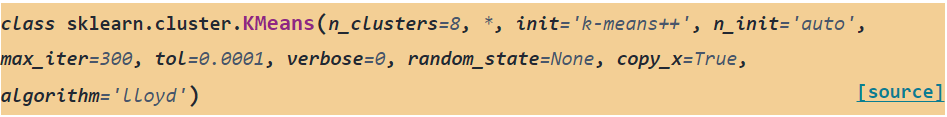
این الگوریتم به این صورت عمل می‌کند که ابتدا تعدادی مرکز انتخاب می‌کند سپس بررسی می‌کند که هر نقطه به کدام یک از این مراکز نزدیک تر استت و آنرا متعلق به آن خوشه می‌کند. وقتی برای همه نقاط این کار انجام شد ، با توجه به نقاط هر خوشه ، مرکز جدید محاسبه شده و دوباره همین مراحل تکرار می‌شود....( توقف‌های زیادی را برای این الگوریتم میتوان در نظر گرفت مثل تعداد مراحل انجام یا اینکه مراکز خوشه ها از یک حد مشخص کمتر تغییر کنند و ...)

این الگوریتم بسیار وابسته به انتخاب مراکز اولیه است بنابراین به جای رندم انتخاب کردن ، الگوریتم‌های جایگزینی مثل kmeans++ به وجود آمدند.

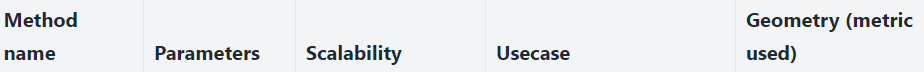
در این الگوریتم تعداد خوشه را خودمان باید معلوم کنیم. یک تکنیک مورد استفاده در این روش و در کل در خوشه بندی استفاده از وزن‌دهی به نقاط است. یعنی بعضی نقاط اهمیت بیشتری دارند پس وزن بیشتری خواهند داشت. انگار که آن نقاط تکثیر شده اند.

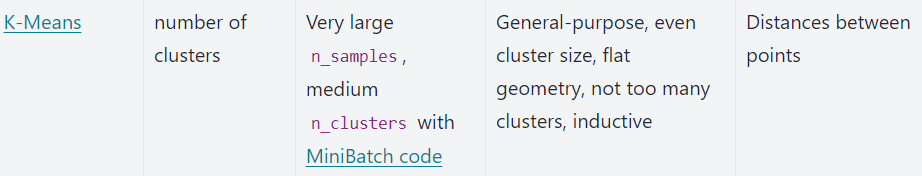
برای سریع تر شدن این الگوریتم روش‌های مختلفی ارائه شده اند که یکی از آن‌ها mini batch kmeans است.

#### ورودی‌های الگوریتم:



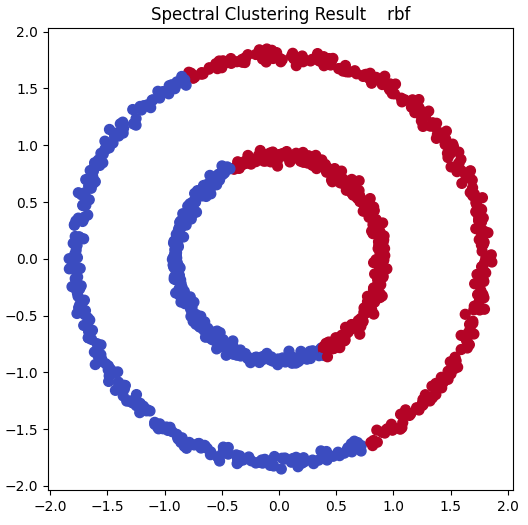
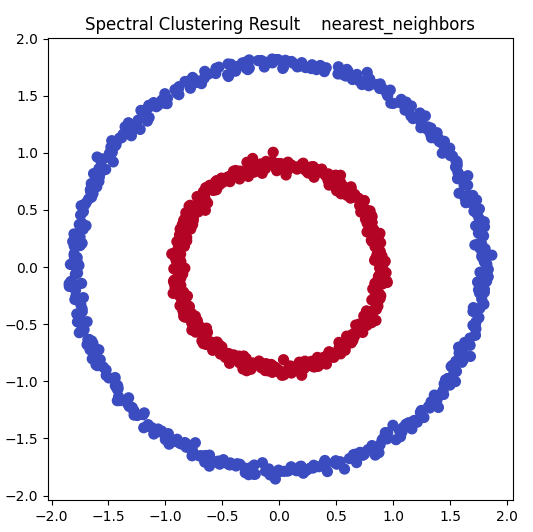
توضیحات کامل در [3] آمده است.

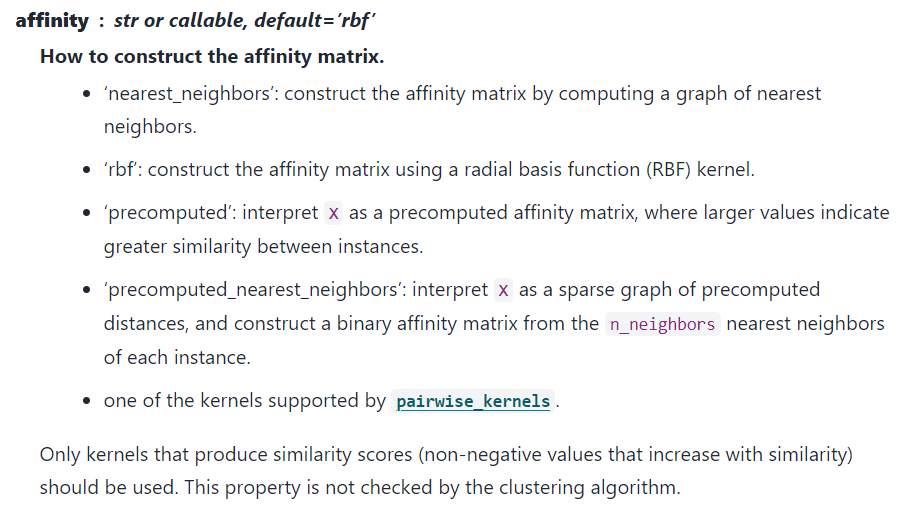




### Spectral clustering

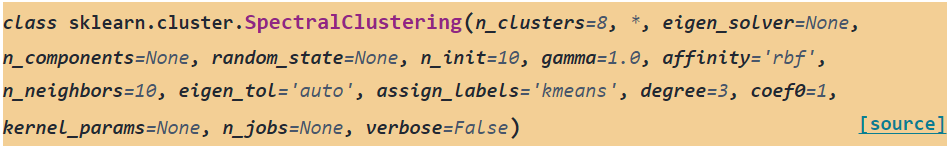
این روش مبتنی بر گراف است و به این صورت عمل می‌کند که هر دیتا از مجموعه دیتا ما به صورت یک راس گراف در نظر گرفته می‌شود. سپس یک ماتریس شباهت (affinity) بر اساس فاصله تشکیل می‌شود. معیار‌های فاصله متفاوتی را می‌توان در نظر گرفت که کاملا موثر در الگوریتم و نتیجه آن خواهد بود. کتابخانه sklearn گزینه‌های متفاوتی را در اختیار ما می‌گذارد.



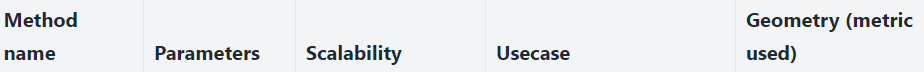


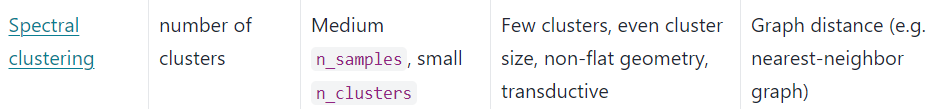
پس از این مرحله Laplacian این ماتریس محاسبه می‌شود. حال مقدار ویژه و بردار ویژه‌های این ماتریس محاسبه می‌شود. مقادیر ویژه از کوچک به بزرگ مرتب می‌شوند. هر چه مقدار ویژه کوچک تر باشد به این معناست که این بردار ویژه کمتر در ساخت آن فضا اهمیت دارد. چون این جا معیار ما فاصله و هدف فهمیدن این است که در چه معیاری نقاط به هم نزدیک هستند اتفاقا بردار ویژه با مقادیر ویژه کوچک تر مورد استفاده ما خواهند بود. پس از انتخاب شدن تعداد دلخواهی از بردار ویژه ها ، حالا روی این تعداد محدود الگوریتم k\_means اجرا می‌شود.

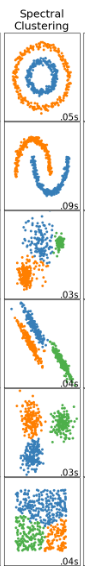
#### ورودی‌های الگوریتم:



توضیحات کامل در [4] آمده است.





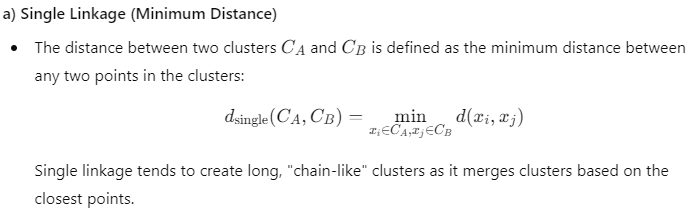


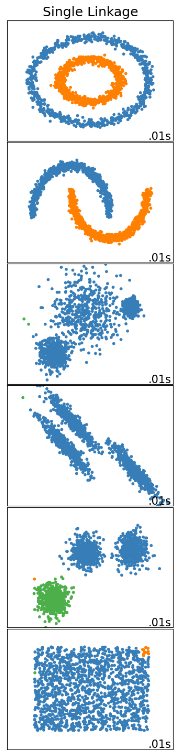
### Hierarchical clustering

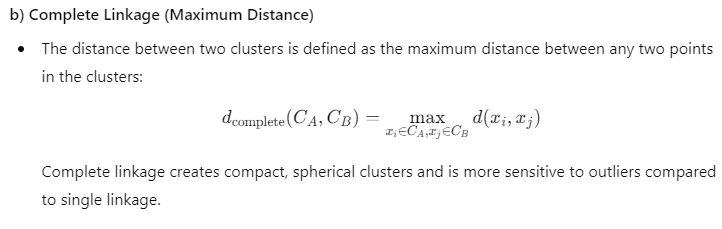
در این الگوریتم‌ها خوشه‌های نهایی با تقسیم یا به هم پیوستن سایر خوشه‌ها ایجاد می‌شود. می‌توان این نوع الگوریتم ها را یک درخت در نظر گرفت که در راس این درخت یک خوشه است که همه داده‌ها در آن قرار دارند و در انتهای درخت به تعداد داده‌ها خوشه داریم یعنی هر داده یک خوشه تشکیل داده است.

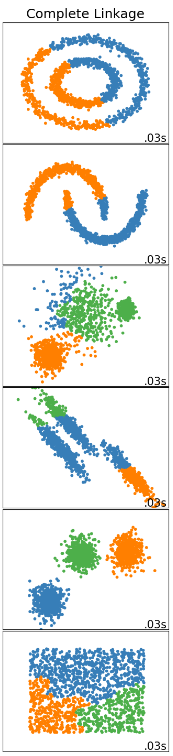
Agglomerative Clustering روش‌هایی از این دست هستند که در ابتدا هر داده یک خوشه بوده و به تدریج با معیار‌هایی که داریم این خوشه ها با هم ترکیب می‌شوند. معیار‌های ترکیب شدن (linkage) متفاوتی برای این روش وجود دارد.

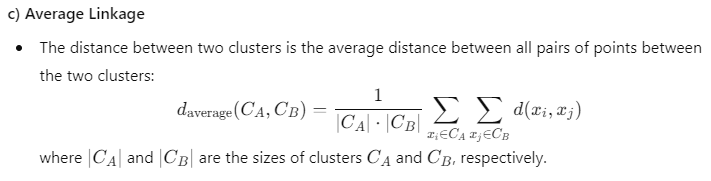
الگوریتم به این صورت عمل می‌کند که با n خوشه آغاز می‌کند ، ابتدا فاصله بین نقاط با یک معیار فاصله (معمولا فاصله اقلیدسی) محاسبه می‌شود. سپس با معیاری که انتخاب شده دو خوشه که بهترین معیار را دارند با هم ترکیب شده و این کار را تا رسیدن به تعداد خوشه نهایی که انتخاب ما بوده ادامه داده می‌شود.

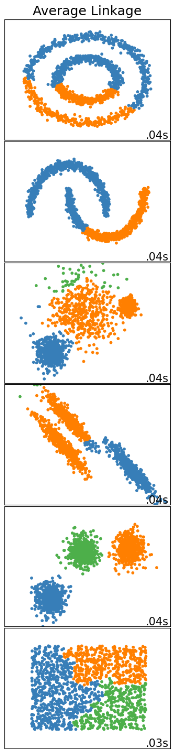


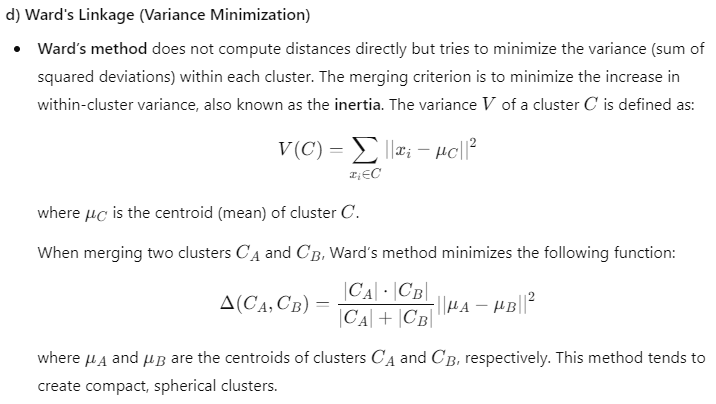


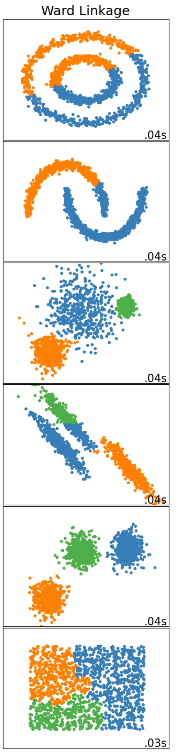




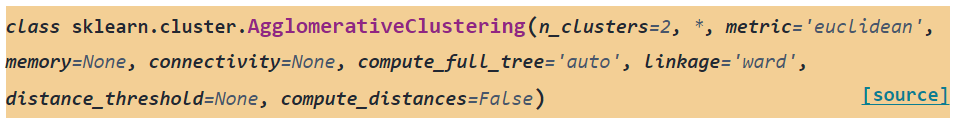




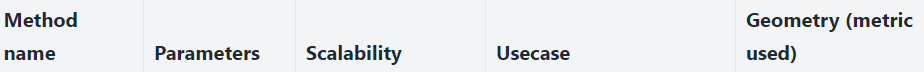


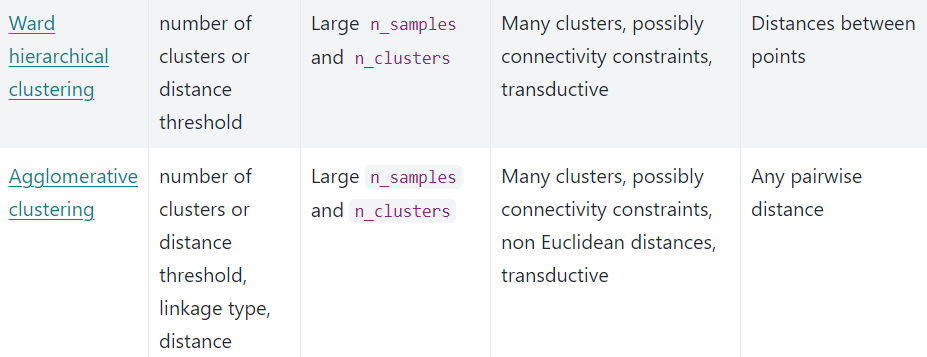


#### ورودی‌های الگوریتم:



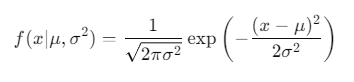
توضیحات کامل در [5] آمده است.



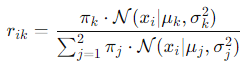
****

### Gaussian mixture

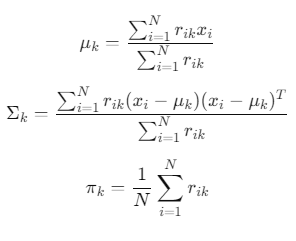
در این روش فرض می شود که توده ها از یک توزیع گوسی ایجاد شده اند.



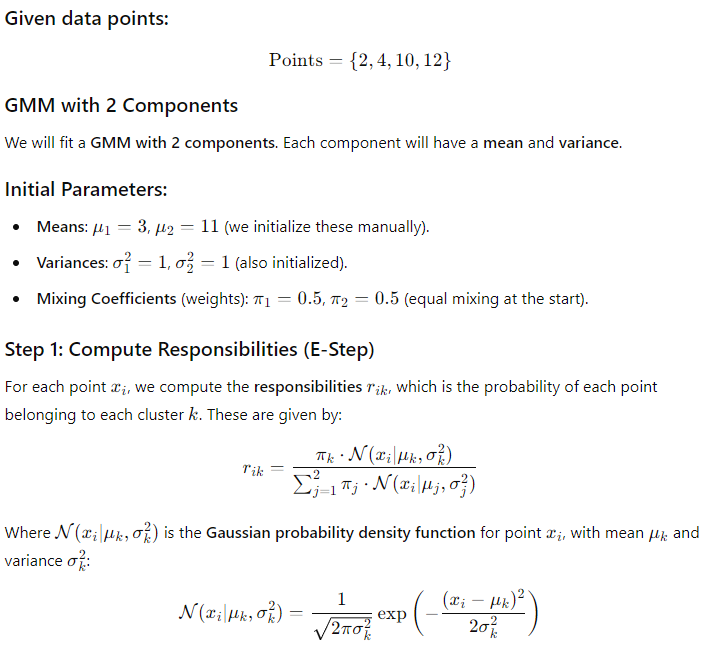
ابتدا فرض می‌شود که به تعداد خوشه توزیع گوسی داریم. برای میانگین و واریانس یک فرض اولیه می‌شود. همچنین برای هر نقطه احتمالاتی در نظر گرفته می‌شود که با چه احتمالی در این توزیع قرار می‌گیرد. سپس بررسی می‌شود که با چه انتظاری هر نقطه به یک توده تعلق دارد. این مرحله Expectation نام دارد. این مرحله Maximization نام دارد.

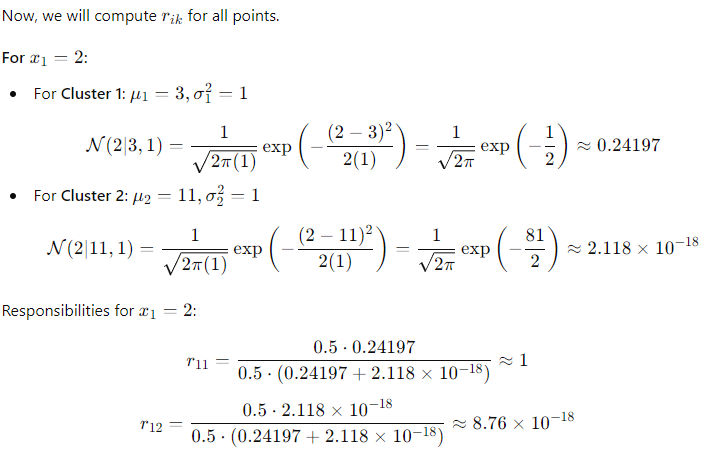


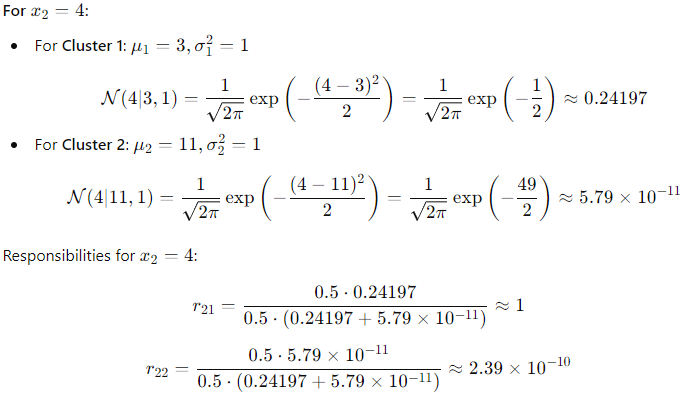
با محاسبه این درجه تعلق، حال میانگین و واریانس جدید هر توزیع محاسبه شده و احتمال حضور هر نقطه در هر توزیع هم به روز می‌شود.

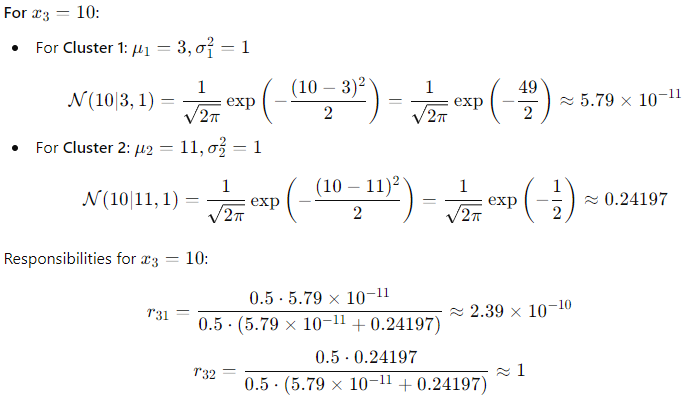


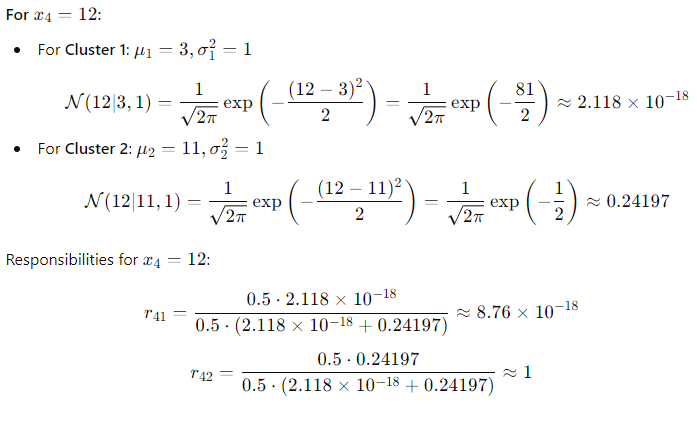
#### مثال عددی

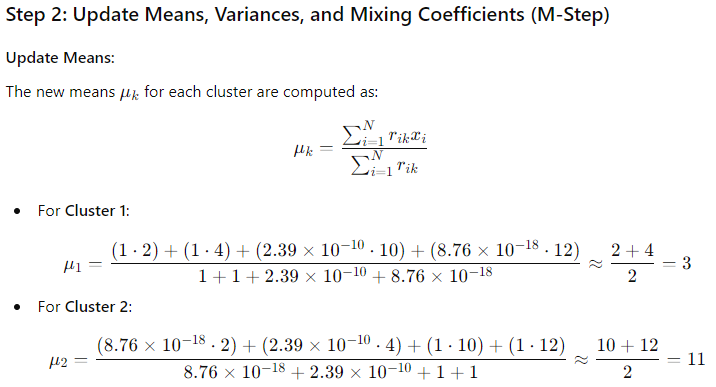


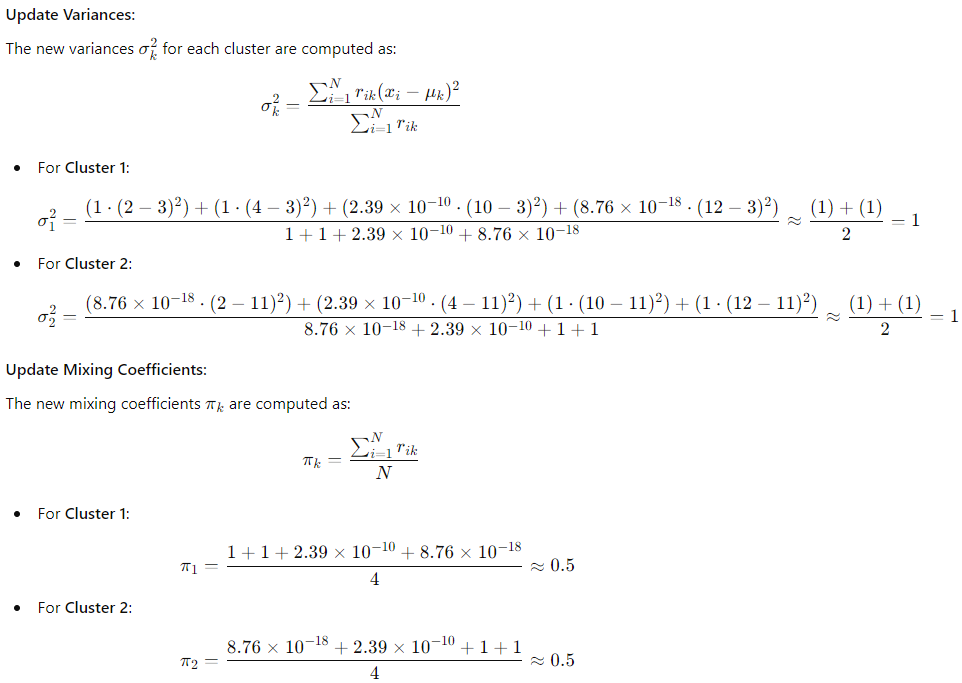






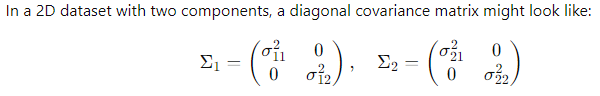




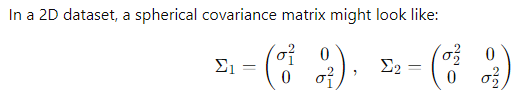


این روش چندین زیر روش هم دارد به این صورت که درباره واریانس توزیع ها فرض‌هایی می‌شود وبه چهار نوع diagonal, spherical, tied , full تقسیم می‌شود.

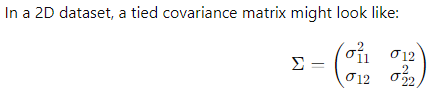
Diagonal:



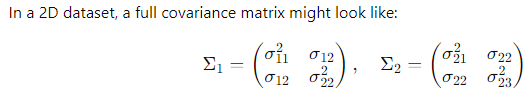
Spherical:



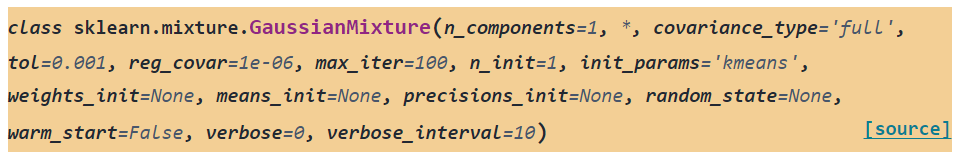
Tied:



Full:

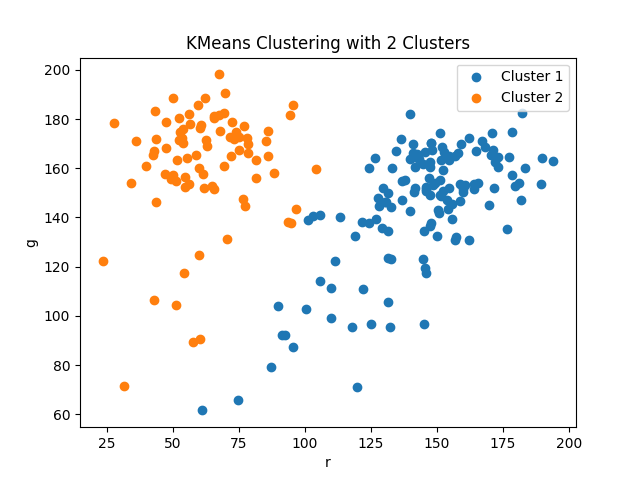
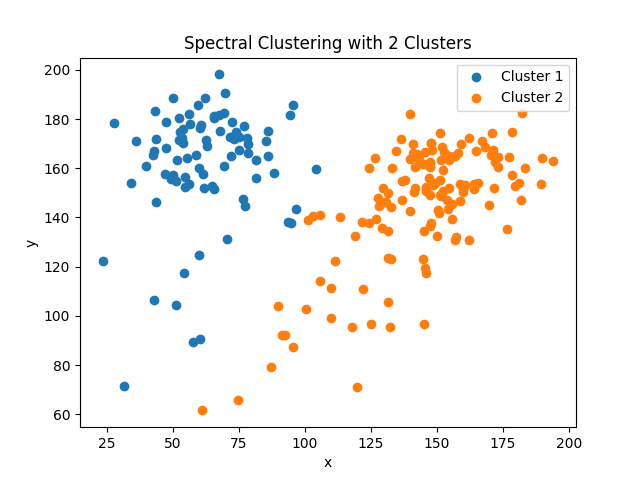


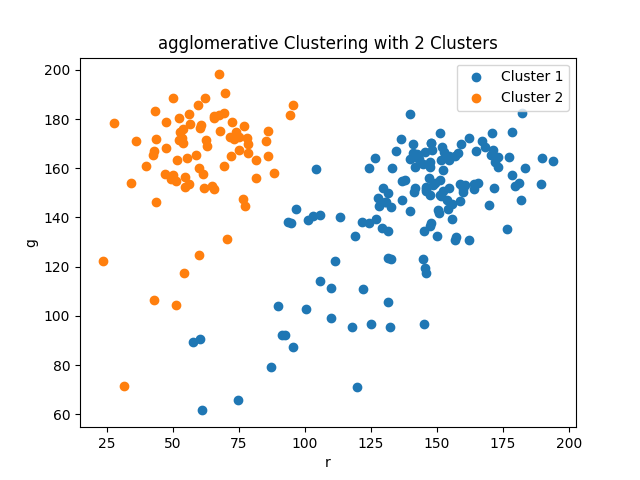
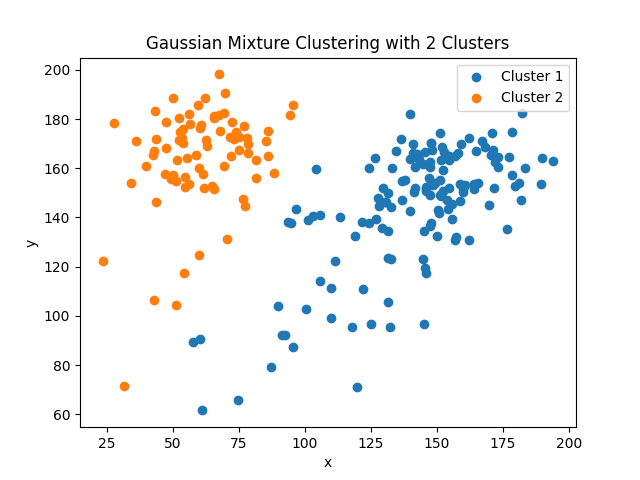
#### ورودی‌های الگوریتم:

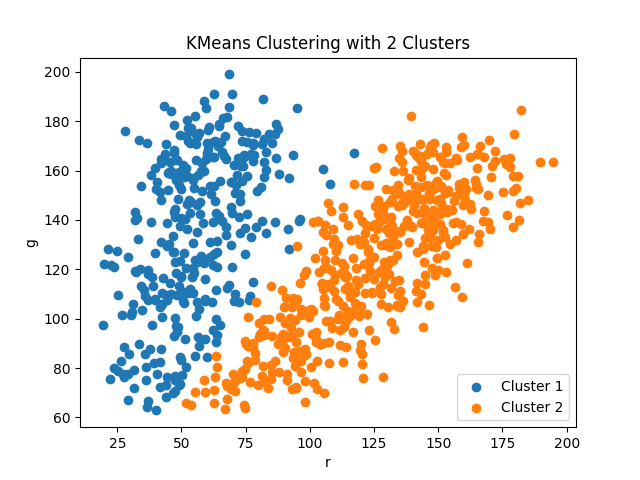
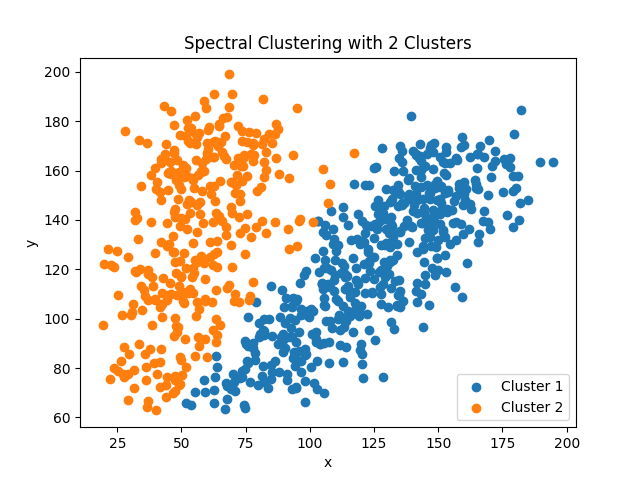


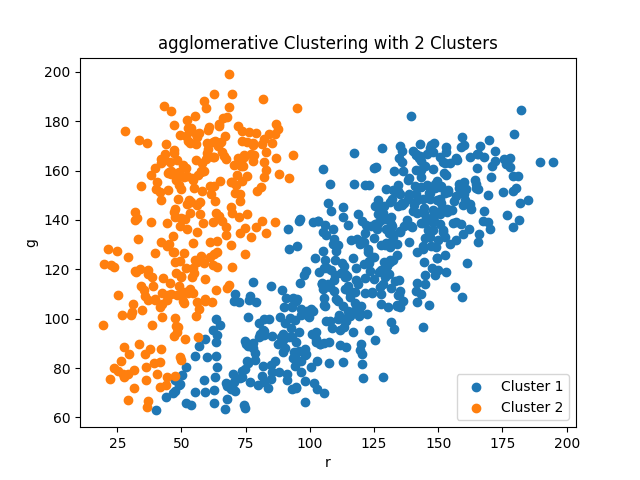
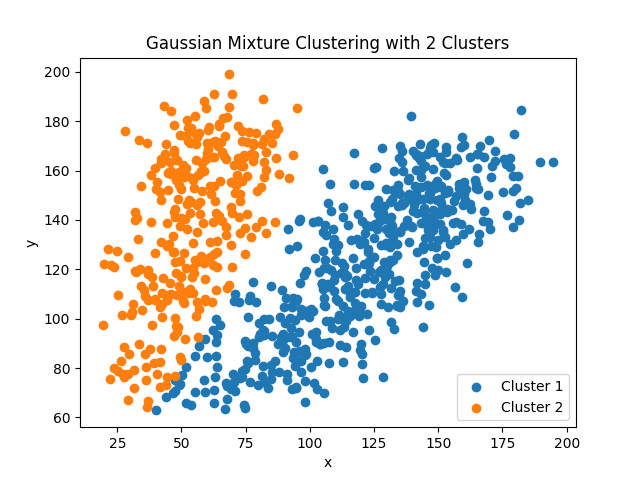
توضیحات کامل در [7] آمده است.

## پیاده ‌سازی و مقایسه روش‌ها روی داده‌های ما









در این پیاده سازی هدف فقط بررسی این مسئله بوده است که این روش‌ها توانایی حل مسئله ما را دارند که به نظر می‌‌رسد در وضعیت فعلی این چنین به نظر می‌رسد. این که مدل ها و پارامتر‌های مختلف هر روش بررسی شود هنوز انجام نشده است.

# منابع

1. <https://scikit-learn.org/stable/modules/clustering.html#references-5>
2. <https://en.wikipedia.org/wiki/Spectral_clustering>
3. <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans>
4. <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.SpectralClustering.html#sklearn.cluster.SpectralClustering>
5. <https://scikit-learn.org/stable/modules/generated/sklearn.cluster.AgglomerativeClustering.html#sklearn.cluster.AgglomerativeClustering>
6. <https://scikit-learn.org/stable/modules/mixture.html#mixture>
7. <https://scikit-learn.org/stable/modules/generated/sklearn.mixture.GaussianMixture.html#sklearn.mixture.GaussianMixture>
8. <https://www.ibm.com/topics/autoencoder>