دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)
دانشکده مهندسی کامپیوتر و فناوری اطلاعات

پایان‌نامه کارشناسی

گرایش نرم‌افزار

عنوان پایان نامه

شناسایی اعداد دست‌نویس با استفاده از خوشه‌بندی طیفی در مقیاس بالا

نگارش

زهرا دهقانیان

استاد راهنما

دکتر مریم امیرمزلقانی

شهریور ماه سال ۱۳۹۸

دانشگاه صنعتی امیرکبیر
(پلی تکنیک تهران)
دانشکده مهندسی کامپیوتر و فناوری اطلاعات

پایان‌نامه کارشناسی

گرایش نرم‌افزار

عنوان پایان نامه

شناسایی اعداد دست‌نویس با استفاده از خوشه‌بندی طیفی در مقیاس بالا

نگارش

زهرا دهقانیان

استاد راهنما

دکتر مریم امیرمزلقانی

شهریور ماه سال ۱۳۹۸

۲۱۲۷۵۵۲
اول ۹۰

بسمه تعالی

فرم تعریف پروژه

فارغ التحصیلی دوره کارشناسی

دانشکده مهندسی

کامپیوتر و فناوری اطلاعات

دانشگاه صنعتی امیرکبیر

(پلی‌تکنیک تهران)

تاریخ: .....

شماره: .....

عنوان پروژه: پیاده‌سازی سیستم شناسایی اعداد دست‌نویس با استفاده از خوشه بندی طیفی در مقیاس بالا

استاد راهنمای پروژه: دکتر مریم امیرمزلقانی

امضاء

مشخصات دانشجو:

نام و نام خانوادگی: زهرا دهقانیان

گرایش: نرم افزار

شماره دانشجوئی: ۹۴۳۱۰۳۹

ترم ثبت نام پروژه: ترم ۷

داوران پروژه:

1 ـ دکتر سلیمان فلاح

امضاء داور:

2 ـ دکتر چهرقانی

امضاء داور:

۱۰/۱۰/۹۸

به نام خدا

تعهدنامه اصالت اثر

تاریخ:

اینجانب زهرا دهقانیان متعهد می‌شوم که مطالب مندرج در این پایان نامه حاصل کار پژوهشی اینجانب تحت نظارت و راهنمایی اساتید دانشگاه صنعتی امیرکبیر بوده و به دستاوردهای دیگران که در این پژوهش از آنها استفاده شده است مطابق مقررات و روال متعارف ارجاع و در فهرست منابع و مآخذ ذکر گردیده است. این پایان نامه قبلاً برای احراز هیچ مدرک هم‌سطح یا بالاتر ارائه نگردیده است.

در صورت اثبات تخلف در هر زمان، مدرک تحصیلی صادر شده توسط دانشگاه از درجه اعتبار ساقط بوده و دانشگاه حق پیگیری قانونی خواهد داشت.

کلیه نتایج و حقوق حاصل از این پایان نامه متعلق به دانشگاه صنعتی امیرکبیر می‌باشد. هرگونه استفاده از نتایج علمی و عملی، واگذاری اطلاعات به دیگران یا چاپ و تکثیر، نسخه‌برداری، ترجمه و اقتباس از این پایان نامه بدون موافقت کتبی دانشگاه صنعتی امیرکبیر ممنوع است. نقل مطالب با ذکر مآخذ بلامانع است.

زهرا دهقانیان

امضا

تقدیم به همسر

و خانواده عزیزم

که همیشه برای لحظات شاد زندگی‌ام مدیون‌شان هستم.

تقدیر و تشکر

سپاس مخصوص خداوند مهربان که به انسان توانایی و دانایی بخشید تا به بندگانش شفقت ورزد. با سپاس از استاد محترم دکتر مریم امیرمزلقانی به جهت سعه‌صدر و راهنمایی‌هایشان در روند تحقیق و تمامی دوستانی که وجودشان همواره مایه‌ی دلگرمی بوده است.

چکیده

امروزه بازشناسی اعداددست‌نویس در کاربردهای مختلف نظیر تشخیص خودکار مبلغ چک‌های بانکی و سیستم‌های پستی رایانه‌ای، از اهمیت بسیار بالایی برخوردار است. تشابه شکل نوشته‌ها، روی هم افتادگی، اتصالات داخلی در حروف مجاور، از چالش‌های بسیار مهم در این زمینه است. . روش‌های موجود به طور کلی به دو دسته تقسیم می‌شوند؛ دسته اول فرض بر وجود برچسب[1] برای داده‌های اصلی دارند و به طبقه‌بندی داده‌ها بر این اساس می‌پردازند. دسته دیگر می‌توانند با داده‌های بدون برچسب کار کنند و به خوشه‌بندی داده‌ها می‌پردازند. در این پروژه روش خوشه‌بندی برای تشخیص اعداد دست‌نویس بکار گرفته می‌شود. در دهه‌های گذشته، خوشه‌بندی طیفی[2] به یکی از مؤثرترین رویکردهای خوشه‌بندی تبدیل شده است. یکی از اشکالات قابل توجه خوشه‌بندی طیفی، محاسبات سنگین آن است. تلاش‌های زیادی برای تسریع الگوریتم‌های خوشه‌بندی طیفی انجام گرفته و نتایج خوبی بدست آمده است. اما بسیاری از الگوریتم-های موجود به این فرض متکی هستند که می‌توان داده‌ها را به طور کامل در حافظه رایانه ذخیره و پردازش کرد. وقتی همه داده‌ها در حافظه رایانه جای نگیرند، این الگوریتم‌ها کارایی بالایی نخواهند داشت. برای حل این چالش، یک الگوریتم خوشه‌بندی طیفی خطی جدید با منابع محاسباتی محدود، مثل حافظه بکار گرفتیم. این الگوریتم از یک روش مؤثر برای تقریب ماتریس ارتباط گراف از طریق اعمال یک گراف دو قطبی استفاده می‌کند. در این‌جا برای جلوگیری از دسترسی تصادفی به داده‌ها، یک روش ساخت و بهینه‌سازی گراف هوشمند بکار گرفته شده‌است که منجر به یک الگوریتم کارآمد خوشه‌بندی طیفی می‌شود که میزان حافظه آن مستقل از تعداد نقاط داده ورودی است. آزمایش‌های انجام شده بر روی مجموعه داده‌های بزرگ نشان می‌دهد که الگوریتم خوشه‌بندی طیفی خطی بسیار سریع‌تر از سایر روش-های خوشه‌بندی می‌باشد.

واژه‌های کلیدی:

خوشه‌بندی طیفی، اعداد دست‌نویس، بازشناسایی تصاویر، خوشه‌بندی طیفی خطی، بهینه‌سازی‌گراف

[1] Label
[2] Spectral Clustering

فهرست مطالب

صفحه

فهرست اشکال

صفحه

فهرست جداول

صفحه

فصل اول

مقدمه

مقدمه

تشخیص حروف و کلمات تایپ شده ودست‌نویس به کمک رایانه[1] مدت‌هاست که ذهن محققین هوش مصنوعی را به خود مشغول کرده است. تاکنون در این زمینه، تلاش‌های زیادی صورت گرفته و پیشرفت زیادی حاصل شده است[1]. روش‌های کلاسیک در تشخیص الگوها را می‌توان به دو دسته‌ی روش‌های ساختاری و روش‌های آماری تقسیم کرد [2]. در روش‌های ساختاری برای تشخیص الگو از خواص هندسی و توپولوژیکی نظیر قوس‌ها، حلقه‌ها، زوایا و امتدادهای موجود در شکل استفاده می‌شود. در روش‌های آماری به هر الگو یک بردار در یک فضای برداری، نسبت داده می‌شود. این فضای برداری را فضای ویژگی[2] الگو می‌نامند و مولفه‌های بردار ویژگی الگو با استفاده از اندازه‌گیری‌های متعدد که روی الگو انجام می‌شود، به‌دست می‌آیند. فضای ویژگی الگو با استفاده از اطلاعاتی که قبلاً در اختیار سیستم تشخیص الگو قرار داده شده، به طبقات و نواحی متعددی تقسیم می‌شوند. هدف از تشخیص یک الگو تعیین طبقه یا ناحیه‌ای است که بردار الگو در آن قرار دارد. به این عمل، طبقه‌بندی[3] و به سیستم تشخیص الگو، طبقه‌بندی‌کننده[4] می‌گویند. اطلاعات لازم جهت تقسیم‌بندی فضای ویژگی الگوها شامل توابع چگالی احتمال شرطی و غیر شرطی و مشترک طبقات و الگوهاست. اما این توابع همواره موجود نیستند و تنها اطلاعاتی که از طبقات و الگوها در دسترس است تعدادی زوج متناظر به‌صورت طبقه یا الگو است. در روش‌های آماری سعی می‌شود با استفاده از این اطلاعات توابع چگالی احتمال لازم تقریب زده شود، اما تقریب توابع چگالی شرطی کار مشکلی است؛ خصوصاً اگر ابعاد فضای ویژگی بزرگ باشد.

دسته‌بندی دیگر روش‌های داده‌کاوی بر اساس نوع داده می‌باشد؛ یک دسته از داده‌های برچسب‌دار استفاده می‌کنند، بدین معنی که در حین اجرای الگوریتم نیاز به داشتن برچسب درست تعدادی از داده‌ها، دارند. دسته دوم بدون نیاز به برچسب داده‌ها، به کشف روابط میان داده‌های اصلی می‌پردازند، که روش‌های خوشه‌بندی نام دارند. در این شاخه، شباهت و تفاوت میان داده‌ها، مورد بررسی می‌باشد و تشخیص برچسب داده‌ها، مورد بحث نیست[3]. خوشه‌بندی بخصوص خوشه‌بندی طیفی، یکی از مهم‌ترین روش‌ها در بینایی رایانه و تشخیص الگو است. اما به دلیل پردازش سنگین نیاز به توسعه دارد. در دهه‌های گذشته، روش‌های خوشه‌بندی طیفی بر مبنای تجزیه مقادیر ویژه[5] در گراف لاپلاسی از داده‌ها، عملکرد برتر نسبت به روش های سنتی مانند K-Means نشان داده‌اند[6-4]. اثربخشی روش‌های خوشه‌بندی طیفی، عمدتاً به دلیل

1 Optical Character Recogniation
2 Feature Space
3 Classification
4 Classifier
5 Eigendecompositions

توانایی بالا در ایجاد ساختار خوشه‌های غیرخطی است، در حالی که بسیاری از روش‌های خوشه‌بندی دیگر، به هندسه اقلیدسی و فرضیات صریح یا ضمنی شکل‌خوشه‌ها تکیه دارند[7]. اما با همه‌ی این مزیت‌ها، الگوریتم‌های خوشه‌بندی طیفی معمولی به دلیل پیچیدگی محاسباتی از مرتبه درجه سوم برای داده‌های در مقیاس بسیار بزرگ کارآمد نیستند[8]. درواقع، با توجه به رشد سریع داده‌های ایجاد شده توسط کاربر، چنین پیچیدگی عملاً باعث می‌شود خوشه‌بندی طیفی در داده‌های بزرگ که در دنیای واقعی وجود دارد، غیرقابل استفاده باشد.

روش‌های زیادی برای کاهش هزینه محاسباتی خوشه‌بندی طیفی پیشنهاد شده است. تقریباً می‌توان آن‌ها را به دو دسته اساسی تقسیم کرد. الگوریتم های دسته اول بر کاهش هزینه محاسباتی در تجزیه مقادیر ویژه تمرکز دارند. اگر یک ماتریس شباهت n نقطه‌ای داشته باشیم، این‌الگوریتم‌ها تجزیه مقادیر ویژه را با رویکردهای تقریبی، تسریع می‌کنند[11-8]. اگرچه نتایج امیدوار کننده‌ای از این روش‌ها بدست آمده است، اما ساخت ماتریس شباهت، نیاز به زمان اجرا از مرتبه درجه دو دارد، که استفاده از آنها را محدود می‌کند. الگوریتم‌های دسته دوم، با تقریب گراف اصلی، هزینه محاسباتی ساخت گراف و تجزیه مقادیر ویژه را به‌طور هم‌زمان کاهش می‌دهند[13-12]. به طور کلی، الگوریتم‌های دسته دوم نسبت به دسته اول کاربردی‌تر هستند؛ زیرا هزینه تمام مراحل خوشه‌بندی طیفی به طور همزمان کاهش می‌دهند. اما این الگوریتم‌ها فرض می‌کنند که کلیه داده‌ها و نتایج میانی در حافظه اصلی قابل دسترسی هستند. با این فرض، به طور قابل توجهی مقیاس داده‌های‌پردازشی محدود می‌شود؛ این یک چالش پژوهشی جدی است که بتوان به طور کارآمد داده‌ها را فراتر از ظرفیت حافظه خوشه‌بندی کرد. به طور کلی می‌توان زمان مورد نیاز برای اجرای الگوریتم‌ها را به دو قسمت تقسیم کرد:

۱) زمان پردازش داده ها در حافظه.

۲) زمان انتقال داده‌ها بین حافظه و دیسک سخت

اکثر الگوریتم‌های خوشه‌بندی طیفی قبلی در مقیاس بزرگ فرض می‌کنند که زمان انتقال داده‌ها به حافظه ناچیز است که باعث می‌شود بکارگیری آن‌ها برای داده‌های بزرگ عملاً ناکارآمد باشد. دسته‌دیگر سعی دارند با پیاده‌سازی الگوریتم در سیستم‌های توزیع شده، زمان انتقال داده‌ها بین حافظه و دیسک سخت را کاهش دهند[15-14]. با این وجود علاوه بر این که برنامه‌نویسی بر روی سیستم توزیع شده یک کار چالش برانگیز است، استقرار و تنظیم یک برنامه توزیع شده نیز به مهارت‌های تخصصی نیاز دارد.

هدف این پروژه پیاده‌سازی الگوریتم‌های خوشه‌بندی طیفی در مقیاس بزرگ برای کاربران عادیست که دارای منابع محاسباتی محدود هستند. در اینجا فرض بر این است که داده‌ها به طور کامل در حافظه ذخیره

نمی‌شود. بنابراین، مبادله داده بین حافظه و دیسک، امری اجتناب ناپذیر است. ایده اصلی این رویکرد تقسیم داده‌های موجود در بلوک‌های کوچک و پردازش خطی هر بلوک است که به حافظه کم و حجم محاسباتی پایین نیاز دارد[17-16]. الگوریتم سریع و مقیاس‌پذیر خوشه‌بندی طیفی خطی[1]، گراف شباهت اصلی را با یک گراف دو بخشی[2] تقریب می‌زند و از دو مؤلفه اصلی برای کاهش زمان محاسباتی و زمان انتقال داده استفاده می‌کند. مؤلفه اول یک چارچوب الگوریتم K-Means خطی است و مؤلفه دوم یک رویکرد تجزیه مقادیر ویژه خطی است. این الگوریتم قادر است بدون کاهش دقت خوشه‌بندی، در زمان نزدیک به زمان خطی، با توجه به تعداد نقاط داده اجرا شود. این کار نیاز به بررسی داده‌های اصلی روی دیسک را محدود می‌کند. در این خصوص، آزمایش‌هایی در محیط واقعی و بدون حافظه کافی برای داده-های‌بزرگ انجام گرفته است و می‌توان گفت این الگوریتم تا چندین مرتبه از رویکردهای پیشرفته دیگر سریع‌تر است[8].

در ادامه این پروژه موارد زیر را به ترتیب خواهیم داشت؛ ابتدا در فصل دوم، مفاهیم پایه‌ی موردنیاز برای ورود به بحث بیان می‌شود. در فصل سوم، مباحث نظری و توضیح دقیق الگوریتم به‌صورت مرحله به مرحله ارائه می‌گردد. در ادامه در فصل چهارم، نحوه پیاده‌سازی بخش‌های مختلف الگوریتم با زبان برنامه‌نویسی پایتون[3] توضیح داده می‌شود. در فصل پنجم به ارزیابی نتایج و ارایه گزارشات اجراها می‌پردازیم و در نهایت، در فصل ششم جمع‌بندی و بررسی کارهای‌آینده ارایه می‌گردد.

[1] Sequential Spectral Clustering
[2] Bipartite
[3] Python

فصل دوم

مفاهیم پایه

مفاهیم پایه

امروزه با توجه به در اختیار داشتن حجم بالای اطلاعات در زمینه‌های مختلف و در دسترس بودن ترکیبی از داده‌های مفید و غیر مفید، لزوم استفاده از روش‌های خاص جهت استخراج اطلاعات مفید از داخل حجم انبوهی از داده‌ها به خوبی احساس می‌گردد. یکی از روش‌هایی که طی سالیان اخیر جهت انجام این امر استفاده می‌گردد، روش‌های داده‌کاوی است. مدل‌ها متنوعی از فرآیند های‌داده‌کاوی وجود دارد که از پرکاربردترین آنها می‌توان به دسته‌بندی و خوشه‌بندی اشاره کرد.

خوشه‌بندی یکی از روش‌های بسیار قدرتمند برای کشف گروه‌ها و وابستگی‌های طبیعی در یک مجموعه داده است. هم‌چنین یک روش برای شناخت الگوهای ساختاری و موضوعی موجود در مجموعه داده، بدون داشتن هرگونه پیش‌زمینه‌ی شناختی در مورد مشخصات و ویژگی‌های داده است. خوشه‌بندی اسناد، به‌عنوان یکی از روش‌های یادگیری ماشین بدون ناظر[1]، در زمینه‌های مختلف پردازش زبان‌های طبیعی از قبیل بازیابی اطلاعات و خلاصه‌سازی سند متنی خودکار کاربرد گسترده‌ای دارد. اساس خوشه‌بندی در اسناد دسته‌بندی سندهایی است که با یکدیگر شباهت دارند. در واقع خوشه‌بندی اسناد به روشی گفته می‌شود که یک مجموعه از اسناد را به‌صورت خودکار به چند مجموعه‌ی کوچک‌تر از اسناد مشابه تقسیم می‌کند؛ به گونه‌ای که اسناد موجود در یک خوشه از لحاظ موضوعی و یا مفهومی به‌یکدیگر شباهت داشته‌باشند.

معمولا در خوشه‌بندی مستندات متنی، با ابعاد بسیار بالای فضای داده مواجه هستیم که خوشه‌بندی طیفی روشی موثر در این نوع خاص داده است. خوشه‌بندی طیفی یک روش کاهش ابعاد محسوب می‌شود. ایده‌ی کلی در این روش ساخت ماتریس ابعاد جدید با استفاده از یک گراف همسایگی است. خوشه‌بندی داده‌ها با ساختار گراف و شبکه، کاربردهای فراوانی مانند تحلیل شبکه‌های اجتماعی دارند. در این نوع خوشه‌بندی با چالش‌هایی نظیر چگونگی اندازه‌گیری شباهت میان اشیاء موجود در گراف و چگونگی طراحی مدل‌ها و روش‌هایی برای خوشه‌بندی آنها روبرو هستیم. در سال‌های اخیر خوشه‌بندی طیفی برای تجزیه و تحلیل داده‌های بزرگ مورد توجه قرار گرفته است. در ادامه به بررسی الگوریتم‌های پایه‌ی موردنیاز همچون K-Meas، K-Mean++، SVD و خوشه‌بندی طیفی می‌پردازیم.

[1] Unsupervised

۲-۱- الگوریتم K-Means

در سال‌های اخیر، اهمیت خوشه‌بندی داده‌ها در علوم مختلف و همچنین تفاوت داده‌های آماری استفاده شده در آنها از نظر نوع و توزیع، باعث ابداع روش‌های متنوعی از خوشه‌بندی داده‌ها شده است. خوشه‌بندی یک تکنیک دسته‌بندی بدون ناظر است که در آن مجموعه داده‌ها که معمولاً بردارهای فضایی چند بعدی می‌باشند، براساس یک معیار شباهت، به تعدادی خوشه تقسیم می‌شوند. هریک از این داده‌های تخصیص داده شده به یک خوشه، نسبت به داده‌هایی که در خوشه‌های دیگرند، بر اساس یک معیار مشخص، شبیه‌تر هستند[18]. به‌طور دقیق‌تر می‌توان گفت، خوشه‌بندی در فضای d بعدی اقلیدسی فرآیندی است که یک مجموعه N عضوی به k گروه یا خوشه براساس یک معیار شباهت تقسیم می‌شود، چنانچه مجموعه S در برگیرنده نقاط $S = \{X_1, X_2, \dots, X_N\}$ باشد و خوشه‌ها با $C_1, C_2, \dots, C_k$ نشان داده شود، آنگاه داریم:

$$\{C_i \cap C_j = 0 \; for \; i,j = 1,2,\dots,k \; \& \; i \neq j\}$$

$$U_{i=1}^{k} c_i = S$$

الگوریتم K-Means یکی از معروف‌ترین و پراستفاده‌ترین تکنیک‌های خوشه‌بندی است که در بسیاری از مسایل به کار رفته است. این الگوریتم با k مرکز خوشه تصادفی شروع می‌شود و مجموعه از اشیاء را به k زیر مجموعه تقسیم می‌کند. این روش، بردارهای داده در فضای d بعدی را در خوشه‌هایی که از قبل تعدادشان مشخص است، دسته‌بندی می‌کند. اساس دسته‌بندی در K-Means فاصله اقلیدسی بین داده‌ها و مراکز خوشه است که به عنوان معیار شباهت در نظر گرفته می‌شود. فاصله اقلیدسی بین داده‌های عضو یک خوشه با مرکز آن خوشه، نسبت به فاصله همین داده‌ها با سایر مراکز خوشه کمتر است. الگوریتم K-Means استاندارد به‌صورت زیر عمل می‌کند:

در ابتدا موقعیت اولیه k مرکز خوشه به‌صورت تصادفی مشخص می‌شود. سپس مراحل زیر تکرار می‌شود:

۱- برای هر داده: بردار داده به خوشه‌هایی تخصیص می‌یابد که فاصله آن نسبت به مراکز خوشه‌های دیگر کمتر باشد. فاصله تا مرکز خوشه براساس رابطه زیر محاسبه می‌شود:

$$Dis(X_p, Z_j) = \sqrt{\sum_{i=1}^{d} = (X_{pi} - Z_{ji})} \qquad (1-2)$$

در رابطه (۲-۱) X_p نشان دهنده بردار داده pام است و Z_j مشخص‌کننده مرکز خوشه jام می‌باشد و d تعداد خصوصیات داده‌ها و مراکز خوشه‌ها می‌باشد.

۲- پس از اینکه تمام داده‌ها به خوشه‌ها تخصیص داده شد، هر یک از مراکز خوشه با استفاده از رابطه (۲-۲) بروزرسانی می‌شود.

$$Z_j = \frac{1}{n_j}\left[\sum_{\forall X_p \in C_j} X_p\right] \qquad (2-2)$$

در این رابطه n_j تعداد بردارهای عضو خوشه jام است. C زیرمجموعه‌ای از کل بردار داده‌های عضو خوشه jام می‌باشند. مرکز خوشه بدست آمده از رابطه فوق برابر میانگین بردارهای داده تشکیل دهنده خوشه است.

مراحل ۱و۲ تا هنگامیکه که شرط توقف برقرار گردد، تکرار می‌شود. شرط توقف می‌تواند یکی از موارد روبرو باشد: تعداد تکرار مشخص، رسیدن به تغییرات اندک در مراکز خوشه‌ها و یا عدم تغییر خوشه‌ی تخصیص داده‌شده به داده‌ها باشد[19].

این روش به سادگی قابل فهم و پیاده‌سازی است و پیچیدگی زمانی آن خطی می‌باشد. اما با همه‌ی این مزیت‌ها، دارای چند ضعف اساسی می‌باشد؛ یکی از مشکلات و ضعف‌های این روش حساس بودن بیش از حد آن به مقادیر اولیه مراکز خوشه هاست. توضیح بیشتر این‌که، تابع هدف روش K-Means دارای بهینه-های محلی متعددی می‌باشد و روش K-Means توانایی آن را ندارد که عبور از بهینه‌های محلی را تضمین کند. یعنی در صورتی که مکان اولیه خوشه در فضای مسئله نامناسب انتخاب شده باشد، این روش به سرعت به یک بهینه محلی همگرا می‌شود و مسئله در همین بهینه محلی گیر می‌کند.

۲-۲- الگوریتم K-Means++

همان‌طور که گفتیم، روش K-Means یک تکنیک خوشه‌بندی است که به طور گسترده مورد استفاده قرار می‌گیرد و در صدد به حداقل رساندن فاصله متوسط بین نقاط در هر خوشه است. این مسئله از مسایل دسته NP-Hard است و این بدین معنیست که هیچ تضمینی برای حداقل دقت ارائه نمی‌دهد، اما سادگی و سرعت آن در عمل بسیار جذاب است. لوید وهمکاران[20] یک راه حل محلی برای این مشکل ارائه دادند که امروزه هم مورد استفاده قرار می‌گیرد. آن‌ها با بهره‌گیری از الگوریتم K-Means به همراه یک مقدار اولیه تصادفی، الگوریتمی به دست آورد که زمان خوشه‌بندی آن از مرتبه $O(\log k)$ می‌باشد. مراحل الگوریتم K-Means++ به شرح زیر می‌باشد:ابتدا اولین نقطه به‌صورت تصادفی یکنواخت از مجموعه داده X انتخاب می‌شود و در ادامه:

۱. تعیین مرکز خوشه جدید c_i، انتخاب $x \in X$ با احتمال $\frac{D(x)^2}{\sum_{x \in X} D(x)^2}$، در این رابطه $D(x)$ به معنی کمترین فاصله یک داده تا مرکز خوشه‌هایی که قبلاً انتخاب شده را نشان می‌دهد. انتخاب مرکز خوشه با این احتمال، منجر به پراکندگی مراکز خوشه‌ها می‌شود. زیرا مراکز خوشه دورتر با احتمال بیشتری انتخاب می‌شوند.

۲. تکرار مرحله قبلی تا انتخاب کل مراکز خوشه

پس از انتخاب k مرکز خوشه‌ی اولیه، الگوریتم مطابق الگوریتم استاندارد K-Means پیش می‌رود.

۲-۳- الگوریتم SVD

دردادهکاوی، از خوشهبندی و دستهبندی به سان ابزارهایی جهت توصیف دادهها و مدل کردن پیشبینی آنها استفاده میشود. اساس مشترک راهحلهای مختلف در تحلیل دادهها پیدا کردن یک مدل مناسب برای نمایش آنها است تا بینش و تصوری صحیح از آنها بهوجود آید. راهحل بسیار معمولی که آن مدل کاهشی مینامیم، مدلی است که سعی میکند از پیچیدگی دادههای اولیه و با بعد بالا بکاهد و ضمن حفظ ملزومات مسئله همراه با دادهها، ساختار پنهان آنها را آشکار کند. در واقع، این مدل ضمن کاهش دادهها، در سطحی نزدیکتر (واقعیتر) به سیستم اولیه قرار دهد. یکی از انواع مدلهای کاهشی خطی، روش تجزیه مقادیر تکین میباشد.

در بسیاری از کاربردها، یک ماتریس درجه پایین نزدیک به ماتریس اصلی داده A، یک پیش بینی مناسب برای ماتریس دادهها است. به عبارتی، از تجزیه مقادیر تکین A میتوان ماتریس دیگری از رتبه[1] p بدست آورد که بهترین تقریب A باشد. به طور خلاصه میتوان گفت که تجزیه مقادیر تکین یک ماتریس را به سه ماتریس دیگر تجزیه میکند

$$A = USV^T \qquad (3-2)$$

که در آن A یک ماتریس $m \times n$، U یک ماتریس متعامد $m \times m$ ، S یک ماتریس قطری $m \times n$ و V یک ماتریس متعامد $n \times n$ است. رابطه ماتریس (۲-۳) را میتوان بهصورت سری زیر نوشت:

$$a_{ij} \approx \sum_{k=1}^{p} U_{ik} S_k V_{jk} \ , \mathrm{p} < \mathrm{n} \qquad (4-2)$$

متغیرهای $\{S_i\}$ مقادیر تکین یا تکین نام دارند و معمولاً از بزرگترین به کوچکترین نوشته میشود. ستونهای U ، بردارهای تکین چپ و ستونهای V، بردارهای تکین راست نامیده میشود[21] .

در واقع ابعاد ماتریسهای تجزیه شده به اندازه p کاهش یافتند. U یک ماتریس متعامد $m \times p$ ، S یک ماتریس قطری $p \times p$ و V یک ماتریس متعامد $p \times n$ است. البته برای انجام عملیات ماتریسی، اختلاف $n - p$ با صفر پر میکنیم[21].

[1] Rank

۲-۴- خوشه‌بندی طیفی

معمولاً در خوشه‌بندی مستندات متنی با مسئله ابعاد بالا مواجه هستیم. در روش خوشه‌بندی طیفی بر خلاف روش‌های کلاسیک، داده‌ها ابتدا به فضایی با ابعاد کمتر نگاشت می‌شوند، سپس در فضای جدید، از روش‌های پیشین استفاده می‌شود. اصلی‌ترین مشکل روش‌های مانند K-Means این است که در برخورد با فضاهای با ابعاد بالا، به علت بزرگ شدن فضای جستجو، قادر به تضمین دست‌یابی به بیشینه سراسری نیستند و در بسیاری از موارد در یک بیشینه‌ی محلی متوقف می‌شوند.

ایده‌ی اصلی خوشه‌بندی طیفی، بردن نقاط داده به فضایی متشکل از بردارهای ویژه‌ی عمود بر هم و استفاده از این بردارهای ویژه برای کاهش ابعاد فضا و در نهایت استفاده از یکی از روش‌های خوشه‌بندی کلاسیک، برای خوشه‌بندی کردن داده‌ها در فضای جدید با ابعاد کمتر است. برای این منظور، ماتریس فاصله، ماتریس ارتباط و ماتریس لاپلاسین[1] ایجاد می‌شود. در اینجا ابتدا ابعاد جدید با استفاده از ماتریس همسایگی محاسبه می‌شود و پس از ساختن ماتریس لاپلاسین، مقادیر ویژه بدست می‌آید. مراحل مختلف الگوریتم خوشه‌بندی طیفی بطور کلی شامل موارد زیر می‌باشد:

۱. ساخت ماتریس فاصله
۲. ساخت ماتریس ارتباط
۳. قطری کردن ماتریس ارتباط
۴. به دست آورن بردارها و مقادیر ویژه ماتریس لاپلاسین
۵. انتخاب K بزرگترین مقدار ویژه و چینش ستونی بردارهای ویژه متناظر با این مقادیر ویژه در ماتریس X
۶. یکه‌کردن[2] سطرهای ماتریس X
۷. خوشه بندی داده ها درفضای ابعاد کاهش‌یافته

در ادامه به بررسی دقیق‌تر پیش‌نیازها و مراحل این الگوریتم می‌پردازیم.

[1] Laplacian
[2] Normalize

۲-۴-۱- ماتریس‌ها و گراف‌ها

فرض می‌کنیم گراف $G(V,E)$ را داریم که V رئوس گراف و E یال‌های گراف را مشخص می‌کند. w_{ij} وزن یال‌های گراف است. هدف این است که در یک گراف، رئوس را با در نظر گرفتن یال‌ها و وزن آن‌ها خوشه‌بندی کنیم.

در این تعریف مجموعه داده $D = \{X_i\}_{i=1}^{n}$ شامل n نقطه در فضای $\mathbb{R}^d$ است، ماتریس A، ماتریس شباهت $n \times n$ متقارن بین نقاط می‌باشد.

$$A=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} \qquad (5-2)$$

$A(i,j) = a_{ij}$ نشان دهنده تشابه و یا ارتباط بین نقطه x_i و x_j است. مقادیر شباهت در ماتریسA غیر منفی و متقارن می‌باشد، یعنی داریم $a_{ij} \geq 0$ و $a_{ij} = a_{ji}$. همچنین می‌توان ماتریسA را، یک ماتریس مجاورت وزنی در گراف بدون جهت $G(V,E)$ درنظر گرفت به‌طوری که هر راس، یک نقطه و هر یال یک جفت نقطه را طبق رابطه (۲-۸) پیوند دهد:

$$V = \{x_i |\ i = 1, \dots, .\}\ \ , E = \{(x_i, x_j) |\ 1 \leq i,j \leq n\} \qquad (6-2)$$

بنابراین در ماتریس A میزان تشابه، وزن هر یال می‌باشد؛ یعنی a_{ij} وزن یال(x_i, x_j) را مشخص می کند. اگر تمام وابستگی ها صفر یا یک باشد، A نشان دهنده ماتریس ارتباط[1] بین رئوس است. درجه راس d_i برای هر راس x_i، به‌صورت $d_i = \sum_{j=1}^{n} a_{ij}$ تعریف می‌شود. ماتریس قطری درجه deg(A) به‌صورت رابطه (۲-۹) می‌باشد.

$$\deg(A) = \Delta=\begin{pmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n \end{pmatrix} \qquad (7-2)$$

وزن یال بین x_i و x_j را معمولاً بوسیله تابع گوسی[2] طبق رابطه (۲-۱۰) محاسبه می‌شود:

$$w_{ij} = \begin{cases} \exp\left(-\dfrac{\|x_i - x_j\|^2}{2\sigma^2}\right) & \text{اگر } x_j \text{ و} x_i \text{ متصل باشند} \\ 0\ , & \text{در غیر این صورت} \end{cases} \qquad (8-2)$$

که در اینجا σ پارامتر انتشار می‌باشد و متناسب با فضای مسئله تعیین می‌گردد.

[1] Adjency Matrix

[2] Gaussian kernel

پس از محاسبه ماتریس درجه، ماتریس لاپلاسین گراف را که یک ماتریس متقارن و متعامد است را از رابطه (۲-۱۱) محاسبه می‌کنیم.

$$L = \Delta - A = \begin{pmatrix} \sum_{j=1}^{n} a_{1j} & 0 & \cdots & 0 \\ 0 & \sum_{j=1}^{n} a_{2j} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sum_{j=1}^{n} a_{nj} \end{pmatrix} - \begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \vdots & \vdots & \cdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{pmatrix} =$$

$$\begin{pmatrix} \sum_{j=1}^{n} a_{1j} & -a_{12} & \cdots & -a_{1n} \\ -a_{21} & \sum_{j\neq 2}^{n} a_{2j} & \cdots & -a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ -a_{n1} & -a_{n2} & \cdots & \sum_{j\neq n}^{n} a_{nj} \end{pmatrix} \qquad (9-2)$$

قابل توجه است که ماتریس لاپلاسین دارای مقادیر ویژه غیر منفی حقیقی است که می‌توان به ترتیب نزولی مرتب کرد: $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$

۲-۴-۲- مفاهیم برش[1] گراف

برش k-way در یک گراف عبارت است از تقسیم‌بندی یا خوشه‌بندی مجموعه راس‌ها است به‌طوری‌که اگر طبق تعریف داشته باشیم مجموعه خوشه‌ها باید شروط زیر را دارا باشد: $C = \{C_1, \dots, C_k\}$

۱- $C_i \neq \emptyset$ برای هر i

۲- $V = \cup_i C_i$ برای هر i

۳- $C_i \cap C_j \neq \emptyset$ برای هر i, j

بر روی مجموعه C با شرایط بالا، می‌توان یک تابع هدف برای مقدار دهی اولیه خوشه‌ها، بهینه‌سازی کرد. به نحوی‌که، اعضای داخل یک خوشه دارای بیشترین شباهت و اعضای خوشه‌های مختلف کمترین شباهت داشته باشند.

فرض می‌کنیم برای هر دو زیر مجموعه رئوس $S, T \subseteq V$ تابع $W(S,T)$ را مطابق رابطه (۲-۱۲) داریم که نشان‌دهنده مجموع وزن همه یال‌هایست که یک راس‌شان در S و راس دیگری‌شان در T باشد.

$$W(S,T) = \sum_{v_i \in S} \sum_{v_j \in T} a_{ij} \qquad (10-2)$$

[1] Cut

با توجه به $S \subseteq V$ ، برش در یک گراف به عنوان یک تقسیم‌بندی V به $S \subset V$ و $\bar{S}$ تعریف می شود به‌طوری‌که $\bar{S} = V - S$ مجموعه متمم راس‌ها می‌باشد. مفهوم وزن برش به عنوان مجموع وزن در یال‌های بین راس‌ها در S و $\bar{S}$ ، با عنوان $W(S,\bar{S})$ تعریف می‌شود.

عبارت $|C_i|$ اندازه خوشه C_i می‌باشد و به معنای **تعداد** اعضای موجود درخوشه i است. همچنین حجم یک خوشه C_i به‌عنوان مجموع وزن تمام یال‌هایی با یک سمت پایان در خوشه C_i مطابق رابطه (۲-۱۳) تعریف می شود.

$$vol(C_i) = \sum_{v_j \in C_i} d_j = \sum_{v_i \in C_i} \sum_{v_r \in V} a_{ij} = W(C_i, V) \qquad (11-2)$$

فرض می‌کنیم $c_i \in \{0,1\}^n$ یک بردار که عضویت داده‌ها را در خوشه C_i ثبت می‌کند و طبق رابطه (۲-۱۴) تعریف می‌شود:

$$c_{ij} = \begin{cases} 1 & \text{اگر } v_j \in C_i \\ 0 & \text{اگر } v_j \notin C_i \end{cases} \qquad (12-2)$$

از آنجا که خوشه‌بندی صورت گرفته، یک افراز است؛ بدین معنا که دو خوشه هیچ عضو مشترکی باهم ندارند، لذا داریم $c_i^T c_j = 0$ و از طرفی می‌توانیم اندازه خوشه‌ها را طبق رابطه (۲-۱۵) بدست بیاوریم.

$$|C_i| = c_i^T c_i = \|c_i\|^2 \qquad (13-2)$$

به کمک عناصر بالا می‌توان مفهوم وزن برش را در قالب عملیات ماتریسی بیان کنیم. فرض می‌کنیم مجموع وزن همه یال‌ها با یک انتهای در C_i را داریم. این یال‌ها شامل یال‌های داخلی خوشه (با هر دو انتها در C_i) ، و همچنین یال‌های خارجی خوشه (با انتهای دیگر در یک خوشه دیگر $C_{j\neq i}$) می‌باشد.

$$vol(C_i) = W(C_i, V) = \sum_{v_r \in C_i} d_r = c_i^T \Delta c_i \qquad (14-2)$$

بدین ترتیب، مجموع وزن تمام یال‌های داخلی از رابطه (۲-۱۷) بدست خواهد آمد:

$$W(C_i, C_i) = \sum_{v_r \in C_i} \sum_{v_r \in C_i} a_{rs} = c_i^T \mathrm{A} c_i \qquad (15-2)$$

و از ترکیب دو رابطه بالا، مقدار وزن یال‌های خارجی یا همان وزن برش، طبق رابطه (۲-۱۸) بدست می‌آید[22].

$$W(C_i, \overline{C_i}) = \sum_{v_r \in C_i} \sum_{v_s \in V - C_i} a_{rs} = W(C_i, V) - W(C_i, C_i) = c_i^T \mathrm{L} c_i \qquad (16-2)$$

۲-۴-۳- تابع هدف نسبت‌برش

از توابع هدف خوشه‌بندی می‌توان، برای بهینه‌سازی مسئله برش k-way استفاده کرد. تابع هدف را برای این مسئله، تابع نسبت برش در نظر می‌گیریم. این تابع بر روی برش k-way به‌صورت زیر تعریف می‌شود:

$$\min_c j_{rc}(C) = \sum_{i=1}^{k} \frac{w(C_i, \overline{C_i})}{|C_i|} = \sum_{i=1}^{k} \frac{c_i^T L c_i}{\|C_i\|^2} \qquad (17-2)$$

تابع نسبت برش سعی می‌کند با در نظر گرفتن اندازه هر خوشه، مقدار شباهت‌ها را از یک خوشه C_i به نقاط دیگر که در خوشه $\overline{C_i}$ نیستند، به حداقل برساند. می‌توان مشاهده کرد که این تابع در دو حالت مقدار کمینه دارد؛ حداقل بودن وزن برش و همین‌طور هنگام بزرگ بودن اندازه خوشه.

بهینه‌سازی این مسئله در حالت تعریف‌شده، یک مسئله‌ی NP-Hard است[22] اما با کاهش قیود مسئله به این‌که مقادیر بردار c_i می‌تواند هر مقدار حقیقی داشته باشد، می‌توان طبق رابطه (۲-۲۰) ،تابع هدف نسبت برش را بازنویسی کرد.

$$\min_c j_{rc}(C) = \sum_{i=1}^{k} \frac{c_i^T L c_i}{\|C_i\|^2} = \sum_{i=1}^{k} \left(\frac{c_i}{\|c_i\|}\right)^T \mathrm{L} \left(\frac{c_i}{\|c_i\|}\right) = \sum_{i=1}^{k} u_i^T L u_i \qquad (18-2)$$

حال به حل مسئله فوق می‌پردازیم. بدین منظور از این بازنویسی نسبت به u_i دیفرانسیل گرفته و برابر صفر قرار می‌دهیم. از طرفی می‌دانیم $u_i^T u_i = 1$ و با تعریف λ_i برای هر خوشه C_i طبق رابطه (۲-۲۱) خواهیم داشت :

$$\frac{\partial}{\partial u_i}\left(\sum_{i=1}^{k} u_i^T L u_i + \sum_{i=1}^{k} \lambda_i (1 - u_i^T L u_i)\right) = 0$$
$$\Rightarrow 2\mathrm{L}u_i - \lambda_i u_i = 0$$
$$\Rightarrow \quad \mathrm{L}u_i = \lambda_i u_i \qquad (19-2)$$

در نتیجه عملیات های بالا و تعریف مقادیر ویژه ماتریس‌ها، می‌توان نتیجه گرفت که u_i یکی از بردار ویژه‌ها و λ_i مقدار ویژه متناظرش برای ماتریس لاپلاسین می‌باشد. بنابراین خواهیم داشت:

$$u_i^T L u_i = u_i^T \lambda_i u_i = \lambda_i \qquad (20-2)$$

با جایگذاری رابطه (۲-۲۲) در تابع هدف نسبت برش رابطه (۲-۲۰) خواهیم داشت :

$$\min_c j_{rc}(C) = u_n^T L u_n + \cdots + u_{n-k+1}^T L u_{n-k+1}$$
$$= \lambda_n + \cdots + \lambda_{n-k+1} \qquad (21-2)$$

اگر فرض کنیم مقادیر ویژه مرتب شده‌اند به طوری که $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$ باشد و کوچکترین مقدار ویژه لاپلاسین یعنی $\lambda_n = 0$ باشد، k کوچکترین مقادیر ویژه بدین صورت: $\lambda_n \leq \lambda_{n-1} \leq \lambda_{n-k+1}$ نمایش داده می‌شود و بردار ویژه های متناظر با این مقادیر ویژه نشان دهنده بردار ویژگی اعضای خوشه‌ها در حالت کاهش ابعاد یافته می‌باشد.

۲-۴-۴- الگوریتم خوشه‌بندی

به کمک مفاهیم و تعاریف صورت گرفته در بخش‌های قبل، به تعریف شبه‌کد الگوریتم خوشه‌بندی طیفی مطابق الگوریتم (۲-۱) می‌پردازیم[22]:

الگوریتم (۲-۱) : Spectral Clustering (D, k):

```
1-    Compute the similarity matrix A ∈ ℝ^{n×n}
2-    Solve Lu_i = λ_i u_i for i = n,...,n−k+1, where λ_n ≤ λ_{n−1} ≤ ··· ≤ λ_{n−k+1}
3-    U ← (u_n u_{n−1} ··· u_{n−k+1})
4-    Y ← normalize rows of U using
5-    C ← {C_1,...,C_k} via K-means on Y
```

الگوریتم (۲-۱) شبه کد خوشه‌بندی طیفی است. در اینجا فرض می‌کنیم گراف اصلی یک گراف همبند است. یک مجموعه داده D را به عنوان ورودی در نظر می‌گیرد و ماتریس تشابه را محاسبه می‌کندبا توجه به این که تابع هدف مسیله ما تابع برش نسبت می‌باشد، k کوچکترین مقادیر ویژه بردار لاپلاسین را محاسبه می‌کنیم. در مرحله بعد به یکه کردن سطرهای ماتریس $U_{n \times k}$ می‌پردازیم و در نهایت بر روی ماتریس ویژگی‌های استخراج‌شده از داده‌های اصلی، الگوریتم خوشه‌بندی K-Means اجرا می‌کنیم و خوشه مربوط به هر یک از اعضا را بدست می‌آوریم.

فصل سوم

خوشه‌بندی طیفی خطی

خوشه بندی طیفی خطی

در بخش پیشین به بررسی الگوریتم‌های مختلف خوشه‌بندی و مبانی موردنیاز برای شروع بحث پرداختیم. روش‌های کلاسیک مانند K-Means عملکردشان بر روی داده‌های با پراکندگی غیر خطی، کارآمد نبود و روش خوشه‌بندی طیفی بر روی این نوع داده عملگرد مناسبی داشتند. اما با وجود نتایج قابل قبول، زمان اجرای بالا سبب عدم بکارگیری گسترده این الگوریتم برای داده‌هاب با مقیاس بالا گردیده است.

روش‌های زیادی برای کاهش هزینه محاسباتی خوشه‌بندی طیفی پیشنهاد شده است. تقریباً می‌توان آن‌ها را به دو دسته اساسی تقسیم کرد. الگوریتم های دسته اول بر کاهش هزینه محاسباتی در تجزیه مقادیر ویژه تمرکز دارند الگوریتم‌های دسته دوم، با تقریب گراف اصلی، هزینه محاسباتی ساخت گراف و تجزیه مقادیر ویژه را به‌طور همزمان کاهش می‌دهند. به طور کلی، الگوریتم‌های دسته دوم نسبت به دسته اول کاربردی‌تر هستند. اما این الگوریتم‌ها هم فرض می‌کنند که کلیه داده‌ها و نتایج میانی در حافظه اصلی قابل دسترسی هستند. با این فرض، به طور قابل توجهی مقیاس داده‌های‌پردازشی محدود می‌شود.

در واقع زمان اجرای یک الگوریتم بطور کلی شامل زمان پردازش دراخل حافظه و زمان انتقال از حافظه جانبی مانند دیسک‌سخت به داخل حافظه جهت پردازش می‌باشد. اکثر الگوریتم‌های خوشه‌بندی طیفی قبلی در مقیاس بزرگ فرض می‌کنند که زمان انتقال داده‌ها به حافظه، ناچیز است که باعث می‌شود بکارگیری آن‌ها برای داده‌های بزرگ عملاً ناکارآمد باشد

در این بخش می‌خواهیم به توضیح الگوریتم خوشه‌بندی طیفی در مقیاس بزرگ برای کاربران عادی که دارای منابع محاسباتی محدود هستند، می‌پردازیم. این الگوریتم ابتدا گراف شباهت اصلی را با یک گراف دو طرفه تقریب می‌زند و سپس از دو مؤلفه اصلی برای کاهش زمان محاسباتی و زمان انتقال داده استفاده می‌کند. مؤلفه اول یک چارچوب الگوریتم K-Means خطی است و مؤلفه دوم یک رویکرد تجزیه مقادیر ویژه خطی است. این الگوریتم قادر است بدون کاهش دقت خوشه‌بندی، در زمان نزدیک به زمان خطی، با توجه به تعداد نقاط داده اجرا شود. در زیربخش بعدی ابتدا به تعریف نمادها و پیش‌زمینه بحث می‌پردازیم. پس از هم‌ادبیات شدن، به توضیحات در خصوص نحوه ساخت گراف شباهت می‌پردازیم. سپس به بررسی دو مولفه اصلی موجود در این الگوریتم پرداخته و در نهایت در بخش آخر، این اطلاعات را تجمیع کرده و در قالب الگوریتم نهایی ارایه می‌دهیم[8].

۳-۱- نمادگذاری و پیش‌زمینه

در این بخش به طور خلاصه نمادها و الگوریتم خوشه بندی طیفی را معرفی خواهیم کرد. فرض می‌کنیم $X = [x_1, \dots, x_n]^T \in \mathbb{R}^{n\times d}$ ماتریس داده را مشخص می‌کند، n تعداد نقاط داده و d ابعاد ویژگی‌ها است. هر نقطه داده $x_i \in \mathbb{R}^d$ به یکی از کلاس‌های $C = \{c_1, \dots, c_K\}$ تعلق دارد. در مجموعه داده X، هر نقطه داده به عنوان یک راس بر روی گراف و هر یال نشان دهنده ارتباط یک جفت رئوس است. در عمل معمولاً از گراف $\mathrm{k-NN}$ استفاده می‌شود. به طور خاص، x_i و x_j در صورتی متصل هستند که حداقل یکی از آنها در k نزدیکترین همسایگان در معیار معین (معمولاً فاصله اقلیدسی) باشد. وزن یال بین x_i و x_j به طبق رابطه (۳-۱) است:

$$w_{ij} = \begin{cases} \exp\left(-\frac{\|x_i - x_j\|}{2\sigma^2}\right) & \text{اگر } x_j \text{ و } x_i \text{ متصل باشند} \\ 0, & \text{در غیر این صورت} \end{cases} \qquad (1-3)$$

در اینجا σ پارامتر پهنای باند است. $W = \{w_{ij}\} \in \mathbb{R}^{n\times n}$ ماتریس مجاور گراف بدون جهت متقارن است. اگر $D \in \mathbb{R}^{n\times n}$ یک ماتریس درجه باشد که iامین عضو قطر برابر با $d_{ij} = \sum_{j=1}^{n} w_{ij}$ باشد. در اینجا L علامت ماتریس لاپلاسین یکه‌شده شده گراف W می‌باشد و طبق رابطه (۳-۲) تعریف می‌شود:

$$L = I - D^{-1/2} W D^{-1/2} \qquad (2-3)$$

تابع هدف خوشه‌بندی طیفی طبق رابطه (۳-۳) تعریف می‌شود[20]:

$$\min_{G^T G = I} T_r(G^T L G) \qquad (3-3)$$

در اینجا $G \in \mathbb{R}^{n\times K}$ نشانگر ماتریس تمام داده‌ها است. یک راه حل برای معادله (۳-۳) انتخاب K تا کوچکترین بردارهای ویژه مارتیس لاپلاسین است (که به تفضیل در بخش قبل به اثبات پرداختیم).

۳-۲- ساخت گراف

ساخت نمودار W به دلیل محاسبه فاصله زوج نقاط داده، از فضای زمان درجه دوم برخوردار است. این روند به راحتی قابل خطی شدن است. به طور خاص، می‌توان فقط یک نمونه از داده‌های x_i را در حافظه نگه داریم و سپس کلیه داده‌های دیگر را از دیسک بارگیری و فاصله آن را از سایر نقاط داده محاسبه کنیم. این الگوریتم معادل محاسبه خطی هر سطر از W است. با این وجود، هنوز این رویکرد دو مشکل اساسی دارد. اول اینکه، پیچیدگی زمانی هنوز از درجه دوم است. دوم اینکه، زمان مورد نیاز برای بررسی چندین باره داده‌ها، ممکن است حتی بیشتر از محاسبه خود مجموعه داده‌های بزرگ زمانبر باشد.

برای کاهش پیچیدگی زمانی، بایستی نیاز به محاسبه فاصله دوطرفه را از بین برد. یک روش معمول برای دستیابی به این هدف، یافتن یک گراف دوبخشی از درجه پایین‌تر است، که به طور تقریبی معادل گراف W می‌باشد. در این روش به جای محاسبه فاصله مستقیم بین دو نقطه x_i و x_j، مقادیر محلی u_k در همسایگی آنها ساخته و سپس فاصله بین x_i و x_j را از رابطه (۳-۴) تقریب می‌زنیم.

$$dist(x_i, x_j) \approx dist(x_i, u_i) + dist(u_i, x_j) \qquad (4-3)$$

رابطه (۳-۴) بیان می‌کند که فاصله یک نقطه تا سایر نقاط با تقریب یک نقطه نزدیک u_kحساب می‌شود. تعداد نقاط میانی باید بسیار کوچکتر از تعداد داده باشد تا مرتبه زمانی مورد نیاز برای محاسبه فاصله یک نقطه کمتر از n^2 بشود. بنابراین، به جای محاسبه W، از روش گراف نقاط کمکی استفاده می کنیم[16]. در اینجا، یک مجموعه کوچک از نقاط کمکی $U = [u_1, \dots, u_m]^T \in \mathbb{R}^{m \times d}$ برای نگه‌داری ساختار اصلی استفاده می‌شود. این نقاط میانی معمولاً با اجرای یک الگوریتم خوشه‌بندی سبک وزن مانند K-Means بر روی داده‌های خام انتخاب می‌شوند.

با نقاط تولید شده، گراف $k - NN$ بین داده‌های خام و نقاط کمکی ساخته می‌شود. اتصالات فقط بین نقاط داده خام و نقاط کمکی برقرار است. این محدودیت در گراف دو طرفه بین داده‌های خام X و نقاط کمکی U حاصل می شود. وزن هر یال طبق رابطه (۳-۵) تعریف می شود[8].

$$Z_{ij} = \frac{K(x_i, u_j)}{\sum_{k \in \phi_i} K(x_i, u_k)} \quad , \forall j \in \phi_i \qquad (5-3)$$

که در آن $K()$ یک تابع هسته معین و $\phi_i \subset \{1, \dots m\}$ ، p نزدیکترین همسایه x_i را در U نشان می دهد. ماتریس ارتباط به شکل $W = \begin{bmatrix} 0 & Z \\ Z^T & 0 \end{bmatrix} \in \mathbb{R}^{(n+m)\times(n+m)}$ تبدیل می‌شود. ماتریس درجه برابر است با $D = \begin{bmatrix} D_r & 0 \\ 0 & D_c \end{bmatrix} \in \mathbb{R}^{(n+m)\times(n+m)}$ که در آن D_r یک ماتریس قطری است که عناصر قطر آن‌ها مجموع سطر Z است و D_c یک ماتریس قطری است که عناصر قطر آن مجموع ستون‌های Z است. از آنجا که Z به‌صورت سطری نرمال تعریف شده است، داریم $D_r = I_n$ که در آن I_n ماتریس هویت n در n است. پس از محاسبه D_c به تعریف ماتریس $\hat{Z}$ طبق رابطه (۳-۶) می‌پردازیم. به کمک این تعریف، پس ما یک گراف معمولی بدون ساختار را به یک گراف دو بخشی تبدیل کرده‌ایم[8].

$$\hat{Z} = Z D_c^{-1/2} \qquad (6-3)$$

اگرچه پیچیدگی زمان محاسبات فاصله کاهش می‌یابد، ولی این رویکرد هنوز هم باید برای استفاده از K-Means برای تولید نقاط کمکی، چندین بار از داده‌ها عبور کند. این برای پردازش مجموعه داده‌های بزرگ هنوز هم مانع است. وقتی مجموعه داده از اندازه حافظه بزرگتر باشد، زمان انتقال بخش اعظم کل زمان اجرا را به خود اختصاص می‌دهد.

۳-۳- K-Means خطی

در این بخش، یک روش جدید برای ساخت گراف بکارگیری می‌شود، که دارای حجم محاسباتی کم و میزان حافظه مصرفی پایینی است. هسته اصلی این روش الگوریتم K-Means خطی است.

شبه کد الگوریتم K-Means خطی (SeqKM[1]) در الگوریتم (۳-۱) ارائه شده است. ایده اصلی SeqKM این است که ابتدا یک زیر مجموعه کوچک از داده‌ها را بطور تصادفی انتخاب کنیم و الگوریتم K-Means++ را روی آن اجرا می‌کنیم و سپس به طور خطی، کل مجموعه داده‌ها را با استفاده از الگوریتم گرادیان تصادفی کاهشی (SGD[2]) پردازش می‌کنیم[23]. K-Means++ تضمین می‌کند که مراکز اولیه از کیفیت نسبتاً خوبی برخوردار هستند و SGD بیشتر مراکز را با توجه به مجموعه داده ها تنظیم می‌کند[24]. حجم پردازش SeqKM ازدرجه‌ی $O(msd + nd)$ بوده که m تعداد خوشه‌ها و s تعداد نمونه‌ها است. این زمان بسیار کمتر از K-Means معمولی است. در اینجا داده‌ها حداکثر دو بار (یکی برای نمونه‌گیری تصادفی و دیگری برای مرحله SGD) بررسی می‌شوند. قابل توجه است که با ذخیره سازی مناسب داده‌ها، نمونه‌گیری فرعی را می‌توان بدون پایش کل مجموعه داده‌ها پیاده‌سازی کرد[8].

الگوریتم K-Means خطی: الگوریتم (۳-۱)

1. **Input:** Number of cluster k data set $X \in \mathbb{R}^{n \times d}$, sample sizr
2. **Output:** Cluster labels of each data point, centers C and cluster labels of all anchor points.
3. V=0 //Per-center count
4. M= Randomly sekect s samples from X;
5. C=Run K-Means++ algorithm on M and get k centers
6. **for** $i = 1$ **to** n **do**
7. $j = \arg min \|x_i - C_j\|^2 \quad \forall_j \in 1, \dots, k$
8. $v[j] = v[j] + 1$
9. $\eta = \frac{1}{v[j]}$
10. $C_j = (1 - \eta)C_j + \eta x_i$
11. **end for**

SeqKM می تواند یک گراف متعادل از نظر سرعت و کیفیت بسازد. با استفاده از SeqKM، ساخت گراف به راحتی به عنوان یک الگوریتم خطی روی دیسک پیاده‌سازی می‌شود، که تنگناهای بالقوه زمان انتقال گفته شده را، از بین می برد.

[1] Sequential K-Means

[2] Stochastic Gradient Gescent

۳-۴- خوشه بندی طیفی خطی

پس از ساخت ماتریس گراف دو طرفه یکه شده $\hat{Z} \in \mathbb{R}^{n\times n}$ ، باید بردارهای تکین چپ و راست $\hat{Z}$ مطابق معادله (۳-۷) مرحله خوشه‌بندی، محاسبه می‌کنیم. راه‌حل SC کوچکترین مقادیر ویژه بردار G از ماتریس لاپلاسین است. این معادله بزرگترین بردارهای تکین $\hat{Z}$ را به‌صورت زیر محاسبه می‌کند، یعنی:

$$svd(\hat{Z}) = A\Sigma B^T \qquad (7-3)$$

svd(.) بیانگر تجزیه مقادیر ویژه تکین و $\Sigma = diag(\sigma_1, \dots, \sigma_m) \in \mathbb{R}^{n\times n}$ ماتریس قطری با مقادیر تکین $\hat{Z}$ است که قطرها $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_m \geq 0, A \in \mathbb{R}^{n\times n}$ بردارهای تکین سمت چپ و $B \in \mathbb{R}^{n\times n}$ بردارهای تکین راست هستند و $G = [A^T, B^T]^T \in \mathbb{R}^{n\times n}$ است. نهایتا با اجرای الگوریتم K-Means روی G ، می‌توانیم برچسب خوشه‌ی هر نقطه داده را بدست بیاوریم.

رویکرد فوق بدلیل کاهش هزینه محاسباتی، خوشه‌بندی طیفی را در n نقطه داده به عملکرد خطی n نزدیک می‌کند. اما، هنگامی که مقدار داده از ظرفیت حافظه بیشتر شود، عملکرد این روش بشدت پایین می‌آید. این به دلیل نیاز به مبادله داده بین حافظه و دیسک است[8].

در اینجا بدلیل ساختار خاص (نازک و بلند بودن) ماتریس Z می‌توان الگوریتم SVD را به صورت خطی پیاده‌سازی کرد. با توجه به اینکه بردارهای تکین راست B و همچنین مقادیر ویژه ماتریس $\hat{Z}^T\hat{Z} \in \mathbb{R}^{m\times m}$ به‌صورت خطی با ضرب داخلی ماتریس Z و با استفاده از $\hat{Z}^T\hat{Z} = \sum_{i=1}^{n} z_i^T z_i$ محاسبه می‌شود .پس از به دست آوردن B، می‌توان A را از طریق رابطه $A = \hat{Z}B\Sigma^{-1}$ محاسبه کرد. پس، می‌توانیم یک رابطه خطی را به کمک رابطه (۳-۸) محاسبه کنیم:

$$\sigma_i = \hat{z}_i B\Sigma^{-1} \qquad (8-3)$$

که z_i iامین سطر Z و σ_i iامین سطر A است.

تجزیه مقادیر ویژه تکین(SSVD) : الگوریتم (۳-۲)

---

1. **Input:** Data matrix $Z \in \mathbb{R}^{m\times m}$, Number of singular value/vector k
2. **Output:** Singular vectors $A \in \mathbb{R}^{n\times k}$, $B \in \mathbb{R}^{m\times k}$, and Singular Value $\Sigma \in \mathbb{R}^{k\times k}$
3. S=0
4. **for** $i = 1$ **to** n **do**
5. $S = S + z_i^T z_i$//Sequentially compute $Z^T Z$
6. **end for**
7. $[B\Sigma] = eig(S, k)$ // eig is eigendecomposition
8. $\Sigma = \Sigma^{1/2}$
9. $R = \Sigma^{-1}B$
10. **for** $i = 1$ **to** n **do**

11. $A_i = z_i R$
12. **end for**

الگوریتم تجزیه و تحلیل مقادیر تکین خطی در الگوریتم (۳-۲) خلاصه شده است. ایده اصلی محاسبه ماتریس $Z^T Z$ به شکل جمع روی داده‌ها (یعنی $Z^T Z = z_i^T z_i$) برای محاسبه B و سپس محاسبه هر ردیف A را به‌صورت خطی می‌باشد. در اینجا همیشه $m \ll n$ برقرار است. بنابراین، $Z^T Z$ و B هر دو ماتریس دارای ابعادی نسبتاً کوچک هستند که برای محاسبه و ذخیره سازی از نظر زمان و مکان کارآمد است.

به کمک دو مؤلفه SeqKM و SSVD ، الگوریتم خوشه‌بندی طیفی خطی[1] را مطابق الگوریتم (۳-۳) تعریف می‌شود. در الگوریتم خوشه‌بندی طیفی خطی نیز مرحله ساخت گراف و مرحله یکه‌سازی را به صورت خطی انجام می‌دهیم. در هر مرحله، فقط به یک قطعه کوچک از داده‌ها (مثل یک سطر) و مقدار کمی از اطلاعات واسطه‌ای نیاز داریم و به همین جهت، استفاده از حافظه در داده‌های بزرگ را می‌توان در سطح بسیار پایین نگه داشت. میزان استفاده از حافظه از اندازه مجموعه داده‌ها مستقل است[8].

خوشه‌بندی طیفی خطی(SeqSC) : الگوریتم (۳-۳)

1. **Input:** Data matrix $X \in \mathbb{R}^{n \times d}$, Cluster number k. Anchor points number m.
2. **Output:** Cluster labels of each data point. All anchor point U ad cluster labels of all anchor points.
3. Generate m anchor points U using SeqKM
4. $D = 0$
5. **for** $i = 1$ **to** n **do**
6. Compute $z_i X_i$ and U
7. $D = D + z_i$
8. **end for**
9. $D = diag(D)$ // Convert D to diagonal matrix
10. **for** $i = 1$ **to** n **do**
11. $\hat{z}_i = z_i D^{-1/2}$ // Compute $\hat{Z}$ Sequentially
12. **end for**
13. Compute $[\mathrm{A}, \Sigma, \mathrm{B}] = SSVD(\hat{Z}, k)$
14. Apply SeqKM on A to get cluster labels

[1] Sequentioal Spectral Clustering

فصل چهارم

پیاده‌سازی

پیاده‌سازی

پیاده‌سازی این پروژه، به زبان پایتون[1] که یکی از پرکاربردترین زبان‌ها برای پروژه‌های حوزه هوش مصنوعی می‌باشد، انجام گرفته است.

روند کلی پیاده‌سازی این پروژه به شرح زیر بوده است :

ابتدا مراحل پیش‌پردازش، شامل اعمال فیلترهای مختلف برای بهبود تصاویر و تغییر قالب داده به‌صورت قابل استفاده در الگوریتم‌ها، انجام می‌گیرد. پس از این مرحله، به انتخاب نقاط کمکی از طریق اجرای الگوریتم SeqKM می‌پردازیم. سپس به کمک نقاط میانی ماتریس Z و در ادامه $\hat{Z}$ را می‌سازیم. پس از ساخت این ماتریس، ماتریس A را که ماتریس تکین چپ $\hat{Z}$ می باشد را به کمک الگوریتم SSVD محاسبه می‌کنیم و بر روی K ستون کوچکتر این ماتریس بار دیگر الگوریتم SeqKM را اجرا می‌کنیم و برچسب هر عکس را استخراج می‌کنیم. حال با توجه به برچسب اختصاص یافته برای هر یک از داده‌ها کیفیت خوشه‌بندی را بررسی و گزارش می‌کنیم.

در ادامه به بررسی نحوه پیاده‌سازی هر کدام از این مراحل و هر یک از مؤلفه‌های الگوریتم خواهیم پرداخت.

[1] python

۴-۱- پیش‌پردازش داده‌ها

تصاویر مورد استفاده در این پروژه هر کدام به وسیله یک ماتریس دوبعدی نمایش داده می‌شوند. هر کدام از خانه‌های این ماتریس، نماینده یک پیکسل از عکس است و یک مقدار بین ۰ تا ۲۵۵ دارد که ۰ به معنی کاملا سیاه و ۲۵۵ به معنی کاملا سفید است. برای این عکس ها یک آستانه[1] سفید بودن در نظر می‌گیریم و عددهای بیشتر از آن را به ۲۵۵ و عددهای کم‌تر از آن را به ۰ تغییر می‌دهیم. در این پروژه این آستانه ۲۰ در نظر گرفته‌شده‌است:

```
def make_0_255(X_train):
    for k in range(len(X_train) ):
        for i in range(len(X_train[0])):
            for j in range(len(X_train[0][0])):
                if X_train[k][i][j]<20 :
                    X_train[k][i][j] = 0
                else :
                    X_train[k][i][j] = 255
    return X_train
```

پس از اعمال این فیلتر بر روی داده‌ها، تصاویر را از حالت دوبعدی به حالت یک آرایه طولانی یک‌بعدی تغییر می‌دهیم تا آماده پردازش‌های بعدی شود:

```
def transform(X_train):
    ans = []
    for img in X_train:
        temp = []
        for row in img:
            temp.extend(row)
        ans.append(temp)
    return ans
```

۴-۲- پیاده‌سازی الگوریتم SeqKM

پیاده‌سازی صورت گرفته برای این الگوریتم دقیقا مطابق توضیحات الگوریتم (۳-۱) می‌باشد. در بخش kmeans++ از پیاده‌سازی آماده موجود در کتابخانه sklearn استفاده شده‌است. معیار فاصله استفاده‌شده در این بخش، فاصله اقلیدسی می‌باشد. ورودی تابع تعداد مراکز خوشه، داده ورودی و تعداد نمونه برای اجرای kmeans++ می‌باشد. برای نمونه‌گیری به‌طور تصادفی از choices موجود در کتابخانه random استفاده می‌کنیم. پیاده‌سازی این بخش به صورت زیر است:

[1] Threshold

```
import random as rd
import math
import Kmeans

def euclidean_distance(img_a, img_b):
    count = 0
    for i in range(0, len(img_a)):
        temp = img_a[i] - img_b[i]
        count = (temp ** 2) + count
    count = math.sqrt(count)
    return count


def seqkm(k, Images, SampleSize):
    print("SeqKM start")
    v = []
    PredictedLabels = []
    f = k
    while f > 0:
        v.append(0)
        f = f - 1
    M = rd.choices(Images, k=SampleSize)
    print("choose " + str(k) + " centroid with kmeans++")
    centers,label = Kmeans.KMeansPlusplus(M, k)
    f = 0
    i = 0
    for image in Images:
        distances = [euclidean_distance(centroid, image)
                     for (centroid) in centers]
        j = distances.index(min(distances))
        PredictedLabels.append(j)
        i = i + 1
        v[j] = v[j] + 1
        epsilon = 1 / v[j]
        f = f +
        print("update centroid number " + str(j))
        for i in range(0, len(image)):
            centers[j][i] = ((1 - epsilon) * centers[j][i] + 0.5) +
(epsilon * image[i]+0.5)
    print("SeqKM done")
    return v, PredictedLabels, centers
```

خروجی این تابع به ترتیب، تعداد اعضای هر خوشه، برچسب‌های اختصاص داده‌شده به هر عکس و مراکز خوشه می‌باشد.

۴-۳-پیاده‌سازی الگوریتم SSVD

در پیاده‌سازی این بخش، برای محاسبه ضرب ماتریسی از تابع matmul موجود در کتابخانه numpy استفاده شده است. همچنین برای محاسبه ماتریس ترانهاده[1] از تابع transpose همین کتابخانه استفاده‌شده است. همین‌طور در این‌جا برای به توان رساندن ماتریس‌ها از کتابخانه scipy از بخش توابع مربوط به جبرخطی استفاده می‌کنیم.

```
import numpy as np
from scipy.linalg import fractional_matrix_power

def ssvd(z):
    zt = np.transpose(z)
    s = []
    v = []
    for row in zt:
        temp = []
        for col in z:
            temp.append(np.matmul(row, col))
        s.append(temp)
    B, sigma = np.linalg.eig(s)
    sigma = fractional_matrix_power(sigma, 0.5)
    sigma_inverse = fractional_matrix_power(sigma, -1)
    R = np.matmul(sigma_inverse, B)
    A = []
    for zi in z:
        A.append(np.matmul(zi, R))
    return A, sigma, B
```

قابل توجه است که به جای نوشتن این تابع می‌توان از یک خط کد به صورت زیر استفاده کرد و تفاوت اساسی این کد در کم‌تر بودن بار محاسباتی و خطی بودن این نوع پردازش است.

```
A, sigma, B= scipy.linalg.svd(z)
```

در تابع نوشته شده، برای تجزیه مقادیر ویژه از تابع کتابخانه‌ای eig از کتابخانه numpy استفاده شده‌است.

[1] Transpose

۴-۳- پیاده‌سازی الگوریتم SeqSC

این بخش، در حقیقت بخش اصلی این پروژه و محل استفاده از توابع نوشته‌شده در بخش‌های قبلی می‌باشد. ابتدا به بررسی هر کدام از اجزا مورداستفاده در این تابع می‌پردازیم. اولین تابع مورد استفاده در این بخش، تابع هسته[1] است. تابع مورد استفاده در این پیاده‌سازی، از نوع گوسی[2] می‌باشد.

```
def kernel(xi, uj):
    # uj = uj[0]
    k_ans = 0
    for i in range(0, len(xi)):
        d = (np.absolute(xi[i] - uj[i]))
        x = -1 * d
        k_ans = k_ans + math.exp(x)
    return k_ans
```

تابع بعدی، تابع محاسبه نقطه‌های کمکی نزدیک به یک تصویر می‌باشد. در این تابع، یک آرایه از فاصله‌ها ایجاد و از کوچک به بزرگ مرتب می‌کنیم و در نهایت p تای کوچک‌تر را به عنوان خروجی تابع بازمی‌گردانیم.

```
def compute_p_nearest(xi, p, anchors):
    distances = []
    ans = []
    counter = 0
    for anchor in anchors:
        ans.append(counter)
        counter = counter + 1
        distances.append(euclidean_distance(xi, anchor))
    ind = np.argsort(distances)
    ans = np.array(ans)
    ans = ans[ind]
    return (ans[0:p])
```

در زیر تابع استفاده‌شده دیگر، که وظیفه محاسبه فاصله اقلیدسی دارد، آمده است:

```
def euclidean_distance(a, b):
    return np.sum(np.subtract(a, b) ** 2)
```

[1] kernel

[2] Gaussian

و حال به بررسی تابع اصلی SeqSc می‌پردازیم. این تابع به عنوان ورودی بردار تصاویر، تعداد خوشه‌ها و تعداد نقاط کمکی دریافت می‌کند و خروجی تابع برچسب اختصاص‌داده‌شده به هر تصویر، لیست مراکز خوشه‌ها و نقاط کمکی می‌باشد.

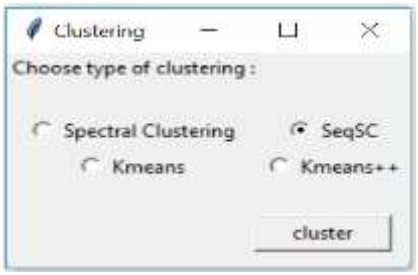
```
import math
import SSVD
import numpy as np
import scipy
import SeqKM

def seqsc(x, k, m):
    print("SeqSC start")
    my_x = transform(x)
    v, label_all, anchors = SeqKM.seqKM(m, my_x, 3 * m)
    p = 5
    d = [0] * m
    lenx = len(x)
    z = []
    for i in range(0, lenx):
        temp = []
        for j in range(0, m):
            temp.append(0)
        z.append(temp)
    z_bar = []
    for i in range(0, lenx):
        z_bar.append([0])
    print("build Z^")
    for i in range(0, len(x)):
        ux = compute_p_nearest(my_x[i], p, anchors)
        sum_k = 0
        for j in ux:
            z[i][j] = kernel(my_x[i], anchors[j])
            sum_k += z[i][j]
            d[j] = d[j] + z[i][j]
         for j in ux:
            z[i][j]/=sum_k
    d = np.diag(d)
    d = scipy.linalg.fractional_matrix_power(d, -0.5)
    for s in range(0, len(x)):
        z_bar[s] = np.matmul(z[s], d)
    A, sigma, B = SSVD.ssvd(z_bar)
    A_my = build_A(A, k)
    label_all, centers = SeqKM.seqKM(k, A_my, k*3)
    print("SeqSC done")
    return label_all, centers, anchors
```

در این تابع تعداد نمونه جهت اجرای خوشه‌بندی در تابع SeqKM، سه برابر تعداد مراکز خوشه در نظر گرفته شده است.

۴-۴-پیاده‌سازی واسط کاربری

برای این پروژه، همانند تعریف صورت گرفته در پورپوزال، یک واسط کاربری با استفاده از کتابخانه tkinter پیاده‌سازی شده‌است. در ابتدای اجرا، باید نوع خوشه‌بندی موردنظر را از منویی به شکل (۴-۱) انتخاب کرد.

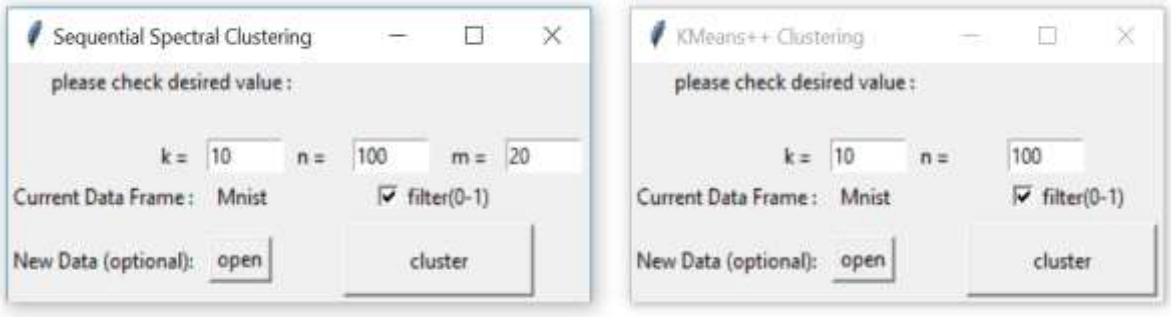


شکل ۴-۱ : منوی انتخاب نوع خوشه‌بندی

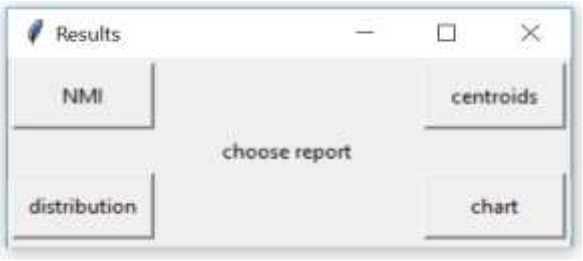
پس از این مرحله با توجه به نوع خوشه‌بندی انتخاب‌شده، ورودی های مورد نیاز برای اجرای الگوریتم و فایل شامل داده‌ها را در منویی به شکل (۴-۲) مشخص می‌کنیم. در صورت لزوم به اجرای فیلتر شرح داده‌شده در بخش ۴-۱، می‌توان علامت مربوطه را زد.

پس از مشخص کردن مقادیر موردنیاز، با فشردن دکمه cluster خوشه‌بندی اجرا می‌شود .

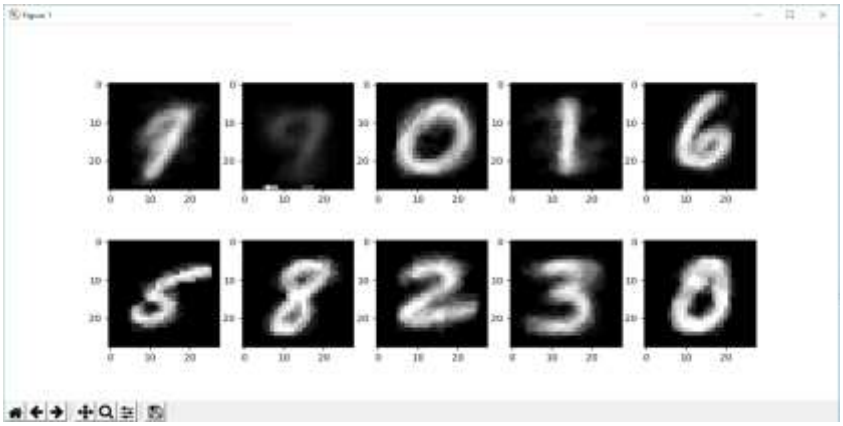
نتیجه اجرا به ۴ طریق مطابق شکل(۴-۳) گزارش می‌شود. با انتخاب گزینه centroids مراکز خوشه‌ها مانند شکل(۴-۴) به نمایش درمی‌آید.



شکل ۴-۲: منو دریافت مقادیر اجرای خوشه‌بندی

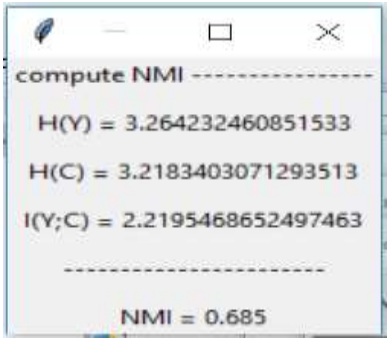


شکل ۴-۳: منو انتخاب نوع نمایش گزارش خوشه‌بندی



شکل ۴-۴ : نمایش مراکز خوشه‌بندی

با انتخاب گزینه NMI، آنتروپی[1] برچسب اصلی داده‌ها و برچسب‌های نسبت داده‌شده توسط الگوریتم انتخابی، اطلاعات متقابل[2] محاسبه‌شده و مقدار NMI برای خوشه‌بندی صورت‌گرفته، مطابق شکل(۴-۵) نمایش داده می‌شود.



شکل ۴-۵: نمایش NMI خوشه‌بندی

با انتخاب گزینه distribution، تعداد اعضای هر خوشه، تعداد از هر کدام از برچسب‌ها، تعداد هر کدام از برچسب‌ها در هر خوشه و تعداد کل تصاویر، مطابق شکل (۴-۶) نمایش داده می‌شود.

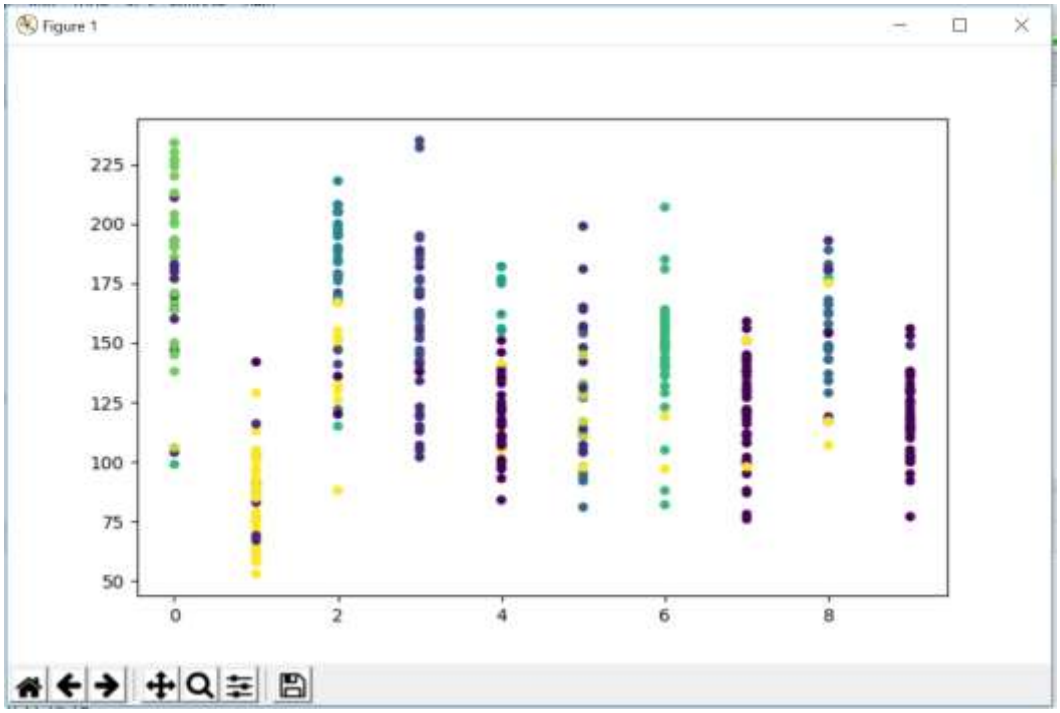
[1] Entropy

[2] Mutual information

	c0	c1	c2	c3	c4	c5	c6	c7	c8	c9	sum_all
T_0	0	0	0	0	0	0	7	6	0	0	13
T_1	0	0	0	0	6	0	0	0	0	8	14
T_2	1	0	1	0	0	0	0	0	0	4	6
T_3	0	9	1	0	1	0	0	0	0	0	11
T_4	2	0	0	1	0	2	0	0	4	2	11
T_5	0	1	0	0	0	0	0	0	0	4	5
T_6	0	0	0	11	0	0	0	0	0	0	11
T_7	5	0	0	0	0	5	0	0	0	0	10
T_8	0	0	8	0	0	0	0	0	0	0	8
T_9	4	1	0	0	0	5	1	0	0	0	11
sum_cluster	12	11	10	12	7	12	8	6	4	18	100

شکل ۴-۶: نمایش وضعیت پراکندگی داده‌ها در خوشه‌ها

گزینه آخر، نمایش نمودار وضعیت داده‌ها می‌باشد. در این نمودار، عکس‌هایی که شامل اعداد یکسانی هستند در یک ستون، که همان ستون عدد موجود در تصویر است، نمایش داده می‌شود. رنگ هر تصویر، برچسب تخمینی توسط الگوریتم است. حالت ایده‌آل یک رنگ بودن اعضای هر ستون است. نمودار تشکیل‌شده شبیه تصویر (۴-۷) می‌باشد.



شکل ۴-۷: نموداز گرافیکی وضعیت خوشه‌بندی داده‌ها

فصل پنجم
آزمایش و ارزیابی

آزمایش و ارزیابی

در این بخش، رویکرد خوشه‌بندی طیفی خطی با سه الگوریتم خوشه‌بندی طیفی، K-Means و K-Means++ مقایسه شده‌است. در هر سه الگوریتم، از پیاده‌سازی استاندارد کتابخانه sklearn که دقیقا منطبق بر الگوریتم‌ها می‌باشد، استفاده گردیده‌است.

تمام آزمایشات ما بر روی یک لپ‌تاپ با مشخصات فنی CPU 3.0GHz Intel Corei7 و RAM 16 GB انجام شده است. ما آزمایشات را با سه مجموعه داده MNIST[1]، Fashion-MNIST[2] و CovType[3] انجام می‌دهیم. تصاویر هر سه این مجموعه داده‌ها، برای اندازه‌گیری کیفیت خوشه‌بندی هستند. الگوریتم پیاده‌سازی‌شده را همانند تعریف پروژه، برای خوشه‌بندی مجموعه اول یعنی تصاویر اعداد دست‌نویس بکار می‌گیریم و از دو مجموعه دیگر برای اطمینان از صحت عملکرد در دیگر حوزه‌ها (و نه فقط تصاویر اعداد) استفاده می‌کنیم.

مجموعه داده MNIST شامل ۷۰،۰۰۰ تصویر از رقم‌های دست نوشته از ۰ تا ۹ است. برای نمایش هر تصویر از ۷۸۴ مقدار پیکسل اصلی استفاده می‌کنیم. فضای رنگی تمامی تصاویر سیاه و سفید است. هر پیکسل شامل یک عدد بین ۰ تا ۲۵۵ می‌باشد و ۰ به معنی سیاه و ۲۵۵ به معنی سفید است.

مجموعه داده Fashion-MNIST شامل ۶۰،۰۰۰ تصویر از انواع لباس هستند. نام هر یک از تصاویر، یک عدد است و و نشان‌دهنده نوع آن لباس است؛ دسته‌بندی به شرح زیر است :

۰-تاپ ۱-شلوار ۲-پولیور ۳-لباس زنانه ۴-کت ۵-صندل ۶-پیراهن مردانه ۷-کتونی ۸-کیف ۹- چکمه

مجموعه‌داده CovType ۵۸۰،۰۰۰ تصویر از پوشش گیاهی جنگل‌ها می‌باشد. این تصاویر در ۷ دسته، تقسیم شده‌اند.

در ادامه ابتدا به صورت نظری، مرتبه پیچیدگی و میزان مصرف حافظه الگوریتم خوشه‌بندی طیفی خطی و خوشه‌بندی طیفی را بررسی می‌کنیم. سپس از شاخص‌های مقایسه‌ی زمان و برای بررسی عملی روش پیاده‌سازی شده با الگوریتم‌های یادشده در ابتدای بخش استفاده می‌کنیم.

[1] http://yann.lecun.com/exdb/mnist/

[2] http://yann.lecun.com/exdb/fashion-mnist

[3] https://archive.ics.uci.edu/ml/datasets/covertype

۵-۱- ارزیابی نظری

الگوریتم خوشه‌بندی طیفی خطی به طور کلی شامل سه مرحله است:

۱) تولید نقاط کمکی U با خوشه‌بندیSeqKM

۲) ساخت گراف دو طرفه Z

۳) اجرای خوشه‌بندی طیفی

زمان موردنیاز برای تولید نقاط میانی زمان از مرتبه $O\ (ksd + nmd)$ است. مرحله ساخت گراف دوطرفه z، از مرتبه $O\ (nmd)$ است و مرحله آخر، زمانی از مرتبه $O\ (nm^2)$ را می‌گیرد. پس در نتیجه کل زمان اجرا از مرتبه $O\ (nm^2)$ خواهد بود[8].

در مقابل، در الگوریتم خوشه‌بندی طیفی، مرحله ساخت ماتریس شباهت از مرتبه $O\ (n^2)$ است. مرحله تجزیه مقادیر ویژه برای ماتریس لاپلاسین از مرتبه $O\ (n^3)$ می‌باشد. مرحله آخر الگوریتم که شامل اجرای K-Means می‌باشد نیز از مرتبه $O\ (nldk)$ می‌باشد؛ که در این محاسبه l تعداد دفعات تکرار الگوریتم، d مرتبه درجه ورودی، n تعداد داده و k تعداد خوشه‌ها می‌باشد. و در نتیجه تمام این محاسبات، این الگوریتم از مرتبه $O\ (n^3)$ می‌باشد[25]. با توجه به این که m از مرتبه پایین‌تری به نسبت n است، پس این الگوریتم پیچیدگی زمانی کم‌تری به نسبت خوشه‌بندی طیفی اصلی دارد.

هم‌چنین در مقایسه مصرف حافظه برای این دو الگوریتم، میزان استفاده از حافظه برای الگوریتم پیشنهادی بدین صورت است؛ اولین فراخوانی SeqKM در الگوریتم (۳-۳) به فضایی از مرتبه $\mathrm{O}((\mathrm{s}+\mathrm{n})^d) \approx$ $O(sd)$ نیاز دارد، در حالی که فراخوانی دومی فضایی معادل $\mathrm{O}(s+k)m) \approx O(sm)$ را اشغال می‌کند. ساخت گراف و یکه‌سازی گراف فضای $O(m^2 + md)$ را نیاز دارد و مرحله SSVD فضایی از مرتبه $O\ (mk)$ می‌گیرد. بنابراین، پیچیدگی کلی برای مصرف حافظه از مرتبه $O\ ((m + d)m)$ می‌باشد. پیچیدگی فضایی الگوریتم خوشه‌بندی طیفی اصلی حداقل ازمرتبه $O(n * \max(m, d))$ است. این مرتبه برای n بسیار بزرگ، به دلیل نیاز به جابجایی مابین حافظه و دیسک و همین‌طور نیاز به تعویض دیسک، باعث ایجاد مشکل در عملکرد خواهد شد[8].

۵-۲- ارزیابی عملی

در این بخش به مقایسه عملکرد الگوریتم با سه الگوریتم خوشه‌بندی طیفی اصلی، K-Means و K-Means++ می‌پردازیم. طبق ادعای صورت‌گرفته، این الگوریتم باید هم از جهت کیفیت خوشه‌بندی و هم از جنبه سرعت، بهبود ایجاد کند. به همین دلیل برای مقایسه کیفیت از شاخص NMI و از شاخص زمان برای مقایسه سرعت استفاده شده‌است.

۵-۲-۱- شاخص NMI

هدف از فرایند خوشه‌بندی، تقسیم‌بندی مجموعه داده به زیرمجموعه‌ها یا خوشه‌هاست که درجه‌ی شباهت بین اعضای هر زیر مجموعه بالا باشد. موضوعی که در خوشه‌بندی بسیار حائز اهمیت است، اعتبارسنجی نتایج خوشه‌بندی است. اعتبارسنجی مشخص می‌کند روش خوشه‌بندی مورد استفاده به چه میزان به صورت صحیح داده‌ها را خوشه‌بندی کرده است. وقتی نتایج خوشه‌بندی فقط براساس داده‌هایی که خوشه-بندی شده‌اند مورد ارزیابی قرار گیرد، اعتبارسنجی، داخلی گفته می‌شود، اما هنگامی که از اطلاعات خارجی جهت ارزیابی نتایج خوشه‌بندی استفاده شود، اعتبار سنجی بیرونی گفته می‌شود.

شاخص اعتبارسنجی می‌تواند در تعیین تعداد صحیح خوشه‌های یک مجموعه داده و یا مقایسه و ارزیابی روشهای گوناگون خوشه‌بندی مورد استفاده قرار گیرد. روش‌های ارزیابی الگوریتم‌های خوشه‌بندی عبارتند از،Entropy، Purity، Jacard، F-Measure و NMI می‌باشند[25]. برای اندازه‌گیری مقدار دقت خوشه‌بندی جهت تعیین میزان پایداری الگوریتم‌ها از معیار F-Measure استفاده می‌شود. از معیار Jacard نیز در تعیین شباهت نتایج خوشه‌بندی استفاده می‌شود. Entropy نشان می‌دهد که یک کلاس در یک خوشه چگونه توزیع شده است، که باید یک عدد کوچکی باشد. از طرفی عملکرد روش‌های مختلف خوشه‌بندی را می‌توان با استفاده از فرایند باز برچسب‌گذاری[1] بین برچسب کلاس‌های واقعی با برچسب خوشه‌های به دست آمده و مقایسه آنها با هم، توسط معیار خلوص[2] با رابطه زیر سنجید.

$$Purity = \frac{1}{N}\sum_{i=1}^{k} \max_{j} |c_i \cap t_j| \qquad (1-5)$$

در رابطه (۵-۱) t_j طبقه‌بندی است که بیشترین تعداد خوشه c_i را دارا می‌باشد. تعداد نمونه‌ها با N و تعداد خوشه‌ها با K مشخص می‌شود. هر گاه تعداد خوشه‌ها نسبت به مقدار نمونه‌ها زیاد می‌باشد، معیار Purity پاسخ مناسب را ارایه نمی‌دهد، بنابراین از معیار NMI مطابق رابطه (۵-۲) استفاده می‌کنیم.

[1] Relabeling

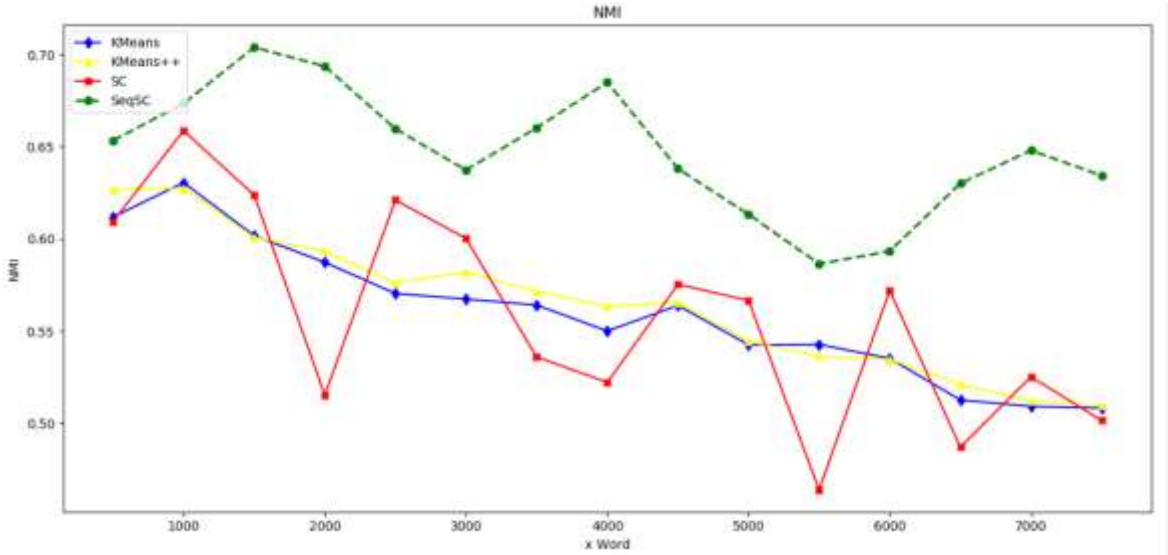
[2] Purity

$$NMI(Y,c) = \frac{2 \times I(Y;C)}{[H(Y) + H(C)]} \, , I(Y;C) = H(Y) - H(Y|C) \qquad (2-5)$$

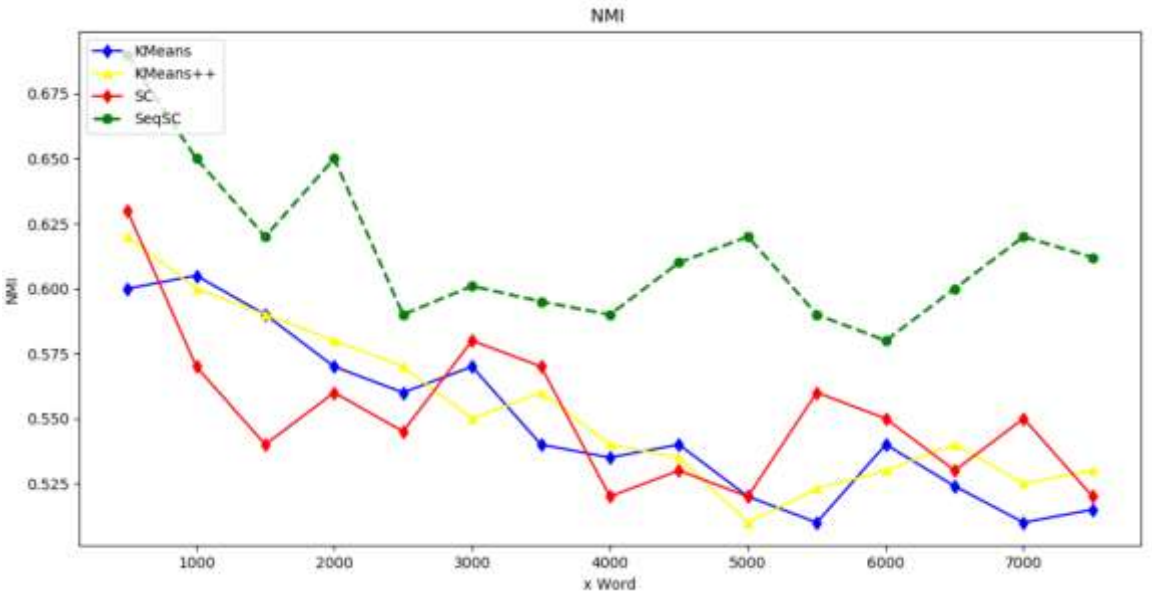
که Y برچسب داده و C برچسب خوشه و $I(Y;C)$ اطلاعات متقابل برچسب خوشه و داده می‌باشد[26].

۵-۲-۲- مقایسه کیفیت

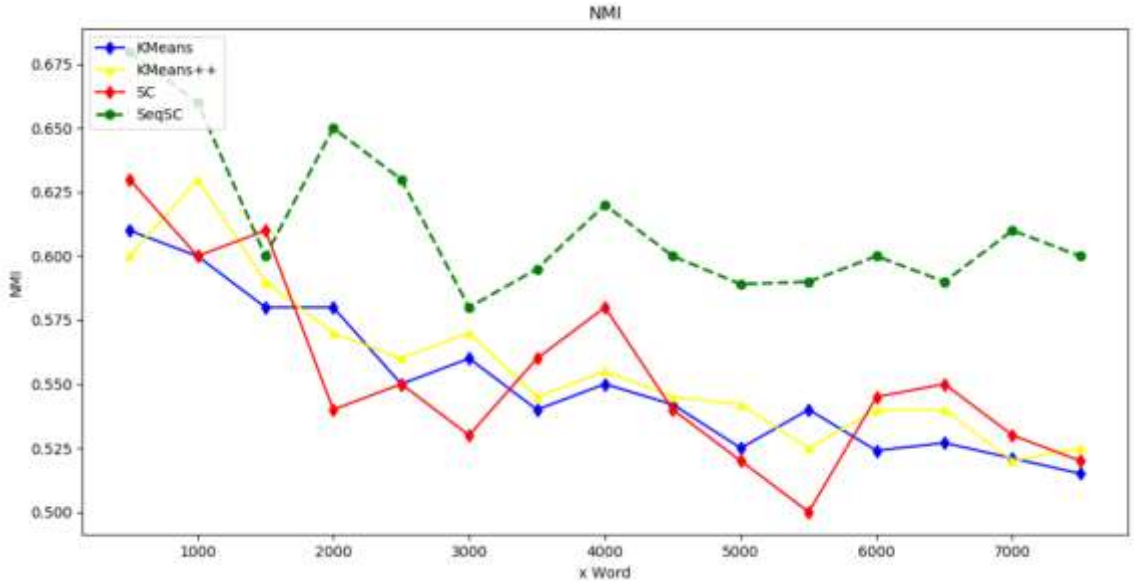
برای مقایسه کیفیت ۲۰ مرتبه هر ۴ الگوریتم را با n با ۱۵ مقدار از ۵۰۰ تا ۷۵۰۰ آزمایش می‌کنیم. شاخص NMI میانگین، برای هر کدام از این ۱۵ مقدار بر روی هر سه مجموعه داده، محاسبه‌شده و در تصویر (۵-۱)، (۵-۲) و (۵-۳) به نمایش درآمده است.



شکل ۵-۱: نمودار نمایش نتایج شاخص NMI برای مجموعه داده MNIST



شکل ۵-۲: نمودار نمایش نتایج شاخص NMI برای مجموعه داده Fashion-MNIST



شکل ۵-۴ : نمودار زمان اجرا متوسط برای مجموعه داده CovType

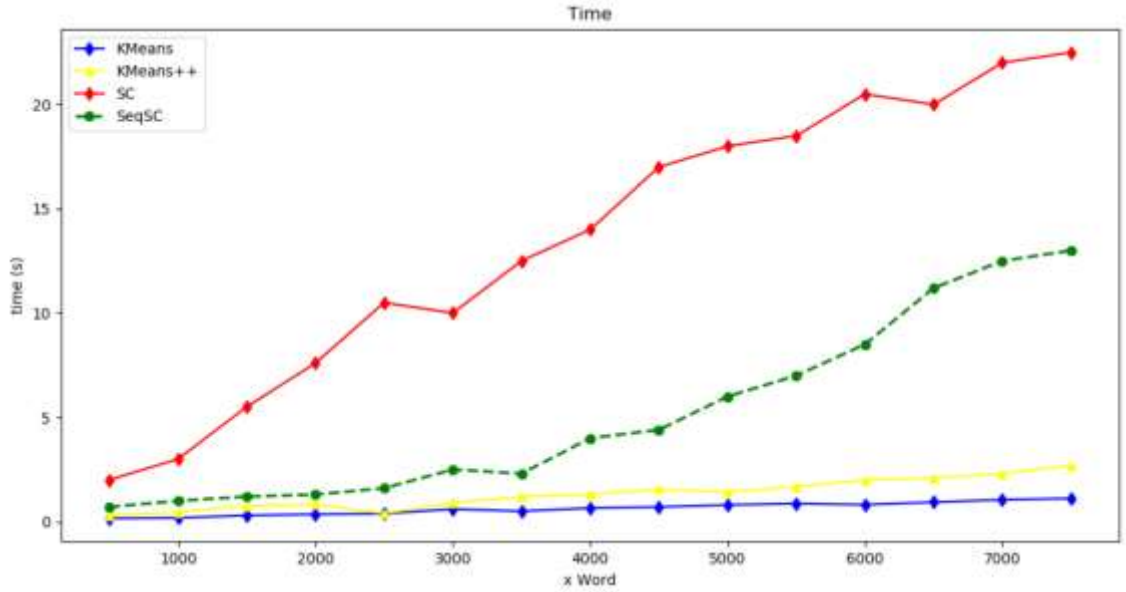
طبق این تصاویر، همان‌طور که می‌بینید، شاخص NMI برای الگوریتم پیاده‌سازی‌شده بسیار بالا بوده و تقریبا در تمامی مقادیر از سایر الگوریتم بهتر جواب داده‌است. این برتری می‌تواند به دلیل استفاده از الگوریتم SeqKM باشد؛ درواقع به علت بهره‌گیری از این الگوریتم در هر دو مرحله انتخاب نقاط میانی و خوشه‌بندی نهایی، هم پراکندگی تمامی داده‌ها به نحو مناسبی پوشش داده می‌شود و هم مراکز خوشه به درستی انتخاب می‌شود و در نتیجه کیفیت خوشه‌بندی تا حد خوبی افزایش می‌یابد .

الگوریتم‌های دسته K-Means یک مسیر با دامنه تغییرات کم نزولی را طی کرده‌اند اما NMI الگوریتم خوشه‌بندی طیفی اصلی به نسبت، بسیار پرتغییر و برای مقادیر مختلف متفاوت است.

قابل توجه است که مقدار شاخص NMI از ۰,۴ به بالا مناسب است. میزان متوسط NMI برای این ۳۰۰ اجرا برابر با ۰,۶۸ بوده‌است که برای یک الگوریتم خوشه‌بندی مقدار بسیار بالایی می‌باشد.

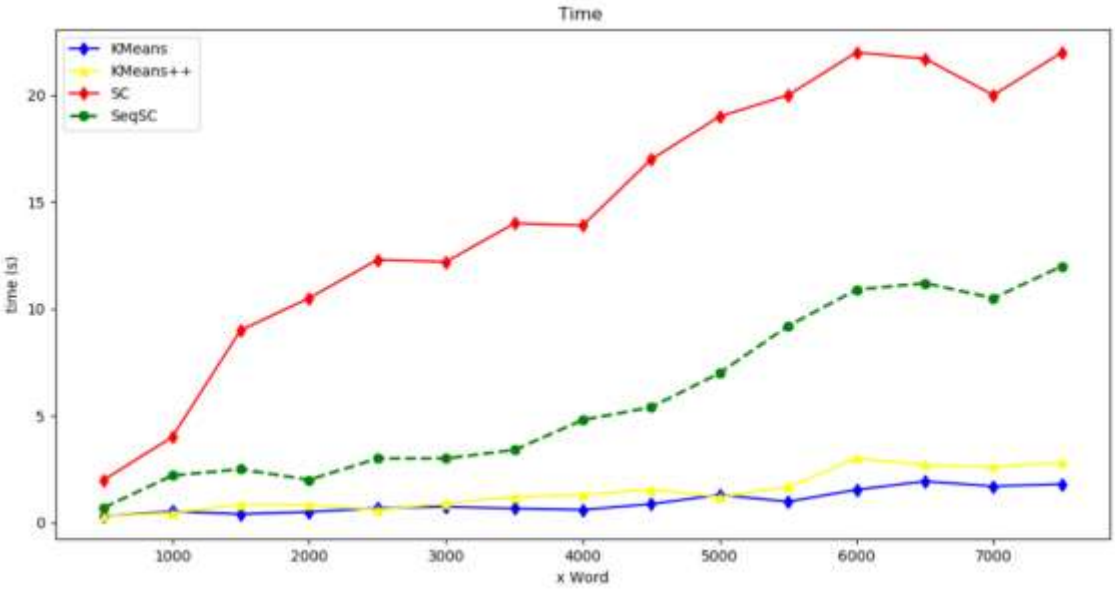
۵-۲-۳- مقایسه زمانی

برای مقایسه هرچه بهتر زمان اجرا نیز ۳۰ مرتبه هر ۴ الگوریتم را با n برابر با ۱۵ مقدار از ۵۰۰ تا ۷۵۰۰ آزمایش می‌کنیم. میزان زمان اجرای متوسط برای هر کدام از این ۱۵ مقدار محاسبه‌شده و در تصویر (۵-۲) به نمایش درآمده است.

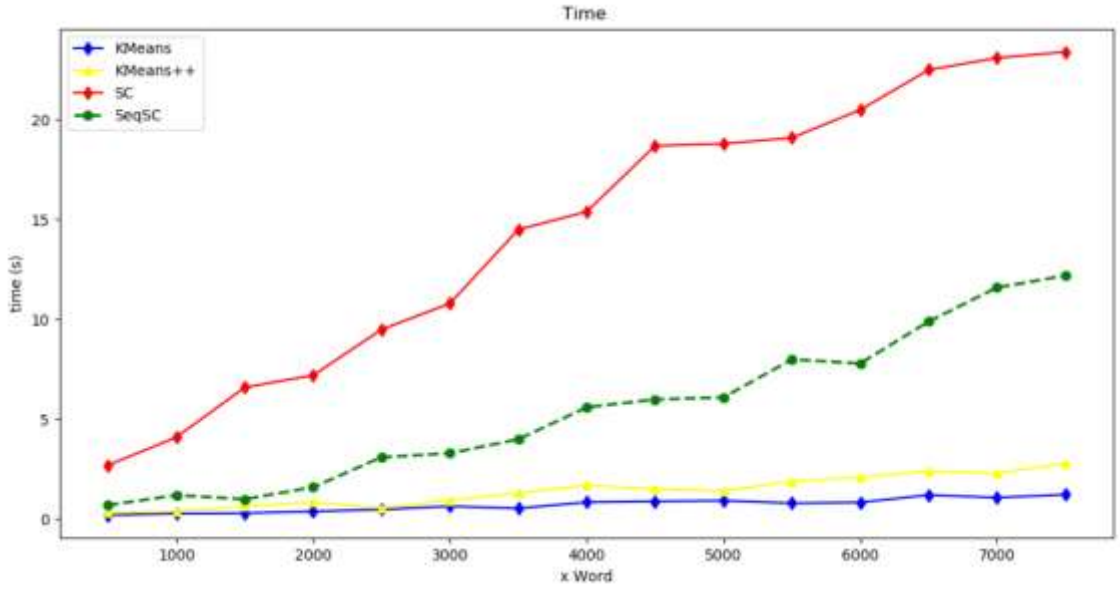


شکل ۵-۴ : نمودار زمان اجرا متوسط برای مجموعه داده MNIST

برای ارزیابی بیشتر الگوریتم، بر روی دو مجموعه داده Fashion-Mnist و CovType نیز با دفعات تکرار و تعداد داده بالا، آزمایش می‌کنیم. زمان اجرایی برروی این دو مجموعه داده مطابق شکل (۵-۴) و (۵-۵) می‌باشد



شکل ۵-۵ : نمودار زمان اجرا متوسط برای مجموعه داده Fashion-MNIST



شکل ۵-۶ : نمودار زمان اجرا متوسط برای مجموعه داده CovType

همان‌طور که در نمودار نیز می‌بینید، میانگین زمان اجرای الگوریتم K-Means در این ۹۰۰ اجرا از دیگر الگوریتم‌ها کم‌تر است. پس از آن الگوریتم K-Means++ با اختلاف کم قرار گرفته‌است و سپس الگوریتم پیاده‌سازی‌شده در رتبه سوم و در نهایت با اختلاف قابل توجه، الگوریتم خوشه‌بندی طیفی اصلی قرار گرفته‌است.

در مورد بیش‌تر بودن زمان اجرای الگوریتم K-Means++ به نسبت الگوریتم K-Means باید گفته‌شود که به دلیل محاسبه احتمالات مورد نیاز برای انتخاب مرکز خوشه‌های اولیه این اختلاف اندک شکل گرفته-است. مشخصا ما نیز انتظار سرعت بیشتر از این الگوریتم‌ها به نسبت الگوریتم پیاده‌سازی‌شده داریم، زیرا حداقل در هر مرتبه اجرای الگوریتم، دو مرتبه الگوریتم K-Means++، یکی برای انتخاب نقاط میانی و دیگری برای محاسبه برچسب‌های نهایی اجرا می‌شود. علت بیش‌تر بودن زمان اجرای الگوریتم خوشه‌بندی طیفی اصلی به نسبت الگوریتم پیاده‌سازی‌شده نیز مشخص است؛ زیرا الگوریتم اصلی باید تجزیه مقادیر تکین را روی ماتریسی با ابعاد بسیار بزرگ‌تر $(n \times n)$ از ماتریس الگوریتم پیاده‌سازی شده $(n \times m)$ انجام دهد که نیاز به زمان بیش‌تری دارد.

فصل ششم

جمع‌بندی و کارهای آینده

جمع‌بندی و کارهای آینده

در این تحقیق، ابتدا به لزوم بازشنایی اعداد دست‌نویس و کاربردهای آن اشاره کریم و سپس روش‌های مختلف بازشنایی اعداد دست‌نویس وهمچنین کارهای پیشین انجام شده را معرفی کردیم که روش خوشه-بندی یکی از روش‌های موثر در این حوزه می‌باشد، ولی این روش بدلیل محاسبات سنگین بخصوص برای داده‌هایی با مقیاس بالا، کمتر مورد استفاده قرار می‌گیرد. در ادامه مبانی موردنیاز که شامل الگوریتم‌های K-Means,K-Mean++,SVD و خوشه‌بندی طیفی می‌باشند بررسی کردیم. سپس الگوریتم خوشه‌بندی طیفی خطی را توضیح دادیم. الگوریتم خوشه‌بندی طیفی خطی از دو مؤلفه اساسی تشکیل شده‌است: الگوریتم K-Means خطی و الگوریتم‌های تجزیه مقادیر تکین خطی.

مزایای خوشه‌بندی طیفی خطی شامل: (۱) الگوریتم K-Means خطی الگوریتم K-Means را با انتخاب نقاط اولیه بهتر، بهبود می‌دهد که می‌تواند مراکز خوشه ای با کیفیت بالا را در اولین بار بررسی و کشف کند. (۲) الگوریتم K-Means خطی مرحله ساخت گراف را به صورت خطی اجرا می‌کند و این عملیات تاثیر بسزایی در زمان اجرایی الگوریتم دارد . (۳) با استفاده از الگوریتم تجزیه مقادیر تکین خطی، خوشه-بندی به صورت خطی انجام می‌شود و کل پیچیدگی زمان با تعداد نقاط داده ورودی تقریباً رابطه خطی دارد. چنین مزایایی باعث می شود که الگوریتم خوشه‌بندی طیفی خطی دقیق و مقیاس پذیر باشد تا بتواند مشکل مجموعه داده‌های انبوه را حل کند.

نتایج تجربی بر روی مجموعه داده‌ها نشان داد که الگوریتم خوشه‌بندی طیفی خطی نه تنها از جنبه‌ی نظری بلکه در پیاده‌سازی نیز عملکرد بهتری به نسبت سایر روش‌ها دارد. این مقایسه به کمک دو شاخص زمان و NMI بررسی گردید و نتایج نظری را تایید کرد.

از کارهای دیگری که در ادامه این پروژه می‌توان انجام داد، استفاده از گراف‌های سلسله مراتبی دوبخشی به جای گراف‌های یک لایه دوبخشی در مرحله ساخت ماتریس Z برای بهبود نقاط میانی انتخابی، می‌باشد. هم‌چنین می‌توان مرحله تخمین تعداد خوشه‌ها را، برای تشخیص خودکار[1] تعداد آن‌ها به مراحل پیش-پردازش داده اضافه کرد. برای این منظور استفاده از روش PCA[2] پیشنهاد می‌گردد[27].

[1] Automated
[2] Principal Component Analysis

منابع و مراجع

[1] C. Suen, M. Berthold and S. Mori, "*Automatic Recogniation of Handprinted Characters, yhe State of the Art*", IEEE, 2002.

[2] R. Duda and P. Hart, "Pattern Classification and Scene Analysis", Wilkey, 1993

[3] J.Han, M.Kamber, "*Data Mining Concepts and Techniques*",Morgan Kaufmamn Series, 2016.

[4] X.Chang, F.Nie, Z. Ma, Y. Yang and X. Zhou, "*A convex formulation for spectral shrunk clustering*". AAAI, 2015.

[5] J. Shi, and J. Malik. "Normalized cuts and image segmentation. Pattern Analysis and Machine Intelligence", IEEE, 2000.

[6] V. Luxburg, *"A tutorial on spectral clustering*", Statistics and computing, 2007.

[7] D.Yan, L.Huang and M. JordanI, "*Fast approximate spectral clustering*", IEEE, 2009.

[8] Y. Li, J. Huang, W. Liu, "Scalable Sequential Spectral Clustering", AAAI, 2016.

[9] C.Fowlkes, S. Belongie, F.Chung and J. Malik, "*Spectral grouping using the nystrom method*". IEEE, 2009.

[10] J. Liu, C. Wang, J. Gao and J. Han, "*Multiview clustering via joint nonnegative matrix factorization*", SDM, 2013.

[11] N. Khoa and S. Chawla, "*Large scale spectral clustering using resistance distance and spielman-teng solvers*", Springer, 2012.

[12] H. Shinnou and M. Sasaki, "*Spectral clustering for a large data set by reducing the similarity matrix size*", LREC, 2008.

[13] T. Sakai and A. Imiya, "*Fast spectral clustering with random projection and sampling*", Springer, 2009.

[14] X. Chen, and D. Cai, "*Large scale spectral clustering with landmark-based representation*", AAAI, 2011.

[15] G. Miao, Y. Song, D. Zhang and H. Bai, "*Parallel spectral clustering algorithm for large-scale community data mining",* SWSM, 2008.

[16] W. Liu, J. He and S. Chang, "*Large graph construction for scalable semi-supervised learning*", ICML, 2010.

[17] W. Liu, J. Wang and S. Chang, "*Robust and scalable graph-based semisupervised learning*", IEEE, 2012.

[18] J. McQueen, "*Some Methods for Classification and Analysis of Multivariate Observations",* Symposium on Mathematical Statistics and Probability, 2015.

[19] C. Brew and S. SchulteimWalde, "*Spectral clustering for german verbs*", ACL conference, 2002.

[20] A. Ng, M. Jordan and Y. Weiss, "*On spectral clustering: Analysis and an algorithm*", Advances in neural information processing systems, 2002.

[21] G. Golub and C. VanLoan,"*MatrixComputations*", JohnsHopkins University Press, 1996.

[22] M. Zaki and W. Meira, "*Data Mining and Analysis Fundamental Concepts and Algorithms*", springer, 2014.
[23] L. Bottou, and Y. Bengio, "*Convergence properties of the k-means algorithms*", Advances in Neural Information Processing Systems, 1995.
[24] D. Arthur and S. Vassilvitskii, "*k-means++: The advantages of careful seeding*", *ACM-SIAM*, 2007.
[25] S. Tsironis and M. Sozio, "*Accurate Spectral Clustering for Community Detection in MapReduce*", Netnips, 2013.
[26] Y. Zhao and G. Karypis, "*Empirical and theoretical comparisons of selected criterion functions for document clustering*", Machine Learning, 2014.
[27] M. Afzalan and F. Jazizadeh, "*An automated spectral clustering for multi-scale data*", Neurocomputing, 2019.

Amirkabir University of Technology
(Tehran Polytechnic)

Computer Engineering and Information Technology Department

B.Sc. Thesis

Handwritten Recognition Using Sequential spectral clustering algorithm

By
Zahra Dehghanian

Supervisor
Dr.Amirmazlaghani

September 2019