**فصل 19 خوشه‌بندی[[1]](#footnote-1)**

**19.0 مقدمه**

در بیشتر بخش‌های کتاب ما به یادگیری نظارت شده پرداختیم - جایی که هم به ویژگی و هم به دسته یا برچسب دسته دسترسی داریم. متأسفانه همیشه این‌طور نیست. اغلب، ما با موقعیت‌هایی مواجه می‌شویم که فقط ویژگی را می‌شناسیم. به‌عنوان‌مثال، تصور کنید که ما رکوردهای فروش را از یک فروشگاه مواد غذایی داریم و می‌خواهیم فروش را باتوجه‌ به عضویت یا عدم عضویت خریدار در یک باشگاه تخفیف تقسیم کنیم. این امر با استفاده از یادگیری نظارت شده غیرممکن خواهد بود؛ زیرا ما هدف یا برچسبی برای آموزش و ارزیابی مدل‌های خود نداریم. بااین‌حال، گزینه دیگری وجود دارد: یادگیری بدون نظارت. اگر رفتار اعضای باشگاه تخفیفان و افراد غیرعضو در فروشگاه مواد غذایی واقعاً متفاوت باشد، میانگین تفاوت رفتار بین دو عضو کمتر از میانگین تفاوت رفتار بین یک عضو و خریدار غیرعضو خواهد بود. به‌عبارت‌دیگر، دو دسته از مشاهدات وجود خواهد داشت. هدف الگوریتم‌های خوشه‌بندی شناسایی گروه‌بندی‌های پنهان مشاهدات است که اگر به‌خوبی انجام شود، به ما امکان می‌دهد تا کلاس مشاهدات را حتی بدون بردار هدف پیش‌بینی کنیم. الگوریتم‌های خوشه‌بندی زیادی وجود دارد و رویکردهای متنوعی برای شناسایی خوشه‌ها در داده‌ها دارند. در این فصل، گزیده‌ای از الگوریتم‌های خوشه‌بندی را با استفاده از scikit-learn و نحوه استفاده از آنها در عمل پوشش خواهیم داد.

**19.1 خوشه‌بندی با استفاده از K-Means**

**مسئله**

شما می‌خواهید مشاهدات را در گروه، گروه‌بندی کنید.

**راه‌حل**

از خوشه‌بندی k-means استفاده کنید:



**توضیح**

خوشه‌بندی k-means یکی از رایج‌ترین تکنیک‌های خوشه‌بندی است. در خوشه‌بندی k-means، الگوریتم تلاش می‌کند تا مشاهدات را در k گروه، گروه‌بندی کند که هر گروه دارای واریانس تقریباً مساوی است. تعداد گروه‌ها، k، توسط کاربر به‌عنوان یک هایپرپارامتر مشخص می‌شود. به طور خاص، در k-means:

1. k نقطه به عنوان «مرکز» خوشه‌ها در نقاط تصادفی ایجاد می‌شوند.
2. برای هر مشاهده:
   1. فاصله بین هر مشاهده و k نقطه مرکزی محاسبه می‌شود.
   2. مشاهده به خوشه‌ای که نزدیک‌ترین نقطه مرکزی را به آن دارد اختصاص داده می‌شود.
3. نقاط مرکزی به روزرسانی شده و به میانگین داده‌های خوشه‌های مربوط به خود تغییر می‌کنند.
4. . مراحل 2 و 3 تکرار می‌شوند تا زمانی که هیچ مشاهده‌ای بین خوشه‌ها جابجا نشود.

در این مرحله الگوریتم همگرا در نظر گرفته شده و متوقف می‌شود.

توجه به سه نکته در مورد k-means مهم است. ابتدا، خوشه‌بندی k-means فرض می‌کند که خوشه‌ها به شکل محدب هستند (به‌عنوان‌مثال، یک دایره، یک کره). دوم، همه ویژگی‌ها به یک اندازه مقیاس‌بندی شده‌اند. در راه‌حل، ما ویژگی‌ها را برای برآورده‌کردن این فرض استاندارد کردیم. سوم، گروه‌ها متعادل هستند (یعنی تعداد مشاهدات تقریباً یکسانی دارند). اگر مشکوک باشیم که نمی‌توانیم به این مفروضات دست پیدا کنیم، ممکن است روش‌های خوشه‌بندی دیگری را امتحان کنیم.

در scikit-learn، خوشه‌بندی k-means در کلاس KMeans پیاده‌سازی می‌شود. مهم‌ترین پارامتر n\_cluster است که تعداد خوشه‌های k را تعیین می‌کند. در برخی موقعیت‌ها، ماهیت داده‌ها مقدار k را تعیین می‌کند (به‌عنوان‌مثال، داده‌های دانش‌آموزان مدرسه یک خوشه در هر کلاس دارند)، اما اغلب ما تعداد خوشه‌ها را نمی‌دانیم. در این موارد می‌خواهیم k را بر اساس معیارهایی انتخاب کنیم. برای مثال، ضرایب silhouette شباهت درون خوشه‌ها را در مقایسه با شباهت بین خوشه‌ها اندازه‌گیری می‌کند. علاوه بر این، ازآنجایی‌که خوشه‌بندی k-means از نظر محاسباتی هزینه‌بر است، ممکن است بخواهیم از تمام هسته‌های رایانه خود استفاده کنیم. ما می‌توانیم این کار را با تنظیم n\_jobs=-1 انجام دهیم.

در راه‌حل خود، ما کمی تقلب کردیم و از داده گل زنبق استفاده کردیم که می‌دانیم در آن سه کلاس وجود دارد؛ بنابراین، k = 3 را تنظیم می‌کنیم. می‌توانیم از labels\_ برای دیدن کلاس های پیش‌بینی‌شده هر مشاهده استفاده کنیم:



اگر این را با کلاس واقعی مشاهده مقایسه کنیم، می‌توانیم ببینیم که علی‌رغم تفاوت در برچسب‌های کلاس (یعنی 1، 2، و 3)، k-means به طور معقولی خوب عمل کرده است:





بااین‌حال، همان‌طور که ممکن است تصور کنید، اگر تعداد خوشه‌ها را اشتباه انتخاب کنیم، عملکرد k-means به طور قابل‌توجهی کاهش می‌یابد، حتی به طور بحرانی.

در نهایت، مانند سایر روش‌های یادگیری، می‌توانیم از خوشه آموزش‌دیده برای پیش‌بینی مقدار مشاهدات جدید استفاده کنیم:





پیش‌بینی می‌شود که مشاهده متعلق به خوشه‌ای باشد که نقطه مرکزی آن نزدیک‌ترین نقطه است. ما حتی می‌توانیم از cluster\_centers\_ برای دیدن آن نقاط مرکزی استفاده کنیم:





**همچنین ببینید:**

[**Introduction to K-means Clustering, DataScience.com**](https://www.datascience.com/blog/k-means-clustering)

**19.2 افزایش سرعت خوشه‌بندیK-Means**

**مسئله**

شما می‌خواهید مشاهدات را به گروه‌های k گروه‌بندی کنید، اما k-means خیلی طول می‌کشد.

**راه‌حل**

از mini-batch k-means استفاده کنید:



**توضیح**

mini-batch k-means مشابه الگوریتم k-means موردبحث در دستورالعمل 19.1 عمل می‌کند. بدون پرداختن به جزئیات زیاد، تفاوت در این است که در mini-batch k-means پرهزینه‌ترین مرحله محاسباتی تنها بر روی نمونه‌ای تصادفی از مشاهدات انجام می‌شود، برخلاف همه مشاهدات. این رویکرد می‌تواند زمان موردنیاز برای الگوریتم برای یافتن همگرایی (یعنی برازش داده‌ها) را با هزینه کمی از نظر کیفیت به میزان قابل‌توجهی کاهش دهد.

MiniBatchKMeans مشابه KMeans کار می‌کند، با یک تفاوت مهم: پارامتر. batch\_size تعداد مشاهدات انتخاب شده تصادفی در هر دسته توسطbatch\_size کنترل می‌شود. هرچه اندازه دسته بزرگ‌تر باشد، فرایند آموزش از نظر محاسباتی هزینه بیشتری دارد.

**19.3 خوشه‌بندی با استفاده از Meanshift**

**مسئله**

شما می‌خواهید مشاهدات را بدون فرض تعداد خوشه‌ها یا شکل آنها گروه‌بندی کنید.

**راه‌حل**

از خوشه‌بندی Meanshift استفاده کنید:



**توضیح**

یکی از معایب خوشه‌بندی k-means که قبلاً در مورد آن صحبت کردیم این است که قبل از آموزش باید تعداد خوشه‌ها را تنظیم کنیم، k، و روش فرضیاتی را در مورد شکل خوشه‌ها ایجاد کرد. یکی از الگوریتم‌های خوشه‌بندی بدون این محدودیت‌ها، Meanshift است.

Meanshift یک مفهوم ساده است، اما توضیح آن تا حدودی دشوار است؛ بنابراین، قیاس ممکن است بهترین رویکرد باشد. یک زمین فوتبال بسیار مه‌آلود را تصور کنید (یعنی یک فضای دوبعدی ویژگی) با 100 نفر که روی آن ایستاده‌اند (یعنی مشاهدات ما). به دلیل مه‌آلود بودن، انسان فقط می‌تواند فاصله کمی را ببیند. هر دقیقه هر فردی به اطراف نگاه می‌کند و قدمی در جهت بیشترین افرادی که می‌تواند ببیند برمی‌دارد. باگذشت زمان، مردم شروع به گروه شدن می‌کنند و به طور مکرر به سمت جمعیت بزرگ‌تر و بزرگ‌تر قدم برمی‌دارند. نتیجه نهایی، خوشه‌هایی از افراد در اطراف میدان است. افراد به خوشه‌هایی اختصاص داده می‌شوند که در نهایت در آن قرار می‌گیرند.

اجرای واقعی MeanShift توسط scikit-learn پیچیده‌تر است؛ اما از همان منطق اولیه پیروی می‌کند. MeanShift دو پارامتر مهم دارد که باید از آنها آگاه باشیم. ابتدا، bandwidthپهنای شعاع ناحیه (یعنی هسته[[2]](#footnote-2)) را تعیین می‌کند که یک مشاهده برای تعیین جهت تغییر استفاده می‌کند. در قیاس ما، پهنای باند این بود که فرد تا چه حد می‌تواند در مه ببیند. ما می‌توانیم این پارامتر را به‌صورت دستی تنظیم کنیم، اما به طور پیش‌فرض یک پهنای باند معقول به طور خودکار (با افزایش قابل‌توجه هزینه محاسباتی) تخمین زده می‌شود. ثانیاً، گاهی اوقات در meanshift هیچ مشاهدات دیگری در هسته یک مشاهده وجود ندارد. یعنی یک نفر در زمین فوتبال ما نمی‌تواند یک نفر دیگر را ببیند. به طور پیش‌فرض، MeanShift همه این مشاهدات «یتیم»[[3]](#footnote-3) را به هسته نزدیک‌ترین مشاهده اختصاص می‌دهد. بااین‌حال، اگر بخواهیم این یتیمان را کنار بگذاریم، می‌توانیم cluster\_all=False را تنظیم کنیم که در آن مشاهدات یتیم برچسب -1 داده می‌شود.

همچنین ببینید:

[The mean shift clustering algorithm, EFAVDB](http://efavdb.com/mean-shift/)

**19.4 خوشه‌بندی با استفاده از DBSCAN**

**مسئله**

شما می‌خواهید مشاهدات را در خوشه‌هایی با چگالی بالا گروه‌بندی کنید.

**راه‌حل**

از خوشه‌بندی DBSCAN استفاده کنید:



**توضیح**

انگیزه DBSCAN این ایده است که خوشه‌ها مناطقی هستند که بسیاری از مشاهدات به‌صورت متراکم در کنار هم قرار می‌گیرند و هیچ فرضی در مورد شکل خوشه نمی‌کنند. به طور خاص، در DBSCAN:

1. یک مشاهده تصادفی، ، انتخاب شده است.
2. اگر دارای حداقل تعداد همسایه‌های نزدیک باشد، آن را بخشی از یک خوشه در نظر می‌گیریم.
3. مرحله 2 به‌صورت بازگشتی برای همه همسایه‌های و سپس همسایه همسایه و غیره تکرار می‌شود. اینها مشاهدات اصلی خوشه هستند.
4. هنگامی که مرحله 3 از مشاهدات نزدیک به پایان رسید، یک نقطه تصادفی جدید انتخاب می‌شود (یعنی شروع مجدد مرحله 1).

پس از تکمیل، مجموعه‌ای از مشاهدات اصلی و هسته‌ای برای تعدادی از خوشه‌ها داریم. در نهایت، هر مشاهده‌ای نزدیک به یک خوشه اما نه یک نمونه هسته، بخشی از یک خوشه در نظر گرفته می‌شود، درحالی‌که هر مشاهده‌ای که به خوشه نزدیک نباشد، به آن برچسب پرت[[4]](#footnote-4) زده می‌شود.

DBSCAN دارای سه پارامتر اصلی برای تنظیم است:

* eps: حداکثر فاصله از یک مشاهده برای مشاهده دیگری که همسایه آن در نظر گرفته شود.
* min\_samples: حداقل تعداد مشاهدات کمتر از فاصله eps از یک مشاهده برای آن که به‌عنوان یک مشاهده اصلی در نظر گرفته شود.
* metric: معیار فاصله‌ای که توسط eps استفاده می‌شود - برای مثال minkowski یا euclidean (توجه داشته باشید که اگر فاصله مینکوفسکی استفاده شود، پارامتر p را می‌توان برای تنظیم توان متریک مینکوفسکی استفاده کرد).

اگر به خوشه‌های موجود در داده‌های آموزشی خود نگاه کنیم، می‌بینیم که دو خوشه شناسایی شده‌اند، 0 و 1، درحالی‌که مشاهدات پرت برچسب -1 دارند:





**همچنین ببینید:**

[DBSCAN, Wikipedia](https://en.wikipedia.org/wiki/DBSCAN)

**19.5 خوشه‌بندی با استفاده از ادغام سلسله‌مراتبی**

**مسئله**

شما می‌خواهید مشاهدات را با استفاده از سلسله‌مراتبی از خوشه‌ها گروه‌بندی کنید.

**راه‌حل**

از خوشه‌بندی تجمعی[[5]](#footnote-5) استفاده کنید:



**توضیح**

خوشه‌بندی تجمعی یک الگوریتم خوشه‌بندی سلسله‌مراتبی قدرتمند و انعطاف‌پذیر است. در خوشه‌بندی تجمعی، همه مشاهدات به‌عنوان خوشه‌های خود شروع می‌شوند. در مرحله بعد، خوشه‌هایی که برخی از معیارها را دارند با هم ادغام می‌شوند. این فرایند تکرار می‌شود، خوشه‌ها درحال ‌رشد تا رسیدن به نقطه پایانی هستند. در scikit-learn، AgglomerativeClustering از پارامتر linkage برای تعیین استراتژی ادغام استفاده می‌کند تا موارد زیر را به حداقل برساند:

1. واریانس خوشه‌های ادغام شده (ward)
2. میانگین فاصله بین مشاهدات از جفت خوشه (average)
3. حداکثر فاصله بین مشاهدات از جفت خوشه (complete)

دانستن دو پارامتر دیگر مفید است. ابتدا، پارامتر affinity معیار فاصله مورد استفاده برای linkpage (minkowski, euclidean و غیره) را تعیین می‌کند. دوم، n\_clusters تعداد خوشه‌هایی را که الگوریتم خوشه‌بندی تلاش می‌کند پیدا کند را تعیین می‌کند. یعنی، خوشه‌ها به طور متوالی ادغام می‌شوند تا زمانی که فقط n\_cluster باقی بماند.

مانند سایر الگوریتم‌های خوشه‌بندی که پوشش داده‌ایم، می‌توانیم از labels\_ برای دیدن خوشه‌ای استفاده کنیم که در آن هر مشاهده اختصاص‌داده ‌شده است:





1. clustering [↑](#footnote-ref-1)
2. kernel [↑](#footnote-ref-2)
3. orphan [↑](#footnote-ref-3)
4. outlier [↑](#footnote-ref-4)
5. agglomerative [↑](#footnote-ref-5)