**فصل ٩: کاهش ابعاد با استفاده از استخراج ویژگی**

**٩.٠. مقدمه**

دسترسی به هزاران و حتی صدها هزار ویژگی، امر رایجی است. برای مثال ، در فصل ۸ تصویر رنگی ۲۵۶ \* ۲۵۶ پیکسل را به ۱۹۶،۶۰۸ ویژگی تبدیل کردیم. علاوه بر این ، چون هر یک از این پیکسل‌ها می­توانند یکی از ۲۵۶ مقدار ممکن را داشته باشند و در نهایت ۲۵۶١٩٦٦٠٨ پیکربندی مختلف را مشاهده می­­­‌کنیم. این مساله مشکل‌ساز است زیرا ما عملا ً هرگز قادر به جمع‌آوری مشاهدات کافی برای پوشش دادن حتی بخش کوچکی از این پیکربندی‌ها نخواهیم بود و الگوریتم‌های یادگیری ما داده‌های کافی برای عملکرد صحیح ندارند.

خوشبختانه همه‌ی ویژگی­ها به صورت یکسان ایجاد نمی­شوند و هدف از استخراج ویژگی برای کاهش ابعاد، این است که مجموعه‌ی ویژگی­های خود را به، p original ، تغییر دهیم به طوری که به یک مجموعه جدید یعنی، new p، می­رسیم، که در آن pnew <poriginal است، در حالی که هنوز بسیاری از اطلاعات اساسی حفظ می‌شود. به عبارت دیگر، ما تعداد ویژگی­ها را کاهش می‌دهیم و تنها با اندکی اتلاف (کاهش) توانایی داده­هایمان در تولید پیش‌بینی­های با کیفیت بالا مواجه می­شویم. در این بخش تعدادی از تکنیک­های استخراج ویژگی را برای انجام این کار پوشش خواهیم داد.

یکی از نکات منفی تکنیک‌های استخراج ویژگی که مورد بحث قرار می‌دهیم این است که ویژگی‌های جدیدی که تولید می­کنیم توسط انسان قابل تفسیر نیستند. آنها شامل توانایی کافی برای آموزش مدل­های ما هستند، اما در چشم انسان به عنوان مجموعه‌ای از اعداد تصادفی ظاهر می­شوند. اگر ‌بخواهیم توانایی خود را در تفسیر مدل‌هایمان حفظ کنیم، کاهش ابعاد از طریق انتخاب ویژگی گزینه بهتری است.

**٩.١. کاهش ویژگی­ها با استفاده از اجزای اصلی**

**مسئله:**

با توجه به مجموعه‌ای از ویژگی­ها، شما می­خواهید تعداد ویژگی­ها را کاهش دهید در حالی که واریانس داده­ها حفظ شود.

**راه حل:**

از تجزیه و تحلیل مؤلفه اصلی با PCA در scikit استفاده کنید:





**بحث**

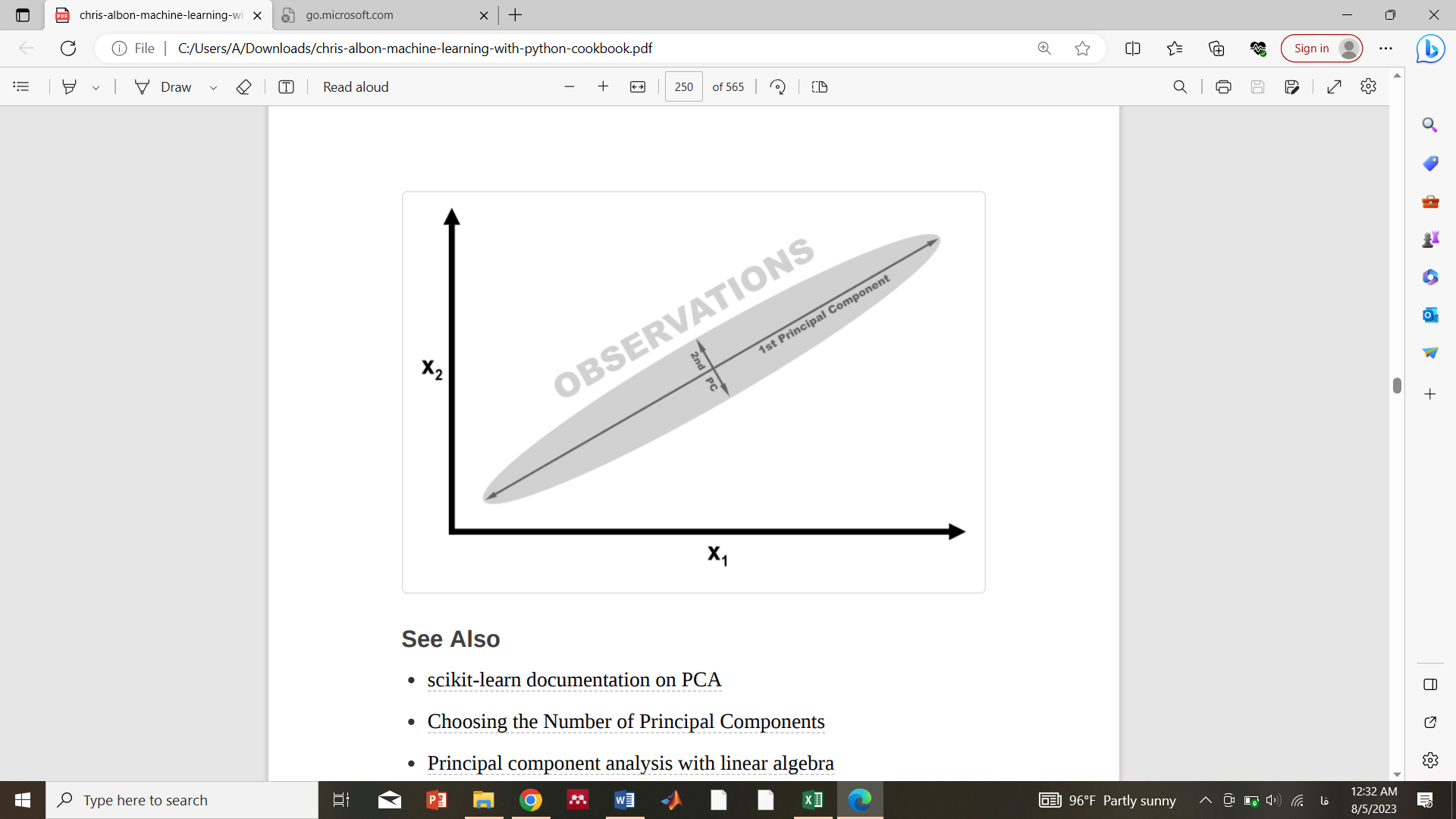
تحلیل مولفه­های اصلی (PCA) تکنیک محبوب کاهش ابعاد خطی است. PCA مشاهدات را بر روی مولفه­های اصلی (با امید کم تر) ماتریس ویژگی که بیش‌ترین واریانس را حفظ می­کنند، انجام می­دهد. PCA یک تکنیک بدون نظارت است، به این معنی که از اطلاعات بردار هدف استفاده نمی­کند و در عوض تنها ماتریس ویژگی را در نظر می­گیرد.

برای توصیف ریاضی نحوه کار pca، منابع خارجی فهرست ‌شده در پایان این دستورالعمل را ببینید. با این حال، ما می‌توانیم با استفاده از یک مثال ساده، تفکرات و اندیشه­های پشت pca را درک کنیم. در شکل زیر، داده‌های ما شامل دو ویژگی 1x و2x هستند. با نگاهی به این تصویرسازی، واضح است که مشاهدات مانند سیگاری با طول زیاد و ارتفاع بسیار کم پخش می­شوند. به طور دقیق‌تر می‌توان گفت که واریانس "طول" به طور قابل‌توجهی بزرگ‌تر از "ارتفاع" است. به جای طول و ارتفاع، به "جهت­ها" با بیش‌ترین واریانس به عنوان مولفه اصلی اول و "جهت" با دومین واریانس به عنوان مولفه اصلی دوم (و غیره) اشاره می­کنیم.

اگر بخواهیم ویژگی­های خود را کاهش دهیم، یک استراتژی این است که تمام مشاهدات در فضای دو بعدی خود را بر روی مولفه اصلی یک بعدی قرار دهیم. ما اطلاعات ثبت شده در مولفه اصلی دوم را از دست می­دهیم، اما در برخی شرایط این یک موازنه قابل قبول خواهد بود. این همان PCA است.

PCA در scikit-learn با استفاده از روش pca پیاده سازی می­شود. n\_components بستگی به آرگومان ارائه شده دو عملیات دارد. اگر آرگومان بزرگتر از 1 باشد، n\_components تعداد ویژگی­ها را برمی­گرداند. این موضوع منجر به این سوال می­شود که چگونه تعداد ویژگی­های بهینه را انتخاب کنیم؟ خوشبختانه ، اگر آرگومان n\_components بین 0 و 1 باشد، pca حداقل مقدار ویژگی­هایی را که واریانس زیادی را حفظ می­کنند، برمی­گرداند. استفاده از مقادیر 0.95 و 0.99 رایج است، به این معنی که به ترتیب 95% و 99% از واریانس ویژگی­های اصلی حفظ شده است. whiten=True مقادیر هر جزء اصلی را طوری تبدیل می­کند که میانگین و واریانس واحد آن­ها صفر باشد. پارامتر و آرگومان دیگر" svd\_solver="randomized است که یک الگوریتم تصادفی را برای یافتن اولین مؤلفه­های اصلی دراغلب اوقات با زمانِ به طور قابل توجهی کمتر پیاده سازی می‌کند.

خروجی راه حل ما نشان می‌دهد که PCA به ما اجازه می­دهد تا ابعاد خود را تا ۱۰ ویژگی کاهش دهیم در حالی که هنوز ۹۹ % اطلاعات (واریانس) را در ماتریس ویژگی حفظ می­کنیم.



**برای مطالعه بیشتر**

[scikit-learn documentation on PCA](https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html)

[Principal component analysis with linear algebra](https://oreil.ly/Uns61)

**۹.۲. کاهش ویژگی­ها در زمان غیرقابل تفکیک بودن داده­ها به صورت خطی**

**مسئله:**

فرض کنید داده­های خطی غیرقابل تفکیک دارید و می­خواهید ابعاد آن­ها را کاهش دهید.

**راه حل:**

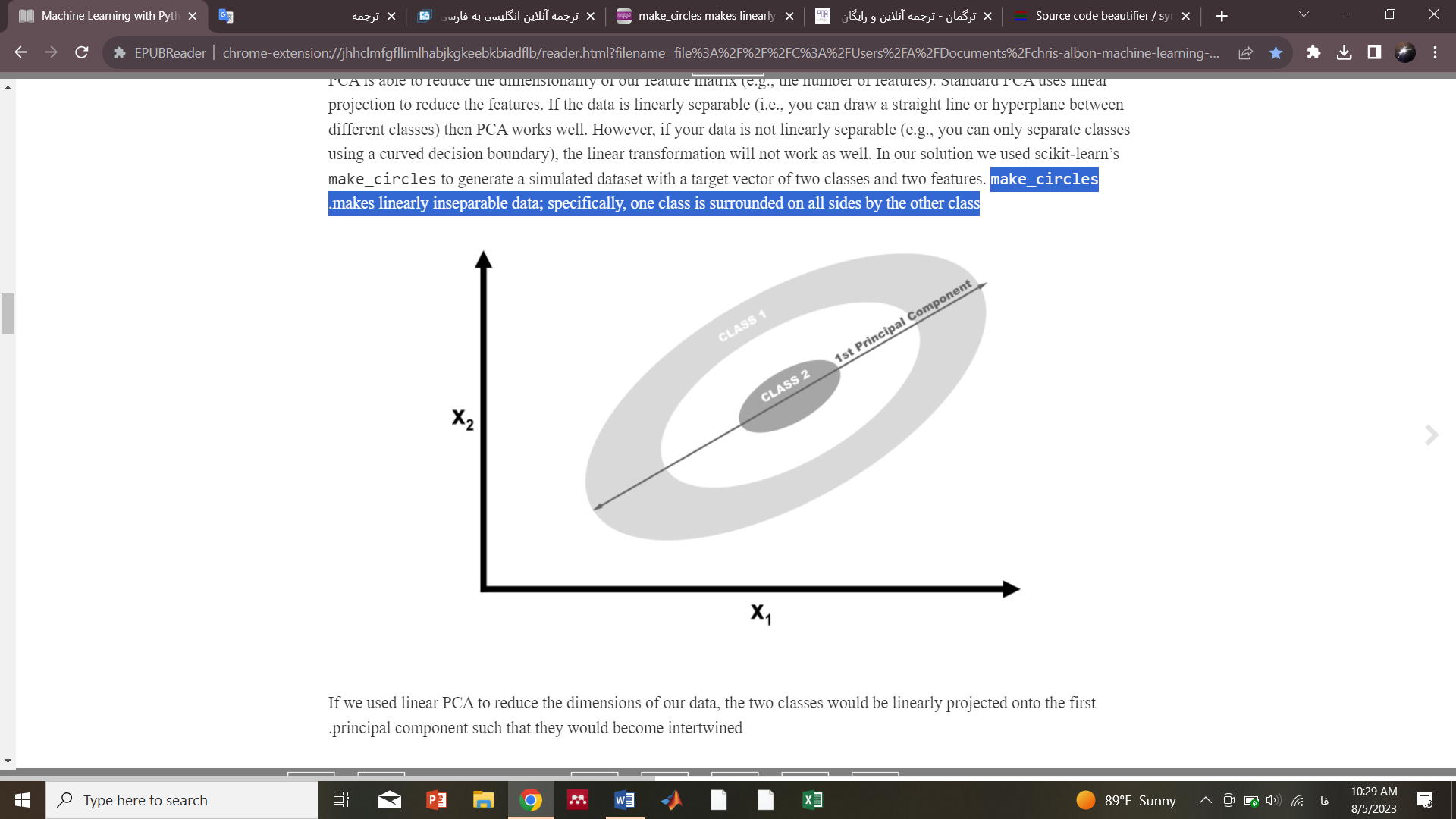
از بسط تحلیل مولفه‌های اصلی که از کرنل‌ها برای کاهش ابعاد غیرخطی استفاده می‌کند، استفاده کنید:



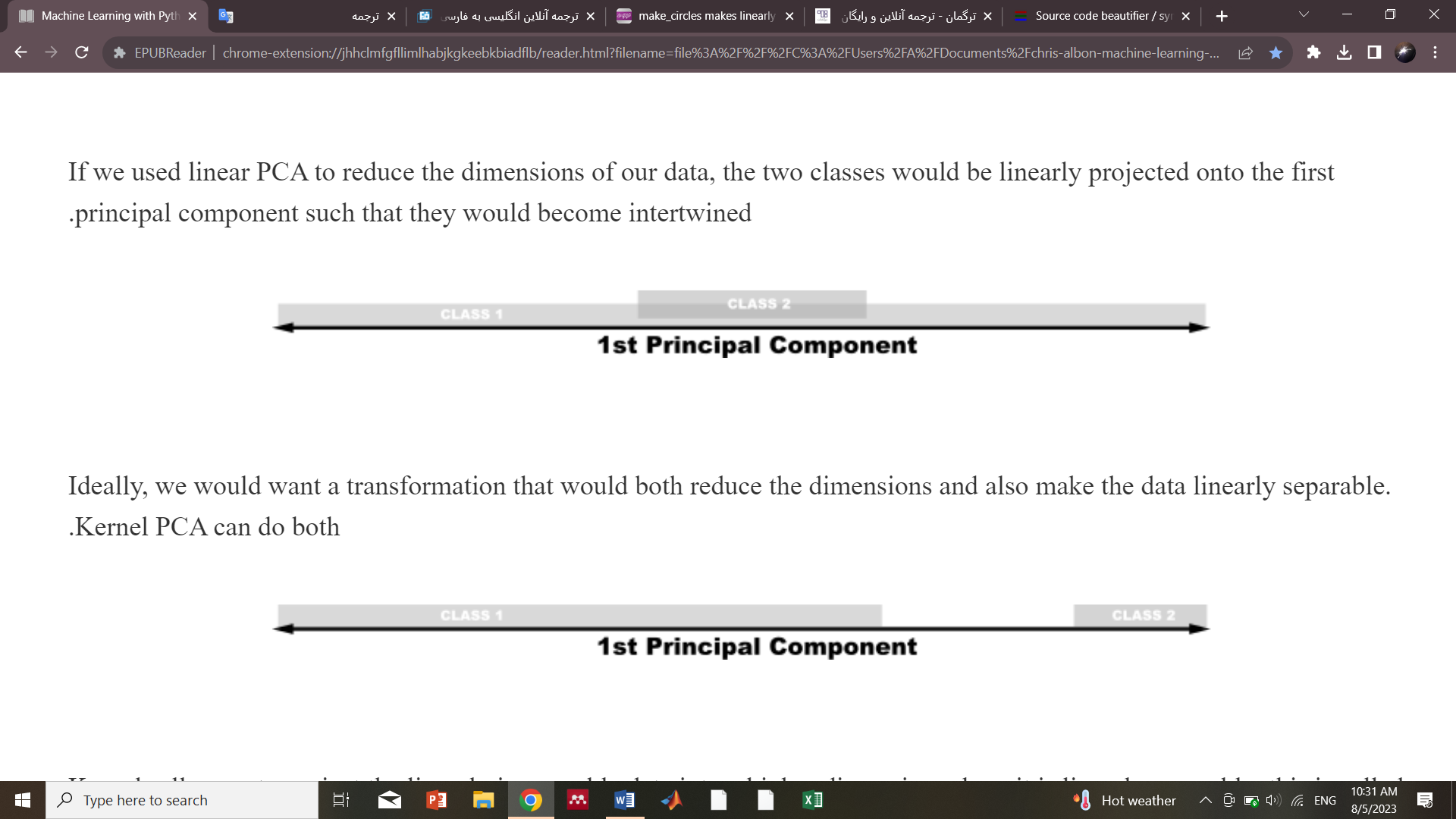


**بحث و گفت وگو**

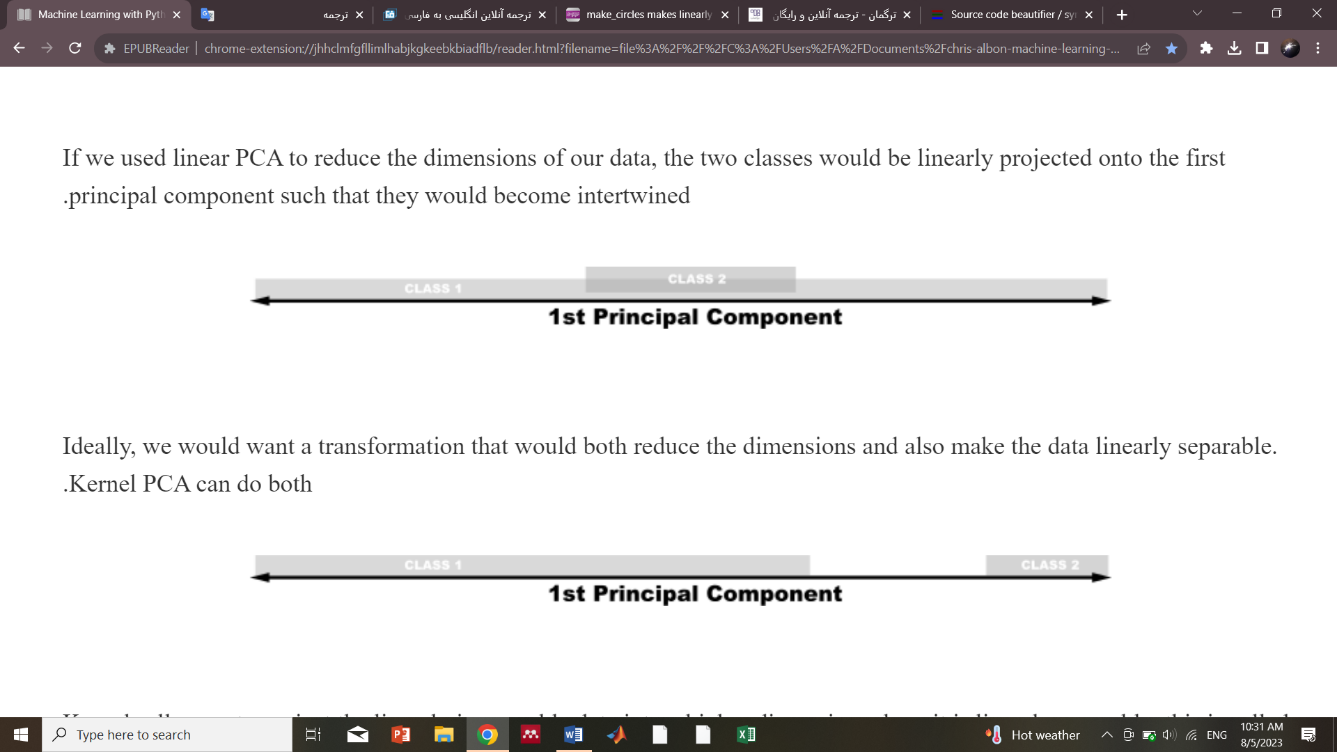
PCA قادر است ابعاد ماتریس ویژگی­های ما (به عنوان مثال، تعداد ویژگی­ها) را کاهش دهد. PCA استاندارد، از تصویرسازی خطی برای کاهش ویژگی­ها استفاده می­کند. اگر داده­ها به صورت خطی تفکیک پذیر باشند (یعنی بتوانید یک خط مستقیم یا یک صفحه بزرگ بین کلاس­های مختلف رسم کنید) آنگاه PCA به خوبی کار می­کند. با این حال، اگر داده­های شما به صورت خطی قابل تفکیک نباشند (به عنوان مثال، شما تنها می‌توانید کلاس­ها را با استفاده از منحنی تصمیم گیری خمیده از هم جدا کنید)، تبدیل خطی به خوبی کار نخواهد کرد. ما در راه حل خود از make\_circles در scikit-learn برای تولید یک مجموعه داده شبیه سازی شده با بردار هدف از دو کلاس و دو ویژگی استفاده کرده‌ایم. make\_circles داده­های غیرقابل تفکیک خطی ایجاد می‌کند. به طور خاص، یک کلاس از اطراف، توسط کلاس­های دیگر احاطه شده است.



اگر از PCA خطی برای کاهش ابعاد داده‌های خود استفاده کنیم، دو کلاس به‌صورت خطی روی مولفه اصلی اول نمایش داده می‌شوند به طوری که درهم ادغام شوند.



در حالت ایده آل، ما یک تبدیل می‌خواهیم که هم ابعاد را کاهش دهد و هم داده­ها را به صورت خطی تفکیک پذیر کند. Kernel PCA می‌تواند هر دو کار را انجام دهد.



کرنل‌ها به ما اجازه می‌دهند تا داده‌های غیر قابل تفکیک خطی را در ابعاد بالاتری قرار دهیم، جایی که به صورت خطی قابل تفکیک هستند. این روش، روش Kernels نامیده می­شود. اگر جزئیات ترفند Kernel را متوجه نمی­شوید نگران نباشید؛ کرنل‌ها را فقط به عنوان روش­های مختلف نمایش داده­ها در نظر بگیرید. تعدادی هسته وجود دارد که می‌توانیم از آن­ها در kernelPCA در scikit – Learning استفاده کنیم که با استفاده از پارامتر Kernel مشخص شده است. یک Kernel رایج برای استفاده، Kernel تابع پایه شعاعی گاوسی RBf است، اما گزینه­های دیگر عبارتند از کرنل چندجمله‌ای (Poly) و کرنل سیگموئید (sigmoid). ما حتی می‌توانیم یک تصویر سازی خطی را مشخص کنیم که نتایج مشابه PCA استاندارد را تولید خواهد کرد.

یکی از معایب Kernel PCA این است که پارامترهای متعددی وجود دارد که باید مشخص کنیم. به عنوان مثال، در دستور العمل 9.1، ما n\_components را روی 0.99 قرار دادیم تا PCA تعداد مؤلفه‌ها را برای حفظ 99 درصد واریانس انتخاب کند. ما این گزینه را در Kernel PCA نداریم. در عوض باید تعداد پارامترها را تعریف کنیم (به عنوان مثال n\_components=1). علاوه بر این، کرنل‌ها با فراپارامترهای خاص خود می‌آیند که ما باید آنها را تنظیم کنیم. برای مثال، تابع پایه شعاعی به یک مقدار گاما(gamma) نیاز دارد.

پس چگونه بفهمیم از چه مقادیری استفاده کنیم؟ از طریق آزمون و خطا. به طور خاص، می‌توانیم چندین بار مدل یادگیری ماشین خود را آموزش دهیم، هر بار با kernel متفاوت یا مقدار متفاوتی از پارامتر. هنگامی که ترکیبی از مقادیر را پیدا کردیم که بالاترین کیفیت مقادیر پیش بینی شده را تولید می­کنند، کار ما، انجام شده است. در فصل 12 با این استراتژی به طور کامل آشنا خواهیم شد.

**برای مطالعه بیشتر**

[scikit-learn documentation on Kernel PCA](https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.KernelPCA.html)

[Kernel tricks and nonlinear dimensionality reduction via RBF kernel PCA](https://sebastianraschka.com/Articles/2014_kernel_pca.html)

**٩.٣. کاهش ویژگی‌ها با حداکثر کردن تفکیک پذیری دسته‌ها**

**مسئله:**

شما می‌خواهید ویژگی­های مورد استفاده توسط یک دسته بندی کننده را کاهش دهید.

**راه حل:**

تحلیل تشخیص خطی (LDA) را امتحان کنید تا ویژگی­ها را بر روی محورهای مؤلفه‌ای که جداسازی کلاس­ها را به حداکثر می‌رساند نشان دهید:





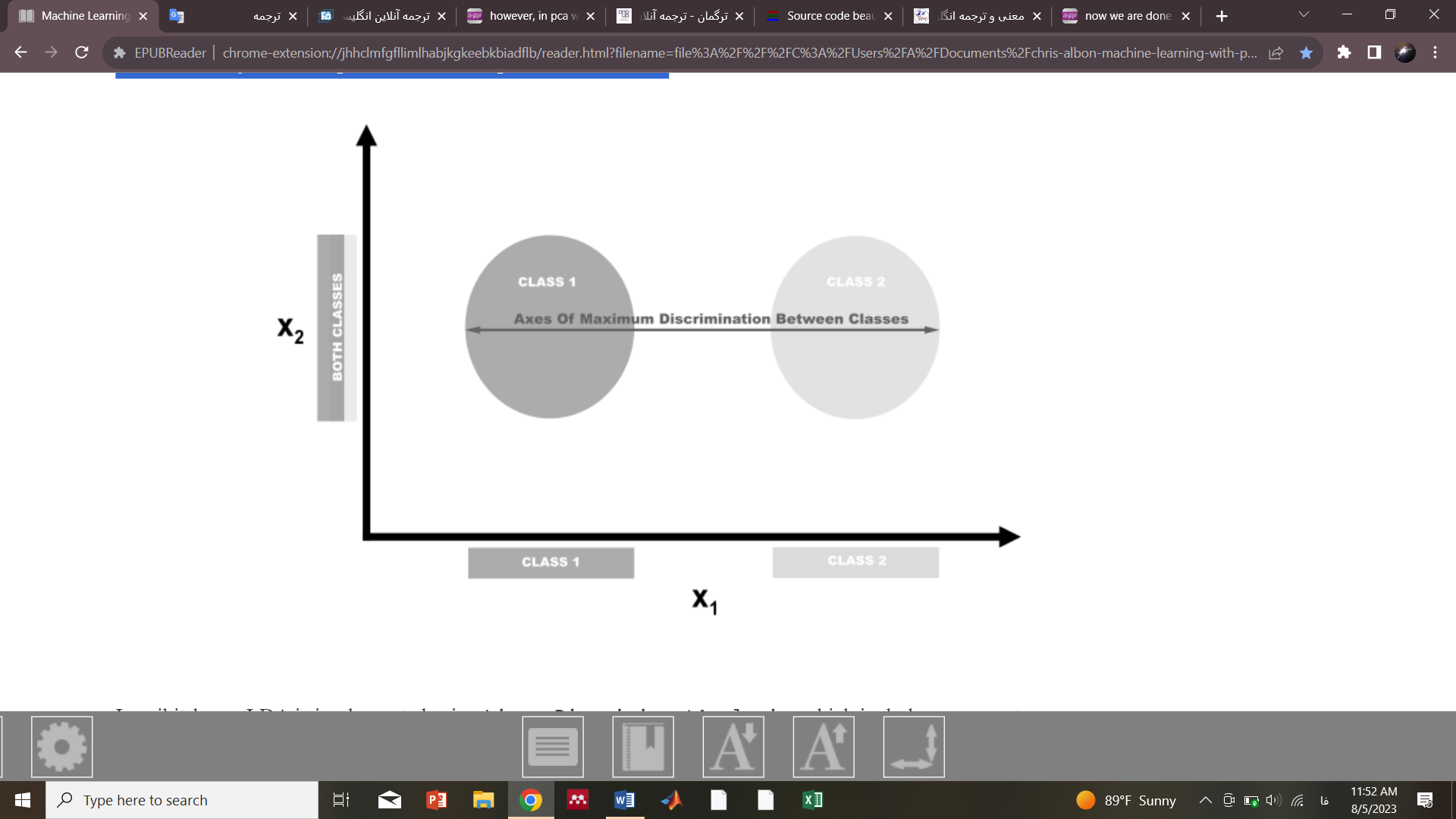
برای مشاهده مقدار واریانس توضیح داده شده توسط هر مؤلفه می­توانیم از explained\_variance\_ratio\_ استفاده کنیم. در راه حل ما مولفه منفرد بیش از ۹۹ درصد واریانس بیان می­شود:





**بحث**

LDA کلاس بندی است که یک تکنیک محبوب برای کاهش ابعاد نیز محسوب می‌شود. LDA مشابه تحلیل مؤلفه اصلی (PCA) عمل می‌کند، زیرا فضای ویژگی ما را در فضایی با ابعاد پایین‌تر نمایش می­دهد. با این حال، در PCA ما فقط به محورهای مؤلفه‌ای علاقه‌مند بودیم که واریانس داده‌ها را به حداکثر می‌رسانند، در حالی که در LDA ما هدف اضافه تری هم داریم که آن به حداکثر رساندن تفاوت بین کلاس‌ها است. در این مثال تصویری، داده­هایی شامل دو کلاس هدف و دو ویژگی داریم. اگر داده­ها را بر روی محور y قرار دهیم، دو کلاس به راحتی قابل تفکیک نیستند (یعنی با هم همپوشانی دارند)، در حالی که اگر داده­ها را بر روی محور x قرار دهیم، یک بردار ویژگی (یعنی ابعاد خود را یک به یک کاهش دادیم) باقی می‌ماند که همچنان قابلیت تفکیک کلاس را حفظ می­کند. البته در دنیای واقعی، رابطه بین کلاس­ها پیچیده‌تر و ابعاد آن­ها بیشتر خواهد بود، اما مفهوم یکسانی باقی می‌ماند.



LDA در Scikit – Learning، با استفاده از LinearDiscriminantAnalysis پیاده سازی می­شود که شامل یک پارامتر به نام n\_componentsاست که نشان دهنده تعداد ویژگی­هایی است که می­خواهیم برگردانیم. برای اینکه بفهمیم از چه مقدار آرگومان با n\_components استفاده کنیم (مثلاً چند پارامتر باید حفظ شود)، می‌توانیم از این واقعیت استفاده کنیم که \_explained\_variance\_ratio واریانس توضیح داده شده توسط هر ویژگی خروجی را برای ما بیان میکند و یک آرایه مرتب شده است. برای مثال:





به طور خاص، ما می‌توانیم LinearDiscriminantAnalysis را با n\_component تنظیم‌شده روی None اجرا کنیم تا نسبت واریانس توضیح داده شده توسط هر مولفه را برگردانیم، سپس محاسبه کنیم که چه تعداد مولفه برای رسیدن به آستانه واریانس توضیح داده شده(اغلب ۰.۹۵ یا ۰.۹۹) نیاز است:





**برای مطالعه بیشتر**

[Comparison of LDA and PCA 2D projection of Iris dataset](https://scikit-learn.org/stable/auto_examples/decomposition/plot_pca_vs_lda.html" \l "sphx-glr-auto-examples-decomposition-plot-pca-vs-lda-py)

[Linear Discriminant Analysis](https://sebastianraschka.com/Articles/2014_python_lda.html)

**۹.۴. کاهش ویژگی­ها با استفاده از فاکتورگیری ماتریسی**

**مسئله:**

یک ماتریس ویژگی از مقادیر غیر منفی دارید و می‌خواهید ابعاد آن را کاهش دهید.

**راه حل:**

از فاکتورگیری ماتریس غیر منفی (NMF) برای کاهش ابعاد ماتریس ویژگی استفاده کنید:





**بحث**

NMF یک تکنیک بدون نظارت برای کاهش ابعاد خطی است که ماتریس ویژگی را به ماتریس­هایی تبدیل می­کند (یعنی به چند ماتریس تقسیم می­شود که محصول آن­ها ماتریس اصلی را تقریب می‌زند) که نشان دهنده رابطه‌ی پنهان بین مشاهدات و ویژگی­های آن­ها است. به طور شهودی، NMF می‌تواند ابعاد را کاهش دهد زیرا در ضرب ماتریس، دو عامل (ماتریس‌های در حال ضرب شدن) می­توانند ابعاد بسیار کمتری نسبت به ماتریس نهایی داشته باشند. به طور رسمی، با توجه به تعداد مطلوب ویژگی­های بازگشتی، *r*، NMF ماتریس ویژگی ما را فاکتوربندی می‌کند به طوری که:

که در آن *V* ماتریس مشخصه ما است (یعنی، *d* مشخصه، *n* مشاهده)، W یک و H یک ماتریس است. با تنظیم مقدار *r* می‌توانیم میزان کاهش ابعاد مورد نظر را تعیین کنیم.

یکی از الزامات اصلی NMA، همانطور که از نام آن پیداست، این است که ماتریس ویژگی نمی‌تواند حاوی مقادیر منفی باشد. علاوه بر این، برخلاف PCA و تکنیک­های دیگری که ما بررسی کرده­ایم، NMA واریانس بیان شده ویژگی­های خروجی را در اختیار ما قرار نمی­دهد. بنابراین، بهترین راه برای یافتن مقدار بهینه n\_components، تلاش برای یافتن دامنه وسیعی از مقادیر است که بهترین نتیجه را در مدل نهایی ما تولید می‌کنند (به فصل ۱۲ مراجعه کنید).

**برای مطالعه بیشتر**

[Non-Negative Matrix Factorization (NMF)](https://en.wikipedia.org/wiki/Non-negative_matrix_factorization)

**۹.۵ .کاهش ویژگی­های داده­های پراکنده**

**مسئله:**

یک ماتریس ویژگی پراکنده دارید و می‌خواهید ابعاد آن را کاهش دهید.

**راه حل:**

از Truncated Singular Value Decomposition (TSVD)استفاده کنید:





**بحث**

TSVD شبیه به PCA است و در واقع PCA اغلب در یکی از مراحل خود از Singular Value Decomposition (SVD) استفاده می‌کند. در SVD معمولی، با توجه به ویژگی­های SVD، *d* ماتریس­های عاملی ایجاد می‌کند که هستند، در حالی که TSVD فاکتورهایی را برمی گرداند که هستند، که n قبلا توسط یک پارامتر مشخص شده است. مزیت کاربردی TSVD این است که برخلاف PCA، بر روی ماتریس­های ویژگی پراکنده کار می­کند.

یک مساله در مورد TSVD این است که به دلیل چگونگی استفاده از یک مولد اعداد تصادفی، علائم خروجی می‌تواند بین اتصالات تغییر کند. یک راهکار ساده، استفاده از تناسبِ تنها یک‌بار در هر خط پیش‌پردازش است، سپس چندین بار از تبدیل استفاده شود.

همانند تحلیل تفکیک خطی، ما باید تعداد ویژگی­هایی(مولفه­ها) را که می‌خواهیم در خروجی بگیریم، مشخص کنیم. این کار با پارامتر n\_components انجام می­شود. یک سوال طبیعی که پیش می­آید این است که: تعداد بهینه اجزا چقدر است؟ یک استراتژی برای این کار این است که n\_components را به عنوان یک پارامتر بیشتر برای بهینه سازی در طول انتخاب مدل در نظر بگیریم (یعنی مقداری برای n\_components انتخاب کنیم که بهترین مدل آموزش‌دیده را تولید می‌کند). به همین ترتیب، چون TSVD نسبت واریانس ماتریس ویژگی اصلی را که توسط هر مولفه بیان می­شود، به ما می‌دهد می‌توانیم تعداد اجزایی را انتخاب کنیم که مقدار مورد نظر واریانس را بیان میکنند (95 % یا 99 % مقادیر مشترک هستند). برای مثال، در راه حل ما سه مولفه خروجی اول تقریبا %30 واریانس داده اصلی بیان می­شود:





ما می‌توانیم این فرآیند را با ایجاد تابعی که TSVD را با n\_componentsروی تعداد یکی کم‌تر از تعداد ویژگی­های اصلی اجرا می­کند، خودکار کنیم و سپس تعداد مولفه­هایی را محاسبه کنیم که مقدار مطلوب واریانس داده اصلی را بیان می­کند:





**برای مطالعه بیشتر**

[scikit-learn documentation TruncatedSVD](https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.TruncatedSVD.html)