راهکاری توزیع‌شده برای خوشه‌بندی کلان‌داده‌های ترکیبی

محسن محمودی و نگین دانشپور

چکیده: با توجه به سرعت روزافزون تولید اطلاعات و همچنین وجود نیازمندی تبدیل اطلاعات به دانش، نیاز به الگوریتم‌های داده‌کاوی به شدت لمس می‌شود. خوشه‌بندی یکی از تکنیک‌های داده‌کاوی است و توسعه آن سبب پیشرفت در جهت فهم بیشتر محیط پیرامون می‌شود. در این مقاله، راهکاری پویا و مقیاس‌پذیر برای خوشه‌بندی داده‌های ترکیبی با ابعاد کلان به همراه نقصان در داده‌ها ارائه گردیده است. به علت هدف‌گذاری حوزه کلان‌داده‌ها، راهکار پیشنهادی به صورت توزیع‌شده، داده‌ها را پردازش می‌کند. در این راهکار از ادغام معیارهای فاصله رایج با مفهوم نزدیک‌ترین همسایگی مشترک و همچنین به کارگیری نوعی از کدگذاری هندسی بهره برده شده است. همچنین روشی برای ترمیم داده‌های از دست رفته در مجموعه داده نیز در آن در نظر گرفته شده است.

با بهره‌گیری از تکنیک‌های موازی‌سازی و توزیع پردازش فی‌مابین گره‌های متعدد می‌توان به مقیاس‌پذیری و تسریع دست یافت. الگوریتم پیشنهادی نیز از این روش‌ها به جهت دستیابی به این مهم بهره می‌برد. ارزیابی این راهکار بر اساس معیارهای سرعت، دقت و حافظه مصرفی با مقایسه با دیگر موارد انجام می‌شود.

کلیدواژه: **اصلاح داده‌ها، پردازش توزیع‌شده، خوشه‌بندی، کلان‌داده، داده‌های ترکیبی.**

۱- مقدمه

خوشه‌بندی[1] یکی از تکنیک‌های مهم داده‌کاوی است که داده‌ها را با توجه به شباهت آنها به شکل یک کلاس‌بندی بدون نظارت که در آن کلاس‌های از پیش تعریف شده وجود ندارد، دسته‌بندی می‌کند [۱]. از کاربردهای مختلف خوشه‌بندی می‌توان بیوانفورماتیک[2]، تحلیل داده‌های وب[3]، مدیریت ارتباط با مشتری[4] و پردازش متن[5] را نام برد [۲]. با توجه به گسترش حجم داده‌ها در حوزه‌های مختلف و لزوم خوشه‌بندی آنها، این امر تبدیل به یک مسئله مهم شده است [۳]. هدف خوشه‌بندی، کلاس‌بندی سطرها است که هر یک از این سطرها، خود دارای چندین ویژگی هستند. نوع هر یک از این خصوصیت‌ها می‌تواند اسمی (قیاسی)، دودویی، ترتیبی و عددی باشد. الگوریتم‌های خوشه‌بندی با توجه به نوع داده‌های ورودی به سه دسته تقسیم می‌شوند. دسته اول و دوم به ترتیب برای داده‌های عددی و اسمی هستند. دسته سوم برای ترکیب این دو نوع داده‌ای است که داده‌های مرکب نام دارند. با توجه به این که نوع داده‌ای مرکب در دنیای واقعی پرکاربرد است، این دسته از اهمیت ویژه‌ای برخوردار است. برای تمیز قراردادن بین هر یک از خصوصیت‌های این نوع داده‌ای، از معیار فاصله استفاده می‌شود. این فاصله، بیانگر شباهت و یا عدم تشابه دو مقدار است. بدیهی است که تعریف فاصله برای نوع داده‌ای عددی، متفاوت با نوع داده اسمی است. در الگوریتم‌های خوشه‌بندی داده‌های مرکب، برای تعیین معیار فاصله مناسب، رابطه‌های مختلفی از جمله فاصله اقلیدسی بهبودیافته و فاصله منهتن ارائه شده است.

برای داده‌های قیاسی، بیشتر روش‌های تشخیص فاصله، صرفاً یک مقدار را در نظر می‌گیرند. برای مثال، نژاد فرد می‌تواند مقادیری نظیر سفید، سیاه، و یا زردپوست درنظر گرفته شود، حال آن‌که انسان‌ها و هر یک از رنگ‌های پوست خصوصیات متنوع بسیار دیگری را دارا می‌باشند و این خصوصیت‌ها صرفاً دارای یک مقدار نیستند [۴]. موقعیت شغلی و یا تفریحات فرد، چندین مقدار متفاوت دارند. یکی از نوآوری‌های الگوریتم پیشنهادی این مقاله، استفاده از کدگذاری هندسی اصلاح‌شده به منظور در نظر گرفتن چندین مقدار است.

کدگذاری هندسی استفاده‌شده در این مقاله، مشکل سرعت و دقت را دارد. نوآوری ارائه‌شده، تلاش در رفع این دو مشکل دارد. دلیل کندی سرعت در روش کدگذاری هندسی، استفاده از محاسبات مثلثاتی است که موجب افت سرعت در هر دسترسی می‌شود که با انجام پیش‌پردازش به صورت موازی و با ساختار چندنخی اصلاح شده است. همچنین راه حل‌های ارائه‌شده برای رفع مشکلات دقت این کدگذاری، شامل اصلاح ساختار هندسی و همچنین نحوه کدگذاری خصوصیت‌ها می‌باشد.

در بسیاری از موارد، محققین الگوریتمی ارائه می‌دهند که یکی از دو نوع داده قیاسی و یا عددی را پوشش می‌دهد. البته در بسیاری از الگوریتم‌ها، داده‌های اسمی را به نحوی تبدیل به نوع عددی کرده و سپس الگوریتم را اجرا می‌نمایند. این در صورتی است که در مسایل دنیای واقعی، ترکیبی از این دو نوع داده حاضر هستند [۵]. لذا هدف و تمرکز این مقاله، بررسی الگوریتم‌های خوشه‌بندی در داده‌های ترکیبی به هدف بهبود آنها بوده است.

یکی از چالش‌برانگیزترین مسایل در خوشه‌بندی، تعیین تعداد خوشه‌ها است که البته در بسیاری از الگوریتم‌ها به صورت ورودی آن، دریافت می‌شود و باید توسط کارشناس خبره تعیین گردد [۲]. این مسئله در این مقاله برای الگوریتم‌های بررسی‌شده، مورد مطالعه خواهد بود. در برخی موارد، دریافت تعداد خوشه‌ها به عنوان ورودی نقطه ضعف و در برخی، قوت آن است. الگوریتم پیشنهادی، نیازی به تعیین تعداد خوشه‌ها ندارد و به صورت خودکار، دسته‌های جدید در صورت لزوم ایجاد می‌گردند. عدم تعیین تعداد دسته‌ها، باعث حذف کارشناس خبره می‌گردد که این موضوع به عنوان یکی از مشکلات همیشگی الگوریتم K-means و الگوریتم‌های مشابه آن است [۶]. البته این ویژگی نباید باعث محدودیت گردد. در الگوریتم پیشنهادی، شاخص تراکم نیز در نظر گرفته شده که بر اساس آن الگوریتم تصمیم می‌گیرد کدام نقاط پرتراکم را به عنوان یک خوشه معین کند [۷]. یکی از نوآوری‌های شاخص این مقاله، حذف دخالت مستقیم

این مقاله در تاریخ ۲۲ شهریور ماه ۱۳۹۶ دریافت و در تاریخ ۱۲ خرداد ماه ۱۳۹۷ بازنگری شد.

محسن محمودی، دانشکده مهندسی کامپیوتر، دانشگاه تربیت دبیر شهید رجایی، تهران، ایران، (email: mahmoudi@sru.ac.ir).

نگین دانشپور (نویسنده مسئول)، دانشکده مهندسی کامپیوتر، دانشگاه تربیت دبیر شهید رجایی، تهران، ایران، (email: ndaneshpour@sru.ac.ir).

1. Clustering
2. Bioinformation
3. Web Analytics
4. CRM
5. Text-Processing

کاربر در فرایند خوشه‌بندی است. همچنین با در نظر گرفتن این که درصورت عدم وجود امکان تنظیم تراکم، کارایی الگوریتم محدود می‌گردد [۸]، در الگورتیم پیشنهادی، امکان اصلاح شاخص تراکم نیز قرار داده شده است.

علاوه بر این که در شرایط دنیای واقعی، نوع داده‌ای مرکب کاربرد دارد، ارزش حجم داده‌ها نیز پررنگ می‌شود. در بسیاری از تحقیقات، مجموعه داده‌های مورد نظر به صورت محدود انتخاب می‌شوند. این در حالی است که در کاربردهای واقعی، نیاز به پردازش حجم بالایی از داده‌ها است. لذا نیاز است که الگوریتم‌های جدیدی در این حوزه ارائه شوند که سرعت و دقت را در کنار هم افزایش بدهند. در این الگوریتم‌ها تلاش می‌شود با کاهش مرتبه زمانی الگوریتم، سرعت آن افزایش یابد و همچنین دقت آن تغییری نیابد [۹].

اطلاعات و داده‌ها هم از نظر حجم و هم از نظر تنوع در حال رشد روزافزون هستند و دنیای امروز را با چالش‌های اساسی روبه‌رو می‌کنند. علاوه بر این اطلاعات، روابط پنهان بین آنها نیز در خوشه‌بندی مورد توجه قرار می‌گیرد. این حجم بالای اطلاعات نیازمند راهکاری قدرتمند در حوزه کلان‌داده‌ها است. در این حوزه، کارهای فراوانی انجام گرفته و هر یک بر روی پردازش کلان‌داده‌ها با روش‌های مختلف، تمرکز کرده‌اند [۶]، [۷] و [۱۰] تا [۱۳]. سرعت و دقت پردازش بسیار پراهمیت است و حتی در بعضی از مسایل، نیاز به پاسخ در لحظه وجود دارد که بی‌شک با الگوریتم‌های متداول داده‌کاوی امکان دستیابی به این مهم، وجود ندارد.

الگوریتم پیشنهادی با توجه به نیازمندی‌های کلان‌داده‌ها طراحی گردیده است. در این الگوریتم که به صورت موازی اجرا می‌شود، سرعت بهبود یافته و همچنین با استفاده از بستر پردازش موازی در آن، ویژگی مقیاس‌پذیری را به ارمغان آورده است. الگوریتم پیشنهادی با توزیع پردازش به همراه توزیع داده‌ها، توانسته در حوزه کلان‌داده، داده‌های ترکیبی را خوشه‌بندی نماید.

چشم‌پوشی از عنصرهای داده‌ای مبهم، منسوخ، غیر قابل اطمینان، پراکنده و پیچیده به سادگی رخ می‌دهد. بنابراین صحت، دقت، قابلیت اطمینان و مفید به نظر رسیدن داده‌های کلان از جمله اشتباهات رایج در تصمیم‌گیری می‌باشد. روش پیشنهادی، این نوع از خطاها را پوشش می‌دهد و مدل‌هایی برای تشخیص شکل احتمالی داده‌های صحیح می‌سازد. این تفکر اصلی در مواجهه با داده‌های غیر قطعی می‌باشد که چالش اصلی در آن فراهم‌کردن نتایج قابل اطمینان با وجود عدم قطعیت است. این چالش در چند سال گذشته توجه تحقیقات زیادی را چه در بخش صنعت و چه دانشگاه به خود جلب کرده است [۱۴].

الگوریتم‌های ارائه‌شده در حوزه خوشه‌بندی داده‌های مرکب، بسیار متنوع و متفاوت هستند و هر یک بر معیاری متفاوت تمرکز دارند. برخی از آنها سرعت خوشه‌بندی را ملاک قرار داده‌اند و تا حدی از دقت چشم‌پوشی می‌کنند [۲]. برخی پویایی در پارامترهای الگوریتم را مد نظر قرار داده‌اند و یا صرفاً به ارائه راه حل جدیدی برای مسئله خوشه‌بندی داده‌های ترکیبی، اکتفا نموده‌اند [۱۵]. اما راهکار پیشنهادی در این مقاله تمامی معیارهای مذکور را در نظر دارد بدین صورت که با راهکاری نوین، سرعت، دقت و کارایی را ملاک قرار داده است. این الگوریتم بر مبنای الگوریتم ADC ارائه گردیده که به صورت ترتیبی عمل می‌کند و در پنج مرحله با بهره‌گیری از ساختار ماتریس، فرایند خوشه‌بندی داده‌های ترکیبی را انجام می‌دهد. از دیگر مزایای این الگوریتم، توانایی آن در رفع نقصان داده است. این ویژگی به دو صورت برخط و برون‌خط عمل می‌نماید که راهکار مناسبی برای پوشش خطاهای موجود در مجموعه داده‌های مختلف از جمله جریان داده[1] است. این الگوریتم برخلاف برخی الگوریتم‌ها (همانند K-Means) در برابر نویز مقاوم است که این امر مسبب کارایی گسترده آن در حوزه مختلف می‌باشد.

الگوریتم ارائه‌شده با عملکرد موازی خود می‌تواند در حوزه کلان‌داده‌ها عمل کند. همچنین با بهره‌گیری از کدگذاری هندسی، علاوه بر توانایی خوشه‌بندی داده‌های عددی، امکان خوشه‌بندی داده‌های قیاسی و ترکیبی وجود دارد.

در این مقاله، ابتدا مقدمه‌ای بر خوشه‌بندی و مفاهیم مرتبط با آن ذکر می‌گردد و سپس با طرح راهکارهای پیشین، مقدمه‌سازی برای توصیف الگوریتم ارائه‌شده انجام می‌گیرد. پس از شرح و ارائه الگوریتم جدید خوشه‌بندی، نتایج آن با کارهای پیشین، مقایسه خواهند شد.

۲- تعاریف

برای ارائه و بررسی راهکار پیشنهادی، نیاز به معرفی اجزای استفاده‌شده در آن است. در این بخش با توجه به این که در راهکار پیشنهادی از مفاهیم فاصله، کدگذاری و Map-Reduce استفاده گردیده است به ارائه تعاریف، پرداخته می‌شود.

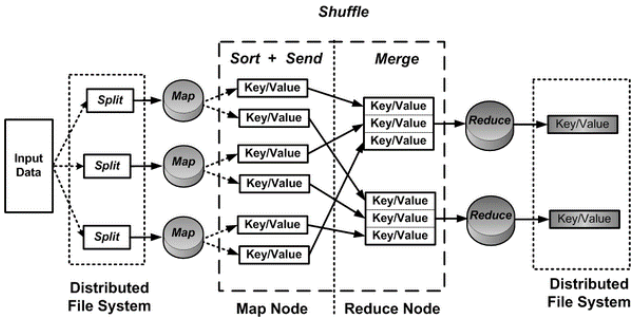
۲-۱ فاصله

فاصله فی‌مابین دو الگو برابر با مقدار اختلاف آن دو با یکدیگر است که با محاسبه ریاضی به دست می‌آید. همان طور که ذکر شد برای معیار فاصله، روابط مختلفی ارائه گردیده که یکی از فواصل که نمونه‌ای جامع است، فاصله Minkowski می‌باشد [۱۵]. شاخص Minkowski به تنهایی نمی‌تواند در داده‌های قیاسی استفاده شود و نیاز است فرایند کدگذاری داده‌ها اعمال شود. البته می‌توان از روش‌های دیگر محاسبه فاصله نیز بهره برد. برای مثال با استفاده از فاصله Gower می‌توان علاوه بر فاصله داده‌های عددی، فاصله داده‌های قیاسی را نیز محاسبه کرد [۱۶]. در الگوریتم پیشنهادی از فاصله Minkowski به همراه کدگذاری هندسی استفاده شده است.

۲-۲ Map-Reduce

Map-Reduce [۱۷] چارچوبی برای کار با داده‌های حجیم است که به خاطر سادگی آن می‌تواند با اجراکردن هزاران گره به راحتی پردازش‌ها را انجام دهد. داده‌های ورودی این چارچوب به شکل ⟨کلید، مقدار⟩ هستند که در فایل سیستمی منحصر به فردی به صورت توزیع‌شده، ذخیره گردیده‌اند. همان طور که در شکل ۱ می‌توان دید، این چارچوب شامل سه فاز اصلی نگاشت[2]، مخلوط‌کردن[3] و کاهش[4] است. برای استفاده از این منطق کافی است ورودی‌های نگاشت‌گر، تابع نگاشت و تابع کاهش تعیین گردد. در ادامه، داده‌ها با توجه به گره‌های فعال سیستم تقسیم می‌شوند و به هر نگاشت‌گر تعدادی داده به صورت زوج کلید- مقدار تعلق می‌گیرد. سپس تابع نگاشت بر روی آن اجرا و با توجه به آن، تعدادی زوج کلید- مقدار جدید ساخته می‌شود. حال سیستم بر اساس کلیدها گروه‌بندی انجام می‌دهد. این کار در بخش مخلوط‌کردن انجام می‌گیرد. در این قسمت، می‌توان از تابع مخلوط‌گر که در این مرحله بر روی داده‌ها پردازش می‌کند نیز استفاده نمود. در ادامه هر یک از گروه‌های ساخته‌شده به یک کاهنده

1. Data Stream
2. Map
3. Shuffle
4. Reduce



شکل ۱: چارچوب Map-Reduce [۱۹].

ارسال می‌شود تا با توجه به تابع کاهش نوشته‌شده، آنها را پردازش نماید. در نهایت، نتایج به صورت زوج‌های ⟨کلید- مقدار⟩ در فایل سیستم مذکور، ذخیره می‌گردند. از ویژگی‌های مشهود این سیستم می‌توان توسعه‌پذیری را نام برد. برای افزایش قدرت و توسعه این سیستم، کافی است فقط گره جدیدی به سیستم افزوده شود. این گره جدید هم به صورت نگاشت‌گر و هم به شکل کاهنده ایفای نقش خواهد کرد. اطلاعات بیشتر در مورد این چارچوب در [۱۸] قابل دسترسی است.

۲-۳ کدگذاری

همان طور که ذکر شد داده‌های قیاسی باید به روش‌های مختلفی قابل مقایسه گردند تا بتوان آنها را نیز در رابطه فاصله دخیل نمود. ساده‌ترین روش ممکن برای این امر، بررسی تشابه متن دو خصوصیت[1] است که اگر کاملاً با یکدیگر مساوی باشند، نتیجه مقایسه ۱ و در غیر این صورت، ۰ خواهد بود. روش دیگر برای این تبدیل، استفاده از کدگذاری[2] است. کدگذاری برای تبدیل نوع داده قیاسی به نوع عددی استفاده می‌شود که باید ویژگی‌های مجموعه قیاسی در این تبدیل حفظ گردد. در واقع کدگذاری خصوصیت x_i را به مقدار y_i تغییر می‌دهد [۱۵]. الگوریتم‌های این تغییر، codification نام دارد. اگر الگوریتمی مختص داده‌های اسمی باشد می‌تواند به راحتی برای استفاده در حوزه داده‌های عددی نیز به کار رود. ولی چون این الگوریتم برای همه انواع داده‌های اسمی کاربرد ندارد، لذا برای خوشه‌بندی داده‌های مرکب، راه حل جامع دیگری استفاده می‌گردد. روش‌های مختلفی برای کدگذاری وجود دارد که در ادامه کدگذاری‌های ترتیبی، گری، دودویی و هندسی معرفی می‌گردند [۱۵].

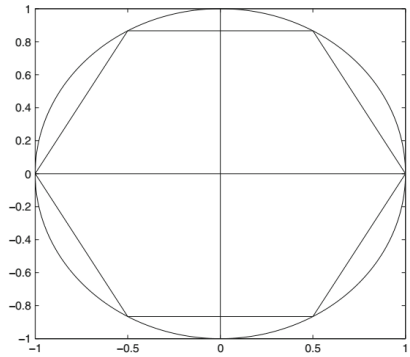
کد ترتیبی: در این روش، یک عدد صحیح به هر یک از مقادیر موجود در دامنه خصوصیت اسمی اختصاص داده می‌شود.

کد گری: این نوع کدگذاری که بر پایه اختلاف یک بیتی بین دو کد متوالی است نیز می‌تواند برای هر یک از مقادیر موجود در دامنه استفاده شود. معایب این روش، تعداد بیت‌های زیاد، مرتبه زمانی بالا و متغیربودن طول رشته است.

کدگذاری دودویی: در این روش به هر یک از مقادیر محتمل در دامنه خصوصیت، یک مرتبه عددی (در مبنای دو) اختصاص داده می‌شود. معایب این روش، تعداد بیت‌های زیاد، متغیربودن طول رشته و خاص‌بودن معیار فاصله است.

کدگذاری هندسی: مبنای این روش [۲۰]، استفاده از مختصات دوقطبی می‌باشد و فضای دوبعدی همان دامنه مقادیر است. همان طور که می‌توان در شکل ۲ مشاهده نمود، یک دایره در فضایی

1. Simple Matching
2. Data Codification



شکل ۲: مختصات دوقطبی- خصوصیت با شش دسته مقدار [۲۰].

دوبعدی قرار گرفته و محیط آن به تعداد دسته‌های خصوصیت و به فواصل مساوی، تقسیم شده است. منظور از فاصله مساوی این است که فاصله بین دو نقطه در محیط آن، یکسان است.

این کدگذاری دارای دو ویژگی است:

اولاً تعداد دسته‌های خصوصیت مورد نظر، مهم نیست. با کدکردن خصوصیت‌ها، برخلاف دیگر کدگذاری‌ها که تعداد کدهای تولیدی، بسیار زیاد خواهند بود، آن خصوصیت به دو عدد صحیح در مختصات قطبی تبدیل می‌شود. در کدگذاری باینری برای کدکردن n خصوصیت، نیاز به n بیت است که متناظر با n بُعد خواهد بود ولی در این کدگذاری ۲ عدد که متناظر با دو بُعد هندسی است، برای نمایش مختصات کافی می‌باشد. با توجه به کاهش تعداد ابعاد، سرعت خوشه‌بندی افزایش می‌یابد.

ثانیاً تمام دسته‌ها از ارزش یکسان برخوردار هستند. این در حالی است که در کدگذاری ترتیبی فقدان این مسئله کاملاً مشهود است. همگن و متعادل‌بودن نقاط، یک ویژگی مناسب برای خوشه‌بندی داده‌های اسمی است.

رابطه (۱) چگونگی محاسبه نقطه جدید در کدگذاری هندسی که در آن N تعداد دسته‌های مختلف و i یکی از آن دسته‌ها است را نشان می‌دهد. همان طور که قبلاً ذکر گردید، به علت کاربرد این کدگذاری در راهکار پیشنهادی، در ادامه به بررسی کدگذاری هندسی پرداخته می‌شود

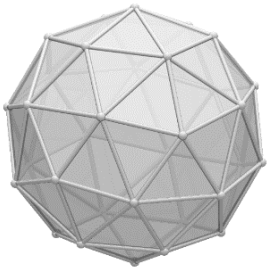
$$Cat_i = [\cos\frac{(i-1)\times 2\times\pi}{N}, \sin\frac{(i-1)\times 2\times\pi}{N}] \qquad (1)$$

برای مثال خصوصیتی که فقط دو دسته دارد را در نظر بگیرید. دسته اول ($i=0$) با دوتایی [۱،۰] و دسته دوم ($i=1$) با دوتایی [۱-،۰] کد می‌شود. مشکل این روش وقتی نمایان است که تعداد قطاع‌های دایره زیاد می‌شود، زیرا با توجه به نوع محاسبات و نزدیکی اعداد تولیدشده توسط الگوریتم، دقت پایین می‌آید. در واقع هرچه تعداد قطاع‌ها بالاتر برود به همان نسبت خطا هم افزایش می‌یابد.

راه حل این مشکل، استفاده از ابعاد بالاتر هندسی است. برای مثال می‌توان به جای دایره و فضای دوبعدی، از کره و فضای سه‌بعدی استفاده نمود که در آن یک چندسطحی، محاط شده است. برای مثال می‌توان یک icosahedrons [۲۱] (شکل ۳) را معرفی کرد.

با در نظر گرفتن فضای سه‌بعدی، از خطا کاسته می‌شود [۲۰] ولی متأسفانه راه حل کاملی برای مشکل ذکرشده در مطالعات انجام‌گرفته در این حوزه وجود ندارد و فقط با افزایش بعد، می‌توان مشکل را به صورت مقطعی حل نمود.

با کدگذاری هندسی می‌توان دو خصوصیت را با یکدیگر مقایسه نمود. برای مثال فرض کنید خصوصیت x به مختصات [۵،۱،۰] و y به مختصات



شکل ۳: icosahedrons.

[۰٫۷,۰٫۸-] کد شده است. حال می‌توان به راحتی، این فاصله را به وسیله شاخص اقلیدسی حساب کرد (برابر با ۱٫۲).

۳- کارهای پیشین

در این بخش، الگوریتم‌های خوشه‌بندی داده‌های ترکیبی مرتبط با این مقاله، بررسی می‌گردد. همچنین بعد از مطالعه هر یک، نقاط ضعف آنها که در طراحی الگوریتم جدید مد نظر بوده‌اند نیز ارائه می‌گردد. اولین الگوریتم، PKMeans [۱۹] است که نسخه موازی‌شده K-Means [۲۲] می‌باشد. PKMeans به وسیله چارچوب Map-Reduce، موازی‌سازی را انجام می‌دهد. این الگوریتم سه بخش نگاشت، ترکیب و کاهش دارد و به صورت تکراری[1] اجرا می‌شود. در هر گام از تکرار، خوشه‌ها به روز خواهند شد و در صورتی که فرایند به روز رسانی تغییری در شرایط خوشه‌ها ایجاد نکند، کار الگوریتم تمام شده است. تابع نگاشت‌گر برای هر یک از ورودی‌ها، فاصله تا تمام خوشه‌ها را محاسبه می‌کند و تابع کاهنده این خوشه‌ها را به روز می‌نماید. به جهت کاهش هزینه ارتباطات اضافی در طی اجرای مراحل فرایند از یک تابع مخلوط‌گر برای دسته‌بندی خروجی‌های نگاشت‌گر استفاده می‌شود.

این الگوریتم، همانند نسخه اصلی K-Means همچنان با معضل نویز و تأثیر آن بر نتایج، مواجه است. البته دیگر مشکلات K-Means را نیز به ارث برده و محدودیت شکل خوشه‌ها، ناتوانی در پردازش داده‌ها با چگالی‌های مختلف و دقت کم از دیگر مشکلات آن است.

در برخی الگوریتم‌ها، خصوصیات فقط می‌توانند دارای یک مقدار باشند اما در دنیای واقعی، این تک‌مقداری، غیر قابل قبول است. به نحوی باید خصوصیات چندمقداری را در نظر گرفت. اگر چنین کاری با الگوریتم‌های داده‌کاوی معمول انجام شوند مشکلی تحت عنوان خصوصیات ساختگی پیش خواهد آمد که عملاً در آن داده‌ها فاقد ارزش هستند. در ادامه الگوریتم SV-k modes [۴] با هدف خوشه‌بندی داده‌های قیاسی بر اساس مقادیر دسته‌ای ارائه شده است که در آن تابع فاصله میان دو شیء بر اساس مقادیر دسته‌ای تعریف شده است. یک الگوریتم کاشف[2] برای به روز رسانی مرکز دسته‌ها با استفاده از یک روند خوشه‌بندی تکراری به همراه الگوریتمی به منظور مقداردهی اولیه مراکز خوشه‌ها توسعه داده شده است.

الگوریتم GDD[3] [۸] بر اساس شاخص فاصله و چگالی الگوها در فضای نمونه، ارائه گردیده است. اولین نوآوری مقاله، یافتن بهترین خوشه‌های احتمالی بدون اطلاعات و پارامترهای اولیه می‌باشد. دومین نوآوری آن در این است که در صورت اجرا بر روی داده‌های دوبعدی، خوشه‌هایی بسیار نزدیک به ادراک بشری انتخاب می‌شوند و به عبارتی دیگر نتیجه نزدیک به نتایج انسانی می‌باشد. GDD شباهت‌های زیادی با الگوریتم‌های داده‌کاوی معمول دارد و از هسته گاوس و فاصله‌ها به منظور ایجاد خوشه‌ها بر اساس شکل و چگالی داده استفاده می‌کند. از آنجا که GDD قبل از اجرا به پارامتر خاصی نیاز ندارد، دفعات اجرا بر نتیجه تأثیر نگذاشته و نتیجه یکسان خواهد بود.

الگوریتم KSC-LCA [۱۶] که نسخه اصلاح‌شده SC-LCA می‌باشد، اقتباسی از برنامه‌ریزی برای مسابقات لیگی است. این الگوریتم، مرکز خوشه‌ها را بر اساس فاصله خوشه‌های داخلی و همچنین یک رابطه مقایسه‌ای برای هر دو نوع قیاسی و عددی به دست می‌آورد. در [۱۶] سه بخش اصلی معرفی می‌گردد. در بخش اول، الگوریتم جستجو بر اساس لیگ‌های مسابقات Chaotic معرفی می‌گردد. دومین بخش، روش جدیدی برای جستجوی خوشه‌ها معرفی می‌نماید. در سومین بخش، مقیاس رابطه Gower و روندی برای به کارگیری داده‌های عددی و قیاسی ارائه می‌گردد.

الگوریتم MR-DBSCAN [۲۳] نیز همچون الگوریتم قبلی با موازی‌سازی یک الگوریتم قدیمی [۲۴]، راهکار نوینی برای خوشه‌بندی در کلان‌داده‌ها ارائه نموده است. DBSCAN [۲۴] یک روش خوشه‌بندی است و مزیت این روش به نسبت روش‌های دیگر خوشه‌بندی مانند K-Means این است که نسبت به شکل خوشه‌ها حساس نمی‌باشد و می‌تواند اشکال غیر منظم را نیز در داده‌ها تشخیص دهد. این الگوریتم نیز همانند دیگر روش‌های خوشه‌بندی نیازمند روشی برای یافتن فاصله داده‌ها است. یکی از مشکلات این الگوریتم این است که نقاط مرزی که می‌توانند در دو خوشه نیز باشند، ممکن است به هر یک از خوشه‌ها تعلق گیرند. این الگوریتم در ۴ گام اجرا می‌شود و به جهت بهبود سرعت این الگوریتم، گام چهارم (ادغام) به صورت موازی اجرا می‌گردد. برای این منظور از چارچوب MapReduce استفاده گردیده است.

بر مبنای الگوریتم K-Prototypes [۲۵]، الگوریتم AMR K-prototypes [۲۶] ارائه گردیده و آن را با استفاده از چارچوب MapReduce موازی و مقیاس‌پذیر نموده است. در الگوریتم K-Prototypes هدف پیداکردن K خوشه است.

ابتدا K خوشه را به صورت تصادفی انتخاب می‌کند و به مرور زمان با اندازه‌گیری فاصله هر یک از الگوها با مرکز هر خوشه، عملیات خوشه‌بندی را انجام می‌دهد. علاوه بر این که با پیاده‌سازی الگوریتم K-Prototype در چارچوب MapReduce، سرعت بهبود یافته و الگوریتم مقیاس‌پذیر گردیده، تغییری در معیار فاصله نیز ایجاد شده است. ایده این اصلاح، استفاده از نامساوی مثلثاتی به جهت کاهش مقایسات است. این الگوریتم، مشکلات K-Prototype را همچنان دارد و پیش‌پردازش‌های مورد نیاز و همچنین دقت کم آن در مقایسه با دیگر الگوریتم‌هایی که با همین سرعت کار می‌کنند از جمله این مشکلات است.

الگوریتم‌های بررسی‌شده پیشین، همه به صورت تکراری بودند. الگوریتم ADC [۲۷]، فرایند خوشه‌بندی را به صورت مرحله‌ای انجام می‌دهد. این الگوریتم، نسخه اصلاح‌شده الگوریتم ASC [۲۸] است که تحت عنوان "روش پویا و سریع برای خوشه‌بندی داده‌های مرکب" معرفی شده[4] و برای داده‌های مرکب با ابعاد کلان[5] که در آنها برخی از خصوصیت‌ها در دسترس نیست، کاربرد دارد. الگوریتم ADC، پایه الگوریتم پیشنهادی است و لذا به صورت کامل بررسی خواهد شد.

1. Iterative
2. Heuristic
3. Gaussian Density Distance
4. https://github.com/MohsenMahmoudi/ADC.git
5. Big Data

ADC(Patterns a){

Matrix AAM = $\begin{bmatrix} \infty & \cdots & dst(x_1,x_p) \\ \vdots & \ddots & \vdots \\ dst(x_p,x_1) & \cdots & \infty \end{bmatrix}$;

Find_Nearset_Neighbors (K);

Matrix AAM = $\begin{bmatrix} 0 & \cdots & CNN(x_1,x_m,k) \\ \vdots & \ddots & \vdots \\ CNN(x_m,x_1,k) & \cdots & 0 \end{bmatrix}$;

Create_Cluster

$$\left(\text{List\_Sorter}\left(\frac{dst\left(x_i^{\Pi},x_p\right)}{CNN\left(x_i^{\Pi},x_p,K\right)},\text{desc}\right)[0]\right);$$

$$\text{Repair\_Data}\left(x_{i,j}^{\Pi},\frac{\left(\sum_{r=1}^{R} CNN(x_i^{\Pi},x_r^{C_{i,j}},K)x_{r,j}^{C_{i,j}}\right)}{\sum_{r=1}^{R} CNN(x_i^{\Pi},x_r^{C_{i,j}},K)}\right);$$

}

شکل ۴: الگوریتم ADC.

در این بخش، ابتدا معیار سنجش شباهت خصوصیت‌های داده‌های مرکب MASM[1] در (۲) معرفی می‌شود

$$dst(x_i,x_j)=\frac{\sum_{n=1}^{N} d_{i,j}^{(n)}\delta_{i,j}^{(n)}}{\sum_{n=1}^{N}\delta_{ij}^{(n)}}\ ,\ (j=\overline{1,M}) \qquad (۲)$$

که N تعداد خصوصیت‌ها و δ نمایشگر عدم حضور یکی از الگوها است و در صورت عدم حضور، δ برابر صفر می‌شود. M نیز برابر تعداد الگوها در مجموعه داده است. d به دو صورت برای خصوصیت‌های اسمی و عددی معرفی می‌شود. برای داده‌های اسمی به صورت (۳) معرفی می‌گردد

$$d_{i,j}^{(n)}=\begin{cases}0 & \text{if } x_i=x_j\\ 1 & \text{otherwise}\end{cases} \qquad (۳)$$

و برای داده‌های عددی به صورت (۴) خواهد بود

$$d_{i,j}^{(n)}=\frac{\left|x_{i,n}-x_{j,n}\right|}{\arg_{(p=1,\ldots,P_n)}\max(x_{p,n})-\arg_{(p=1,\ldots,P_n)}\min(x_{p,n})} \qquad (۴)$$

در این رابطه، فاصله نرمال‌شده بین خصوصیت‌های عددی دو الگو محاسبه می‌گردد. $x_{i,n}$ یکی از خصوصیت‌های یک الگو است که در صورت کسر، فاصله بین دو خصوصیت را محاسبه می‌کند و در مخرج کسر، بیشترین فاصله بین آن دو خصوصیت در میان تمام الگوهای عددی موجود در مجموعه داده می‌باشد.

در ادامه به بررسی الگوریتم ADC پرداخته می‌شود. این الگوریتم شامل ۵ گام است. در بسیاری از این مراحل، استفاده از ماتریس برای نگهداری اطلاعات و پردازش پیشنهاد شده است. استفاده از پردازنده گرافیکی و همچنین برخی اصلاحات در ساختار نگهداری ماتریس، برای افزایش بهره‌وری پیشنهاد می‌شوند [۲۷]. در شکل ۴ می‌توان مراحل اجرای الگوریتم ADC را ملاحظه نمود.

در مرحله اول الگوریتم ADC، ماتریس مجاورتی[2] (AM) طبق (۵) برای مقایسه هر جفت ممکن در مجموعه داده ساخته می‌شود

$$AM=\begin{bmatrix} \infty & \cdots & dst(x_1,x_p) \\ \vdots & \ddots & \vdots \\ dst(x_p,x_1) & \cdots & \infty \end{bmatrix} \qquad (۵)$$

در ادامه K همسایه نزدیک هر الگو یافت می‌شود که این بخش بیشترین هزینه پردازش را دارد. همان طور که قبلاً پیشنهاد شد، استفاده از روش‌های مختلف بهبوددهی می‌تواند سرعت را افزایش دهد.

حال باید ماتریس تعداد همسایه‌های مشترک[3] $(CNNM)$، طبق (۶) حساب گردد

$$CNNM=\begin{bmatrix} 0 & \cdots & CNN(x_1,x_m,k) \\ \vdots & \ddots & \vdots \\ CNN(x_m,x_1,k) & \cdots & 0 \end{bmatrix} \qquad (۶)$$

که در آن $CNN(x_i,x_j,k)$ تعداد k همسایه مشترک بین دو الگوی x_i,x_j است.

حال برای هر یک از خصوصیت‌های پاک‌شده، ابتدا لیستی از فواصل (با ترتیب نزولی)، بین الگوی مورد نظر $(x_{i,j}^{\Pi})$ و $\theta_{i,j}^{\Pi}$ (الگوهایی که خصوصیت مورد نظر را دارند) طبق (۵) تهیه می‌شود

$$\theta_{i,j}^{\Pi}=sort_{desc}\left\{\frac{dst(x_i^{\Pi},x_p)}{CNN(x_i^{\Pi},x_p,K)}\right\} \qquad (۷)$$

با انتخاب R مقدار اول در لیست، یک خوشه جدید تشکیل می‌گردد. در نهایت برای بازیابی داده‌ها به جای چشم‌پوشی و یا پرکردن با مقادیر پیش‌فرض (که در بسیاری از تحقیق‌ها پیشنهاد می‌شود)، راهکار ویژه‌ای استفاده می‌شود. در این راهکار با بررسی هر یک از الگوها و اعمال وزن همسایگی بر مقدار خصوصیت مورد نظر (که در الگوی مورد نظر، در دسترس نیست) میانگینی ساخته می‌شود. این میانگین به خاطر وزن‌دهی‌ها، بسیار نزدیک به مقادیر واقعی است. با انجام آزمایش بر روی داده‌های بدون نویزی که برخی از خصوصیت‌های آنها پاک شده‌اند به طور میانگین[4]، دقت ۶۰٪ حاصل شده است[5]

$$x_{i,j}^{\Pi}=\frac{\sum_{r=1}^{R} CNN(x_i^{\Pi},x_r^{C_{i,j}},K)x_{r,j}^{C_{i,j}}}{\sum_{r=1}^{R} CNN(x_i^{\Pi},x_r^{C_{i,j}},K)} \qquad (۸)$$

که $x_{i,j}^{\Pi}$ مقدار جایگزینی برای خصوصیتی است که در دسترس نیست.

این الگوریتم دو مشکل اصلی دارد، اولاً که به خاطر نگهداری اطلاعات پردازش در ساختمان داده ماتریس در هر گام، میزان حافظه مصرفی بسیار زیاد است، به نحوی که علی‌رغم بهبودهای متعدد، نباید حداکثر تعداد الگوهای موجود در مجموعه داده از ۱۰۰۰۰ تجاوز کند. در این آستانه، حدود ۱/۲ گیگابایت حافظه مصرف می‌شود. مشکل بعدی این الگوریتم، ساختار سلسله‌مراتبی گام‌ها و لزوم رعایت توالی آن است که باعث کاهش شدید مقیاس‌پذیری می‌شود.

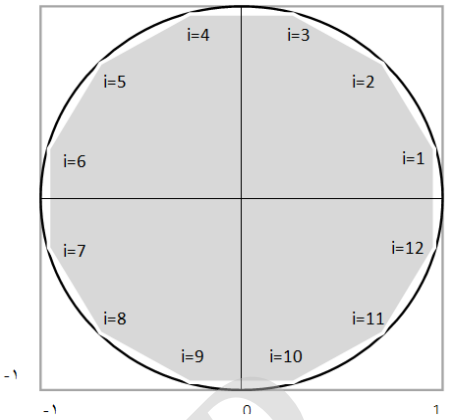
٤- الگوریتم پیشنهادی

در این مقاله الگوریتمی ارائه گردیده که سعی شده راهکاری برای پردازش کلان‌داده‌ها باشد. در این حوزه از پردازش، چندین نیاز اصولی

1. Mixed-Type Attribute Similarity Measure
2. Adjacency Matrix
3. Common Nearest Neighbor's Matrix

۱. در dataset با ۵۰، ۱۰۰۰ و ۳۰۰۰۰۰ الگو.
۲. هر دو نوع داده اسمی و عددی مد نظر است.

شکل ۵: اجزای مختلف راهکار پیشنهادی.



شکل ۶: تخصیص موقعیت ۱۲ دسته به فضای دوبعدی دایره.

وجود دارد. با توجه به گستردگی حجم اطلاعات و نیاز به پردازش آنها در زمان محدود، چندین معضل به وجود می‌آید. اول آن که در بسیاری از موارد، طراح الگوریتم، دقیقاً نمی‌داند با چه میزان و درجه از حجم داده روبه‌رو است. این مسئله برآمده از گسترش روزافزون تولید داده است. لذا الگوریتم باید بدون صرف هزینه‌هایی مثل اصلاح الگوریتم، مقیاس‌پذیر باشد. این ویژگی معادل مفهوم Plug and Play است که در آن سیستم باید بتواند صرفاً با افزودن یک گره پردازشی دیگر، مقیاس کاری الگوریتم را به شکل چشم‌گیری بیفزاید [۱۸]. هرچند که امروزه با حضور ابزاری همچون Hadoop دست‌یابی به این مهم، چندان دشوار نیست ولی هنوز با چالش‌هایی مواجه است. یکی از این چالش‌ها، تضاد ذاتی برخی از الگوریتم‌های ترتیبی با چارچوب Map-Reduce است. در الگوریتم‌هایی همچون PK-Means تشابه ساختار، کاملاً مشهود بوده و دستاوردهایی بسیار وسیع، حاصل گردیده است اما صرفاً تغییر الگوریتم‌ها به جهت اجرا بر روی چند پردازنده و گره پردازشی، مسبب مقیاس‌پذیری و تسریع نخواهد شد. شایان ذکر است که در بسیاری از موارد، حتی موجب کاهش بهره‌وری و حتی دقت نیز می‌شود.

راهکار پیشنهادی سه بخش مجزا دارد (شکل ۵). بخش اول مربوط به پیش‌پردازش داده‌ها و کدگذاری خصوصیت‌های قیاسی می‌باشد. برای این منظور الگوریتم کدگذاری بهبودیافته‌ای بر اساس کدگذاری هندسی ارائه گردیده است. در بخش دوم، الگوریتم خوشه‌بندی ADC به دو روش به صورت توزیع‌شده و موازی بهبود یافته است. در بخش آخر با استفاده از خوشه‌بندی انجام‌شده، مقادیر ناموجود، بازیابی می‌شوند. نحوه بازیابی همچون الگوریتم ADC است که در آن بر اساس دیگر الگوهای موجود در خوشه، مقدار مناسب برای آن خصوصیت تعیین می‌گردد. با توجه به این که روند انجام بازیابی بدون تغییر و مطابق الگوریتم ADC است از تکرار آن خودداری می‌گردد.

۴-۱ کدگذاری هندسی بهبودیافته

با بررسی بخش نتایج مقاله کدگذاری هندسی [۲۰]، دو مشکل سرعت و دقت محرز است. بروز این مشکلات در حجم محدودی از داده‌ها، نمایان نیست اما در حوزه کلان‌داده‌ها، به علت وجود تنوع و حجم بالای داده، نمی‌توان از آنها چشم‌پوشی نمود.

جدول ۱: نمونه مجموعه داده شامل خصوصیت قیاسی.

Age	Gender	illness
۱۹	M	Hypertension, Smokes
۲۴	F	Diabetes, Alcoholism
۴	F	Handcap
۵	M	Diabetes

جدول ۲: لیست ترکیب‌های مختلف با ترتیب افزایشی.

i	Binary	Illness
۱	۰۰۰۰۱	Handcap
۲	۰۰۰۱۰	Alcoholism
۳	۰۰۰۱۱	Alcoholism, Handcap
...	...	...
۲۹	۱۱۱۰۱	Diabet, Alcoholism, Hypertension, Smokes
۳۰	۱۱۱۱۰	Diabet, Alcoholism, Hypertension, Handcap
۳۱	۱۱۱۱۱	Diabet, Alcoholism, Hypertension, Handcap, Smokes

فرایند کدگذاری داده‌های قیاسی با اکتشاف دامنه مقادیر شروع می‌شود. در این فاز، تمام مقادیر محتمل برای آن خصوصیت با نوع قیاسی استخراج شده و بر اساس آنها کد تعریف می‌شود.

در جدول ۱ نمونه مجموعه داده شامل خصوصیت قیاسی آورده شده است. ستون illness شامل لیست بیماری‌های فرد و ستون دوم نشان‌گر جنسیت فرد است.

مجموعه مقادیر ستون illness به صورت لیست زیر است:

- Hypertension
- Smokes
- Diabetes
- Alcoholism
- Handcap

همچنین برای ستون جنسیت، مقادیر زیر استخراج می‌شود:

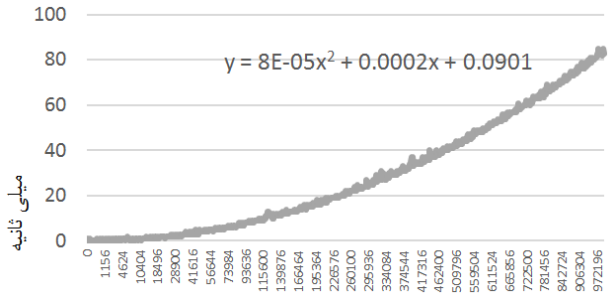
- M
- F

حال باید به هر ترکیبی از مقادیر دامنه هر خصوصیت قیاسی، شناسه‌ای تخصیص داده شود. برای خصوصیت بیماری، تعداد این ترکیب برابر $۲^۵ - ۱$ خواهد بود. به هر یک از این ترکیب‌ها می‌توان مانند جدول ۲ شناسه اختصاص داد و با این ترکیب می‌توان به جای استفاده از یک خصوصیت، مانند [۴] از چندین خصوصیت استفاده نمود.

حال هر یک از این شاخص‌ها در صفحه دوبعدی نگاشت می‌گردند که در شکل ۶، این نگاشت نمایش داده شده است. در این نگاشت یک مشکل بزرگ وجود دارد. با توجه به جدول ۲ اولین دسته و آخرین دسته را در نظر بگیرید.

طبق کدگذاری هندسی، این دو دسته فاصله بسیار کمی با یکدیگر دارند اما در واقع اختلاف آنها برابر $n-۱$ که بیشترین فاصله ممکن می‌باشد، است. برای رفع این مشکل، ترتیب کدگذاری را باید اصلاح نمود. به جای استفاده از ترتیب دودویی هر یک از مقدارهای دامنه به صورت افزایشی، می‌توان آن را به شکل دیگری لیست کرد. ایده اصلی این لیست جدید، استفاده از گری کد است. ویژگی اصلی گری کد، اختلاف یک بیتی در اعداد متوالی است.

همان طور که در جدول ۳ می‌توان مشاهده نمود، مشکل ترتیب در



شکل ۷: نمودار رشد زمان بر اساس تعداد درخواست‌ها.

جدول ۳: اصلاح ترتیب کدگذاری.

i	Gray	Illness
۱	۰۰۰۰۱	Handcap
۲	۰۰۰۱۱	Alcoholism, Handcap
۳	۰۰۰۱۰	Alcoholism
...	...	...
۲۹	۱۰۰۱۱	Hypertension, Alcoholism, Handcap
۳۰	۱۰۰۰۱	Hypertension, Handcap
۳۱	۱۰۰۰۰	Hypertension

کدگذاری رفع شده و فاصله بین دو خانه مجاور، کاملاً یکسان خواهد بود. این اصلاح، بدون تغییر در ماهیت اصلی کدگذاری اعمال شده و لذا تأثیری بر سرعت آن نخواهد داشت و صرفاً مشکل مقادیر مرزی در دامنه را رفع می‌کند.

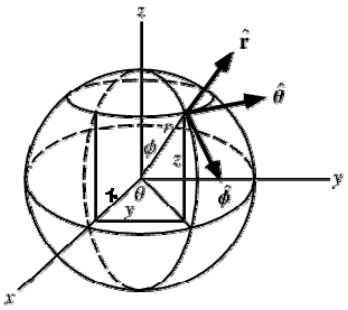
در این بخش از مقاله، برای رفع محدودیت سرعت و همچنین افزایش دقت، بهبودهایی برای این کدگذاری پیشنهاد می‌گردد. در ادامه به صورت مجزا هر یک از این بهبودها بررسی خواهد شد.

٤-١-١ سرعت

یکی از معضلات اصلی کدگذاری هندسی، سرعت آن است. محاسبات مثلثاتی در این کدگذاری باعث افت سرعت در هر دسترسی خواهد شد. در حالی که کدگذاری‌های دودویی و گری به خاطر استفاده از محاسبات بیتی، بسیار سریع (در حدود ۱۰ میلی‌ثانیه) انجام می‌شوند، این کدگذاری در تعداد ۱۰۰۰۰ دسترسی، حدود ۱۰۰ میلی‌ثانیه زمان می‌برد (شکل ۷).

برای رفع این مشکل، لازم است بخشی از کار، قبل از شروع خوشه‌بندی انجام گیرد. در صورتی که در طی مراحل کدگذاری، محاسبات هندسی مربوط انجام گیرد، لازم است به تعداد دفعات مورد نیاز، توابع هندسی فراخوانی گردند. این در حالی است که اگر در ابتدای امر، برای هر یک از دسته‌های موجود در دامنه، این محاسبات انجام شده و در حافظه ذخیره گردند، به میزان قابل توجهی زمان اجرا کاهش می‌یابد. باید در نظر داشت که اگر نتایج در یک ساختار جدول درهم‌سازی[1] نگهداری گردند، مرتبه هر دسترسی به $O(1)$ تقلیل می‌یابد. البته زمان مورد نیاز برای پیش‌پردازش را نیز باید در نظر داشت. اگر پس از اکتشاف دامنه، ۱۶ مقدار مختلف استخراج شود، $۲^{۱۶} = ۶۵۵۳۶$ دسته مختلف برای کدگذاری، موجود خواهند بود. شایان ذکر است که انجام این پیش‌پردازش، کاملاً به صورت موازی و در ساختار چندنخی پیاده‌سازی شده است. لذا این بخش از فرایند نیز بسیار سریع انجام می‌گیرد، هرچند که انجام این کار، فقط یک بار و به صورت برون‌خط[2] است.

1. Hash Table
2. Offline



شکل ۸: مختصات کروی.

جدول ۴: کدگذاری کروی با ۱۶ دسته.

	θ_1	θ_2	θ_3	θ_4
φ_1	۱	۲	۳	۴
φ_2	۵	۶	۷	۸
φ_3	۹	۱۰	۱۱	۱۲
φ_4	۱۳	۱۴	۱۵	۱۶

٤-١-٢ دقت

همان طور که در کدگذاری هندسی [۲۰] ذکر گردید با افزایش تعداد دسته‌ها به خاطر محدودیت در دقت محاسبات اعشاری، از دقت کدگذاری کاسته می‌شود. در مقاله اصلی راه حل این مشکل، استفاده از فضای سه‌بعدی مطابق شکل ۸ به جای فضای دوبعدی آورده‌شده در شکل ۲ است [۲۰]. با استفاده از کُره[3] می‌توان تا حدی مشکل دقت را رفع کرد.

مختصات کروی (۸) شامل سه بعد x، y و z است که برای آنها، (۹) برقرار است

$$\begin{aligned} x &= \sin\theta\cos\varphi \\ y &= \sin\varphi\cos\theta \\ z &= \cos\theta \end{aligned} \qquad (۹)$$

لذا برای کدگذاری در این مختصات، از (۱۰) استفاده می‌شود

$$Cat_i = [(\sin\theta, \cos\varphi), (\sin\varphi, \cos\theta), (\cos\theta)] \qquad (۱۰)$$

کدگذاری در فضای دوبعدی، صرفاً وابسته به یک متغیر (θ) بود و با جایگزینی $((i-۱)\times۲\times\pi)/N$ به جای آن، تمام دسته‌ها نمایش داده می‌شد. اما در فضای کروی، متغیر φ نیز دخیل است. برای مقداردهی آن از ایده آرایه دوبعدی و نحوه تبدیل آن به آدرس‌های حافظه، استفاده شده است. برای مثال اگر تعداد دسته‌ها ۱۶ باشد، همانند جدول ۴ می‌توان θ و φ را مقداردهی نمود. حال اگر باز هم تعداد دسته‌ها افزایش یابد، مجدداً مشکل ذکرشده نمایان می‌گردد. در این بخش از مقاله، راه حلی جامع برای کدگذاری به تعداد نامحدود دسته، پیشنهاد می‌گردد. ابتدا باید رابطه‌ای برای معرفی تمام ابعاد پیشنهاد گردد. در (۱۱) با تعیین n به عنوان بُعد مورد نظر، می‌توان روابط مختصاتی آن را کسب نمود [۲۹]

$$\begin{aligned} x_1 &= r, \cos\varphi_1 \\ x_2 &= r, \sin\varphi_1, \cos\varphi_2 \\ x_3 &= r, \sin\varphi_1, \sin\varphi_2, \cos\varphi_3 \\ &\cdots \\ x_{n-1} &= r, \sin\varphi_1, \ldots, \sin\varphi_{n-2}, \cos\varphi_{n-1} \\ x_n &= r, \sin\varphi_1, \ldots, \sin\varphi_{n-2}, \sin\varphi_{n-1} \end{aligned} \qquad (۱۱)$$

3. Sphere

γ_2	θ_1	θ_2	θ_3	θ_4
φ_1	17	18	19	20
φ_2	21			
φ_3	25			
φ_4	29			

γ_1	θ_1	θ_2	θ_3	θ_4
φ_1	1	2	3	4
φ_2	5	6	7	8
φ_3	9	10	11	12
φ_4	13	14	15	16

شکل ۹: آرایه سه‌بعدی.

با توجه به این که هرچه بُعد افزایش می‌یابد، تعداد متغیرها نیز افزایش یافته است، باید چاره‌ای تمهید کرد. راه حل پیشنهادی همانند راه حل در فضای کروی و استفاده از آرایه $n-1$ بعدی است. با فرض داشتن یک فضای ۳بعدی و با توجه به (۱۱)، شکل ۹ و (۱۲) پیشنهاد می‌گردند

$$x_1 = r,\cos\theta$$
$$x_2 = r,\sin\theta,\cos\varphi \qquad (۱۲)$$
$$x_3 = r,\sin\theta,\sin\varphi,\cos\gamma$$

به این صورت، الگوریتم به نحوی تغییر یافته است که در آن به صورت خودکار، بُعد مورد نیاز تعیین گشته و سپس ساختارهای لازم، ایجاد می‌گردند. این در حالی است که نیاز به تنظیم هیچ پارامتری نیست و به نسبت تعداد دسته‌ها، بُعد تعیین می‌گردد.

PADC ۲-٤

این الگوریتم توسعه‌ای بر الگوریتم ADC است و در ۵ گام انجام می‌گیرد. الگوریتم ADC متکی بر استفاده از ساختار ماتریسی است. استقلال داده‌های ماتریسی امکان توزیع پردازش را فراهم می‌نماید و همچنین امکان استفاده از پردازنده‌های خاص‌منظوره در محاسبات ماتریسی، عامل انتخاب این الگوریتم بوده است. این مهم، باعث کارایی بالای الگوریتم در حوزه کلان‌داده‌ها می‌گردد. با توجه به این که تمام فرایند این الگوریتم در حافظه نگهداری می‌شود، محدودیت حجم محاسبات، بسیار چشم‌گیر است. لذا در الگوریتم پیشنهادی، تلاش گردیده است که با توسعه سرعت پردازش الگوریتم، محدودیت مذکور نیز رفع گردد. ایده اصلی این توسعه، توزیع پردازش گام‌های پنج‌گانه است. با توزیع هر یک از این گام‌ها، سرعت اجرای الگوریتم بهبود می‌یابد. همچنین چون در ساختار محاسباتی الگوریتم، تغییری اعمال نمی‌گردد، دقت آن کاملاً ثابت خواهد بود. با توجه به این که در هر گام، اطلاعاتی که برای مقایسه‌ها و یافتن همسایگی هر یک از الگوها به کار می‌رود کاملاً مستقل هستند می‌توان پردازش را بین چند پردازنده و یا چند گره پردازشی، تقسیم نمود. در بخش آزمایشات و نتایج، این الگوریتم به دو صورت پیاده‌سازی چندنخی و پیاده‌سازی با چارچوب Map-Reduce مقایسه می‌گردند.

آزمایش‌ها نشان می‌دهد که سرعت الگوریتم بهبودیافته در مقایسه با الگوریتم اصلی به صورت چشم‌گیری افزایش یافته است. البته همان طور که در قبل ذکر گردید، قطعاً موازی‌سازی تمام الگوریتم، باعث بهبود آن الگوریتم نخواهد بود و حتی شاید موجب کاهش سرعت آن به خاطر سربار گردد. لذا باید ابتدا مشکلات آن الگوریتم را به صورت دقیق بررسی نمود و از نقاط ضعف آن مطلع شد. برای الگوریتم ADC نیز چنین فرایندی طی شده است. هر گام آن تحلیل و نقاط ضعف آن شناسایی شده است. این الگوریتم به خاطر محاسبات بسیار زیاد خود می‌تواند به شدت کند باشد اما از جهت دیگر با استفاده از ماتریس در الگوریتم، می‌توان سرعت را بهبود داد. در این مقاله، هدف پیداکردن مشکلات این الگوریتم و سعی در بهبود آن معایب است.

با مطالعه الگوریتم ADC و با بهره‌گیری از ابزار نظارتی[1]، تفاوت توزیع پردازش در گام‌های مختلف الگوریتم و همگن‌نبودن آنها، قابل استنباط است. گام اول و دوم این الگوریتم، شامل محاسبات حجیم است و تمرکز بر روی بهبود این دو گام می‌باشد. ایده اصلی بهبود، توزیع و موازی‌سازی گام‌های مختلف الگوریتم که مستقل هستند می‌باشد.

برای توزیع پردازش این الگوریتم، از عامل‌ها[2] استفاده گردیده است. در این سیستم، هر الگو به صورت عاملی مجزا درمی‌آید و با ایفای نقش به صورت گروهی، مسئله خوشه‌بندی را حل می‌نمایند. همکاری این عوامل در فرایند، صرفاً به صورت ارائه نتایج به عامل مرکزی بوده و در میانه مسیر انجام، به صورت مستقل و بدون تعامل خواهند بود. ابتدا از ساختار چندنخی به جهت موازی‌سازی پردازش استفاده می‌شود. در این ساختار، هر نخ، شبیه‌سازی یک عامل است و با توجه به ذات یک سیستم چندعامله[3]، طبیعی است که با افزایش تعداد عوامل (نخ‌ها) سیستم مقیاس‌پذیر باشد. البته به خاطر محدودیت‌های سیستم عامل و سخت‌افزار در افزایش تعداد الگوها، یک آستانه وجود دارد. این آستانه به صورت (۱۳) تعیین می‌گردد

$$Threads = \#CPU\ sockets \times \#Cores\ per\ socket \times \#Threads\ per\ core \qquad (۱۳)$$
$$Threads = ۲ \times ۴ \times ۲ = ۱۶$$

در حجم محدود از داده‌ها، استفاده از ساختار چندنخی، بسیار مؤثر خواهد بود اما با گذشت حجم از یک آستانه تحمل، لزوم استفاده از روش دیگر، کاملاً ملموس است. استفاده از چند پردازنده به جهت موازی‌سازی [۱۱] به عنوان یک راه حل برای تسریع فرایند اجرای الگوریتم‌ها است اما این راه حل در تمام مسایل، قابل استفاده نمی‌باشد. به علت حجیم‌بودن اطلاعات مورد نیاز در هر گام از الگوریتم ADC، بدون توزیع داده در کنار توزیع پردازش، امکان استفاده از الگوریتم موازی‌شده در کلان‌داده‌ها وجود ندارد. لذا در الگوریتم پیشنهادی، راهکاری به جهت ایجاد امکان توزیع داده نیز در نظر گرفته شده تا بتوان پردازش را نیز موازی نمود.

در گام اول الگوریتم، ماتریس مجاورت مطابق با (۵) به صورت (۶) تشکیل می‌گردد. در ماتریس مجاورت، فاصله هر الگو با الگوی دیگر محاسبه می‌شود. در این گام هر یک از الگوها در نقش یک عامل هستند و با استقلال خود، بدون بروز تداخل، این گام را انجام می‌دهند. اگر تعداد نخ‌ها به عنوان عامل محدودکننده در نظر گرفته نشود، تسریع حداکثر p^2[4] برابر و مرتبه زمانی مقایسات این گام، $O(۱)$ می‌شود.

در ماتریس AM می‌توان به محاسبه ماتریس بالامثلثی اکتفا نمود، پس تعداد فراخوانی تابع dst برابر $p^2/۲$ است. با توجه به این که فرض محدودنبودن تعداد نخ‌ها صحیح نیست، مرتبه اجرا $O((p^2/۲)/t)$ است که در آن t برابر حداکثر تعداد نخ فعال در واحد زمان می‌باشد. پارامتر t حداکثر می‌تواند به اندازه تعداد فراخوانی‌های تابع dst باشد و افزایش آن بیشتر از این مقدار، موجب بیکاری آن نخ‌ها خواهد بود.

1. Java Mission Control
2. Agents
3. Multiagent System

۴. P تعداد الگوها

```
for (i = 0; i < width; i++) {
 for (j = i; j < height; j++) {
   neighbors [i][j] = minValue(adjacencyMatrix(i));
   remove(adjacencyMatrix(i,minValueRowIndex));
 }
}
adjacencyMatrix(i,j){
 return dist(Patterns.get(i),Patterns.get(j));
}
```

شکل ۱۰: گام اول و دوم الگوریتم.

```
for (i = 0; i < width; i++) {
   new Thread(function(){
   for (j = i; j < height; j++) {
    neighbors [i][j] = minValue(adjacencyMatrix(i));
    remove(adjacencyMatrix(i,minValueRowIndex));
   }
  }).start();
}
```

شکل ۱۱: نحوه ایجاد نخ بر اساس الگوهای عددی در الگوریتم ADC.

```
dist(Pattern a, Pattern b) {

  float sum = 0;

  for (i = 0; i < a.getValues().length; i++) {
   if (!a.getValue(i).isNumeric())
    sum += bin_dist(a.getValue(i),b.getValue(i));
   else
    sum += Cont_dist(a.getValue(i),b.getValue(i));
  }
  return sum;
}
```

شکل ۱۲: مقایسه دو الگو در الگوریتم ADC.

علاوه بر بهبود مرتبه زمانی الگوریتم، برای حافظه نیز باید چاره‌ای اندیشید. در ماتریس AM می‌توان مقادیر هر خانه را در لحظه مورد نیاز، محاسبه کرد که این امر موجب کاهش شدید مصرف حافظه خواهد شد. البته باید در نظر داشت که اگر بیش از یک بار نیاز به دسترسی به هر یک از خانه‌های ماتریس AM وجود داشته باشد، بهتر است نتیجه محاسبات در حافظه نگهداری شود. در ادامه اشاره می‌شود که مقادیر این ماتریس در گام بعدی، فقط یک بار با محاسبه تابع dst تعیین می‌گردد، لذا نیازی به ذخیره آن وجود ندارد.

در گام دوم الگوریتم که با آن همسایه‌های مشترک هر الگو پیدا می‌شوند، همچون گام قبل می‌تواند برای هر الگو، به صورت جداگانه و توزیع‌شده پردازش انجام گیرد. با ترکیب دو گام اول الگوریتم، روند اجرا به صورت شکل ۱۰ می‌گردد.

در شکل ۱۱ می‌توان دید که در این شبه‌کد، تداخلی در نوشتن بر روی حافظه توسط نخ‌های مختلف وجود ندارد. لذا فرایند، بدون مشکل اجرا خواهد شد.

حال نحوه محاسبه فاصله بررسی می‌گردد. فاصله بین دو الگو به صورت مقایسه یک‌به‌یک هر یک از خصوصیت‌ها با یکدیگر، انجام می‌گیرد. با توجه به این که نوع اطلاعات ورودی این الگوریتم به صورت مرکب است، برای هر یک از انواع داده‌ای، یک تابع جداگانه برای محاسبه فاصله، مطابق شکل ۱۲ ارائه شده است. فاصله بین دو الگو، حاصل جمع فاصله بین یک‌به‌یک خصوصیت‌های الگوها است که در خروجی قرار می‌گیرد. برای مقایسه دو خصوصیت قیاسی از Simple Matching استفاده شده است.

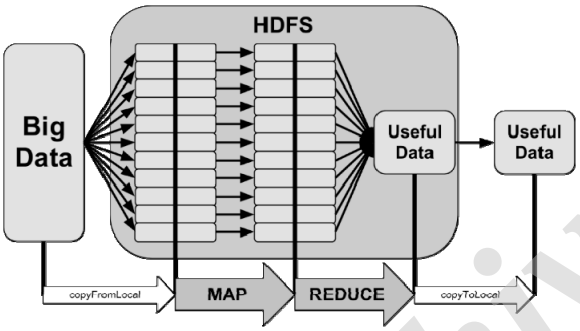
در طی فرایند محاسبه فاصله دو خصوصیت عددی، عمل نرمال‌سازی هم انجام می‌شود. در شکل ۱۳ علاوه بر نرمال‌سازی، نحوه محاسبه فاصله بین دو خصوصیت نیز نمایش داده می‌شود و برای محاسبه این فاصله، از معیار منهتن استفاده می‌گردد.

```
Cont_dist(float a, float b) {
 float minValue = Float.MAX_VALUE;
 float       maxValue       =       Float.MIN_VALUE;
 for (Pattern pattern : Patterns) {
  float value = pattern.getValue();
  maxValue = value > maxValue ? value : maxValue;
  minValue = value < minValue ? value : minValue;
 }
 return Math.abs(a - b) / (maxValue - minValue);
}
```

شکل ۱۳: فاصله دو خصوصیت عددی در الگوریتم ADC.

```
newDist(Pattern a, Pattern b) {

  float sum = 0;

  for (i = 0; i < a.getValues().length; i++) {
   if (!a.getValue(i).isNumeric())

    sum+=pow((GCode(a.getValue(i))-
         GCode(b.getValue(i)),2);

   else

    sum += pow((a.getValue(i)-b.getValue(i)),2);

  }
  return sqrt(sum);
}
```

شکل ۱۴: محاسبه فاصله با اعمال کدگذاری هندسی.



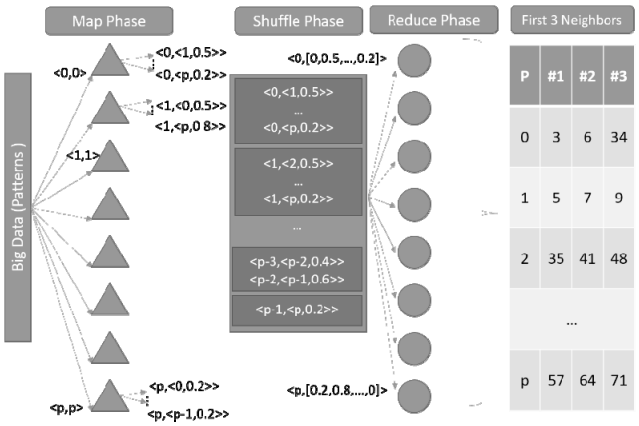
شکل ۱۵: نحوه حضور فایل سیستم هادوپ در الگوریتم پیشنهادی.

با اعمال کدگذاری هندسی بر روی مجموعه داده، نیاز به اصلاح توابع محاسبه فاصله وجود دارد. چون نوع داده‌ای قیاسی کدشده را می‌توان با معیارهای فاصله عددی پردازش نمود، محاسبه فاصله، محدود به فقط یک تابع خواهد بود.

همانطور که در شکل ۱۴ قابل مشاهده است، تابع GCode، خصوصیت قیاسی را به صورت برخط کدگذاری می‌کند.

برخلاف تمام تلاش‌ها برای رفع مشکلات این الگوریتم، باز هم قابلیت مقیاس‌پذیری به آن افزوده نشده است. هر چند که میزان حافظه مصرفی و زمان اجرای الگوریتم، بهبود یافته است ولی هنوز به ساختاری جهت گسترش نامحدود الگوریتم نیاز است.

مشکل اصلی در میزان حافظه مصرفی این الگوریتم و ناتوانی آن در پردازش کلان‌داده‌ها است. برای رفع این معضل از چارچوب Map-Reduce [۳۰] استفاده شده است. همچنین اطلاعات مورد نظر برای پردازش در هر مرحله به فایل سیستم هادوپ منتقل شده تا در اختیار تمام گره‌های پردازشی باشد. در شکل ۱۵ نحوه حضور فایل سیستم هادوپ در الگوریتم پیشنهادی نمایش داده شده است. این فایل سیستم به صورت گسترده در اختیار تمام گره‌های پردازشی قرار می‌گیرد و اطلاعات مشترک مورد نیاز را در اختیار آنها خواهد گذارد. در الگوریتم PADC، فایل سیستم داده‌های الگوها، نتایج محاسبات نگاشت و لیست همسایگان نزدیک هر



شکل ۱۶: الگوریتم PADC در چارچوب MapReduce.

الگو را نگهداری می‌کند. با توجه به سربارهای موجود در این چارچوب، در حجم محدود از داده‌ها تسریعی وجود ندارد. اما با گذشت از آستانه تحمل، با توجه به ناتوانی راهکار چندنخی (در مجموعه داده No_Show، حدود ۱۰۵۰۰) امکان مقایسه وجود ندارد.

شکل ۱۶ الگوریتم PADC در چارچوب MapReduce را معرفی می‌کند. در این چارچوب از یک map و یک reduce استفاده شده و با پایان کار این چارچوب، جمع‌بندی نهایی به صورت متمرکز انجام می‌گیرد. در مرحله map، ورودی‌ها به صورت زوج $\langle i,j \rangle$ که i نمایانگر شماره سطر داده در فایل اطلاعات و j شناسه الگو هستند، خواهند بود. در این مرحله، محاسبات گام اول و دوم انجام می‌گیرد و خروجی هر یک از mapperها به صورت زوج‌های $\langle i,\langle j,dist(i,j)\rangle\rangle$ است که همان سطر و ستون ماتریس حاصل در مرحله دوم الگوریتم (neighbors) خواهد بود. با رسیدن به مرحله shuffle هر یک از خروجی‌های map پس از دسته‌بندی، به شکل لیستی از فواصل بین الگوها خواهند شد. در ادامه reducerها، برای هر الگو با مرتب‌سازی این لیست بر اساس فاصله به صورت صعودی، k همسایه اول هر الگو را تشخیص می‌دهند. شایان ذکر است که این بخش از فرایند نیز می‌تواند به شکل یک چارچوب map-reduce دیگر باشد که صرفاً جنبه بهینگی دارد و در اینجا به آن پرداخته نمی‌شود. با مشخص‌شدن همسایه‌های هر الگو، باید همسایه‌های مشترک الگوهای مختلف پیدا شوند تا خوشه‌ها تشکیل گردند.

۵- آزمایش‌ها و نتایج

پس از معرفی و بررسی دقیق الگوریتم‌ها و روش‌های خوشه‌بندی، در این بخش به مقایسه آنها پرداخته می‌شود. این مقایسه به طور جامع بین الگوریتم‌هایی با کاربرد مشترک و همچنین با شاخص‌های مختلف برای هر نوع الگوریتمی که برای سنجش ویژگی‌های آن معرفی می‌شود انجام می‌گردد که برخی از آن شاخص‌ها باید به صورت ویژه مد نظر باشند. برای مثال شاخص سرعت، یکی از مهم‌ترین مسایل خوشه‌بندی کلان‌داده‌های مرکب است.

۵-۱ روش و ابزار مقایسه و تحلیل

روش مقایسه، پیاده‌سازی و اجرای الگوریتم‌ها روی مجموعه داده‌های مشخص و نهایتاً استخراج شاخص‌ها است. با این شاخص‌ها می‌توان دو الگوریتم را با یکدیگر مقایسه کرده و برای مسأله مورد نظر خود، یکی را به صورت صحیح انتخاب نمود.

مقایسه الگوریتم‌ها با یکدیگر، نیازمند بستری جامع به جهت پیاده‌سازی و اجرای الگوریتم‌ها است. یکسان‌بودن محیط و شرایط اجرا، اعمال ابزار نظارتی به جهت اندازه‌گیری شاخص‌ها و ضرورت وحدت مجموعه داده پردازش‌شده، می‌طلبد که تمام الگوریتم‌ها به شکل یک قالب یکسان پیاده‌سازی گردند. در اینجا چارچوبی[1] که امکان مقایسه و تحلیل الگوریتم‌ها را مهیا می‌کند پیشنهاد می‌شود. نسخه ۰.۷ چارچوب در آدرس http://elki.dbs.ifi.lmu.de/ در دسترس است [۳۱] که در آن می‌توان به تعداد زیادی از پیاده‌سازی‌ها، الگوریتم‌ها، اندازه‌گیری‌ها، تکنیک‌های شاخص‌گذاری، ارزیابی مقادیر و بخش‌های شبیه‌سازی دسترسی داشت. ELKI یک نرم‌افزار متن‌باز داده‌کاوی (AGPLv۳) است که الگوریتم و بخش کاربری آن با تأکید بر روش‌های تحلیل خوشه[2] و کشف نویز[3] به زبان جاوا نوشته شده است.

۵-۲ نتایج و تحلیل

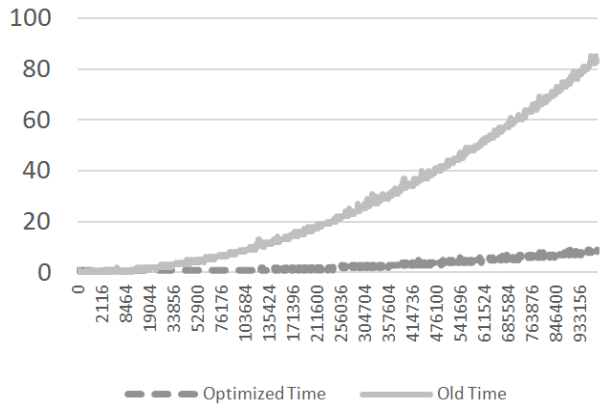
در این بخش ابتدا کدگذاری بهبودیافته و سپس الگوریتم پیشنهادشده تحلیل می‌گردد و برای تحلیل عملکرد الگوریتم پیشنهادی، از مجموعه داده‌های Kaggle[4] استفاده شده است. یکی از مجموعه‌های انتخابی، شامل ۳۰۰ هزار اطلاعات از ویزیت پزشکان[5] که دارای ۱۵ خصوصیت مختلف است می‌باشد. در آزمایشات، این مجموعه داده به ۱۰ هزار الگو تقلیل یافته است. مجموعه انتخابی بعدی [۳۲]، شامل ۲ میلیون الگو است که گزارشگر مصرف برق و نحوه مصرف آن[6] از سال ۲۰۰۶ الی ۲۰۱۰ می‌باشد. شاخص‌های تحلیل، دقت، سرعت و حافظه مصرفی الگوریتم‌ها است که هر یک به صورت جداگانه بررسی شده‌اند.

۵-۲-۱ کدگذاری هندسی

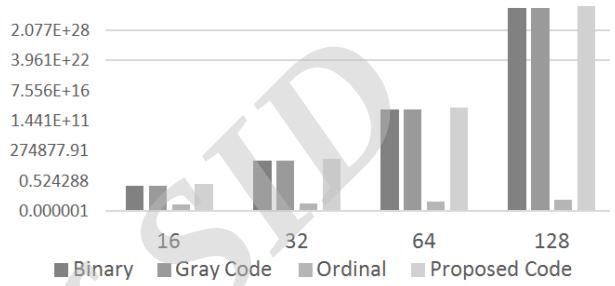
همان طور که در قبل ذکر شد، با حذف محاسبات مثلثاتی در هر بار کدکردن، مرتبه زمانی کاهش می‌یابد. اگر فراخوانی توابع مثلثاتی در مرتبه $O(i)$ [۳۳] باشد، در مختصات دوبعدی، نیاز به $O(2i)$ در هر مرتبه از کدگذاری است. حال اگر به تعداد n دفعه عمل کدکردن انجام گیرد، محاسبات مثلثاتی با مرتبه $O(2in)$ انجام می‌شود. حال آن که در الگوریتم بهبودیافته، به خاطر حذف محاسبات مثلثاتی در هر دسترسی، این مرتبه زمانی برای n دسترسی به $O(n)$ تقلیل می‌یابد. تأثیر این بهبود را می‌توان در نمودار شکل ۱۷ ملاحظه نمود. در این نمودار، محور افقی، تعداد الگوهای موجود در مجموعه داده آزمایش و محور عمودی، زمان کدگذاری در واحد میلی‌ثانیه است.

این میزان تسریع، مستقل از تعداد دسته‌های دامنه است و به هر میزان که این تعداد، افزایش یابد، تغییری در زمان اجرا حاصل نخواهد شد. البته از میزان حافظه مصرفی نیز نباید غافل بود. در دیگر کدگذاری‌ها، مصرف حافظه بسیار کم است، برای مثال در کدگذاری دودویی، صرفاً ارزش مکانی هر یک از خصوصیت‌ها ذخیره می‌گردد. یعنی برای هر مقدار ممکن در دامنه، فقط یک عدد کافی است و لذا مرتبه حافظه مصرفی، (تعداد دسته‌ها) O می‌باشد. در کدگذاری هندسی بهبودیافته پیشنهادی، یک جدول برای نگهداری شاخص‌ها و نگاشت آنها ذخیره می‌گردد و در ادامه یک جدول درهم‌سازی برای دسترسی به نتایج محاسبات مورد نیاز

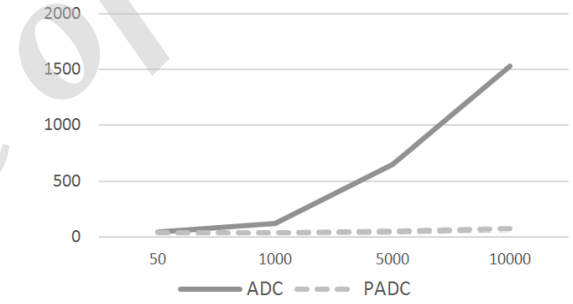
1. Framework
2. Cluster Analysis
3. Outlier Detection
4. https://www.kaggle.com/joniarroba/noshowappointments
5. https://github.com/idotsuk/patients_no_show
6. Power Consumption



شکل ۱۷: مقایسه زمان کدگذاری هندسی بهبودیافته با نسخه اصلی در مجموعه داده تولیدی تصادفی.



شکل ۱۸: میزان حافظه مصرفی کدگذاری‌ها در مجموعه داده تولیدی تصادفی.



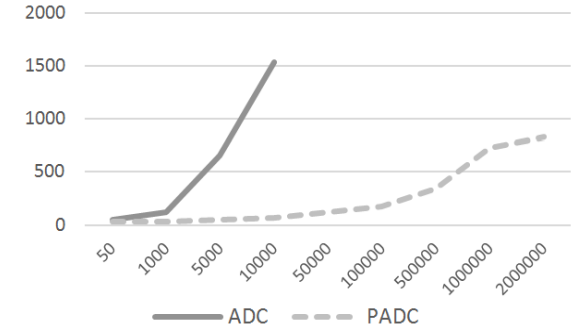
شکل ۱۹: مقایسه زمان اجرای دو الگوریتم PADC (۱۶ Thread) و ADC در مجموعه داده No-show −۱۰k.

است که در مجموع، مرتبه حافظه، $O(۲\times۲^n)=O(۲^n)$ (n تعداد اعضای دامنه است) خواهد بود. برای کدگذاری دودویی نیز مرتبه حافظه $O(۲^n)$ است و لذا تفاوت چشم‌گیری در میزان حافظه مصرفی نخواهد بود. شکل ۱۸ میزان حافظه مصرفی کدگذاری‌ها در مجموعه داده تولیدی تصادفی را نمایش می‌دهد. در این نمودار که محور عمودی آن میزان حافظه مصرفی (کیلوبایت) و محور افقی، تعداد دسته‌ها است می‌توان دید که حافظه مصرفی کدگذاری پیشنهادی با کدگذاری باینری و کد گری، برابر است.

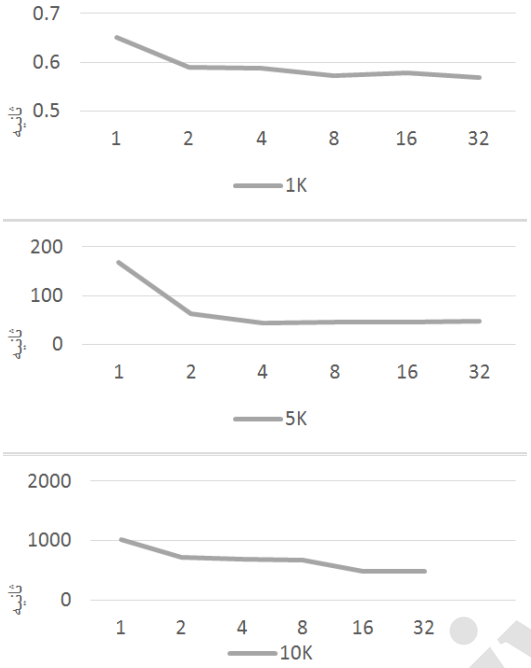
۵-۲-۲ الگوریتم پیشنهادی

همان طور که در بخش قبل ذکر گردید، این پیاده‌سازی به دو صورت انجام گرفته است. در پیاده‌سازی با ساختار چندنخی، به علت کم‌بودن سربار در حجم پایین از داده‌ها، بسیار توانمندتر از چارچوب MapReduce است. شکل ۱۹ و ۲۰ نمایشگر بهبود سرعت الگوریتم به وسیله ساختار چندنخی است.

در (۱۳) اگر ۲ سوکت و برای هر سوکت با توجه به محدودیت‌های پردازشی پردازنده، ۴ هسته موجود باشد، با احتساب دو نخ فعال برای هر هسته (یک نخ فعال و یک نخ رزرو)، حداکثر ۱۶ نخ فعال می‌توان داشت.



شکل ۲۰: مقایسه زمان اجرای دو الگوریتم PADC (۱۶ Thread) و ADC در مجموعه داده Power consumption −۲M.



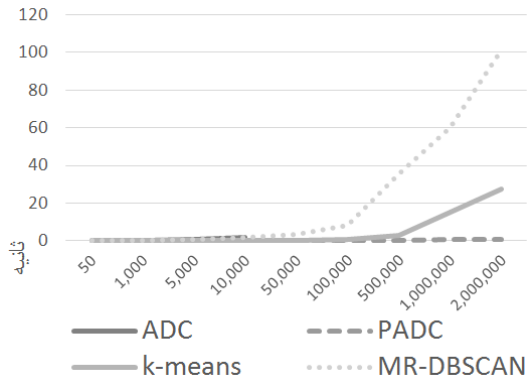
شکل ۲۱: تأثیر تغییر تعداد نخ‌ها بر زمان اجرا در مجموعه داده No-show −۱۰k.

همان طور که در شکل ۲۱ مشاهده می‌شود روند کاهش زمان اجرا با ۱۶ نخ، کاهش نخواهد یافت و در واقع، کارایی پردازنده ۱۰۰٪ خواهد بود. لذا برای رفع این مشکل در زمان اجرای هر گام از خوشه‌بندی که به صورت چندعامله انجام می‌پذیرد، فقط تعدادی محدود از آنها فعال می‌شوند. این امر موجب می‌گردد تا حدی از سرعت این روش کاسته شود.

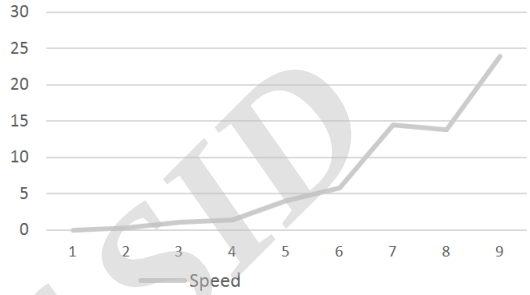
با توجه به محدودیت‌های حافظه، امکان تست الگوریتم اصلی و این الگوریتم بهبودیافته بر روی داده‌های حجیم‌تر وجود ندارد. اما در چارچوب MapReduce این مشکل وجود نداشته و الگوریتم پیشنهادی، مقیاس‌پذیر است.

با مقایسه عملکرد چهار الگوریتم PADC، ADC، K-Means و MR-DBSCAN می‌توان برتری الگوریتم پیشنهادی در سرعت اجرا را استنتاج کرد. شکل ۲۲ نمودار مقایسه این چهار الگوریتم است. الگوریتم پایه ADC بیشتر از ده هزار الگو را نمی‌تواند خوشه‌بندی کند و لذا نمودار آن منقطع گردیده است. در مقایسه با دو الگوریتم K-Means و همچنین MR-DBSCAN زمان طی‌شده برای فرایند خوشه‌بندی الگوریتم پیشنهادی ناچیز است.

در مقایسه نتایج الگوریتم پیشنهادی با الگوریتم k-means [۸]، متوجه می‌شویم که نتایج در ابتدای امر تقریباً مشابه هستند ولی تدریجاً رشد زمان اجرا در الگوریتم k-means شتاب بیشتری را در شبیه‌سازی انجام‌شده نشان می‌دهد که این امر موجب اختلاف ملموس زمان اجرا در ۲ الگوریتم



شکل ۲۲: مقایسه زمان اجرای چهار الگوریتم PADC، ADC، K-Means و MR-DBSCAN.



شکل ۲۳: زمان اجرای الگوریتم PADC در چارچوب MapReduce بر اساس تعداد ورودی‌ها در مجموعه داده Power consumption -۲M با ۴ گره اجرایی.

شده و شکل به وضوح نشان می‌دهد با بزرگ‌ترشدن مقیاس، الگوریتم پیشنهادی به مراتب بهتر عمل می‌کند.

در مقایسه الگوریتم پیشنهادی با الگوریتم MR-DBSCAN [۲۰] مشاهده می‌شود که علی‌رغم مشابه‌بودن الگوی رشد که به علت استفاده هر دو الگوریتم از انجام موازی خوشه‌بندی می‌باشد، زمان اجرا در الگوریتم پیشنهادی برای این مقایسه نیز بهبود قابل ملاحظه‌ای را نشان می‌دهد. علت این امر ذات ماتریسی الگوریتم پایه ADC و ارائه الگوریتم موازی‌سازی بر اساس ویژگی‌های آن می‌باشد.

در شکل ۲۳ سرعت اجرای ((زمان (ثانیه)) /الگو) الگوریتم PADC با ۴ گره پردازشی، نمایش داده شده و بدیهی است که با افزایش گره‌ها، سرعت اجرا افزایش خواهد یافت.

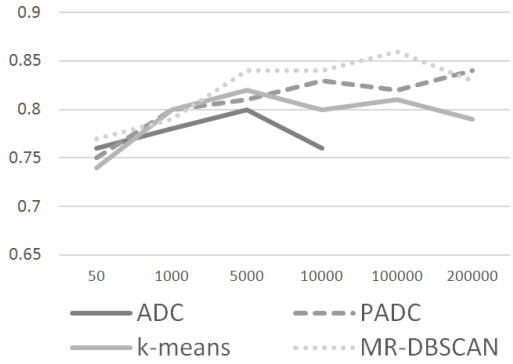
کم‌بودن سرعت اجرا در حجم کم، به دلیل سربار راه‌اندازی سیستم است، لذا استفاده از الگوریتم پیشنهادی در کلان‌داده‌ها و با تعداد گره مناسب، پیشنهاد می‌گردد.

دقت الگوریتم‌ها بر اساس شاخص RAND (رابطه (۱۴)) تبیین گردیده و این شاخص در علم آمار و خوشه‌بندی، شاخص تشابه دو خوشه از اطلاعات است. نحوه محاسبه این شاخص به صورت زیر است

$$R = \frac{a+b}{a+b+c+d} = \frac{a+b}{\binom{n}{2}} \qquad (۱۴)$$

که در آن a تعداد الگوهای موجود در S است که در هر مجموعه، در یک دسته یکسان قرار دارند. b تعداد الگوهایی است که در هر مجموعه، در هیچ دسته یکسانی قرار ندارند. c و d نیز نمایشگر تعداد الگوهایی است که در مجموعه انتخابی در یکی از دو دسته نگنجد.

در طراحی الگوریتم پیشنهادی، علاوه بر تسریع فرایند خوشه‌بندی، افزایش و یا حفظ دقت نیز مد نظر بوده است. شکل ۲۴ نمایشگر مقایسه



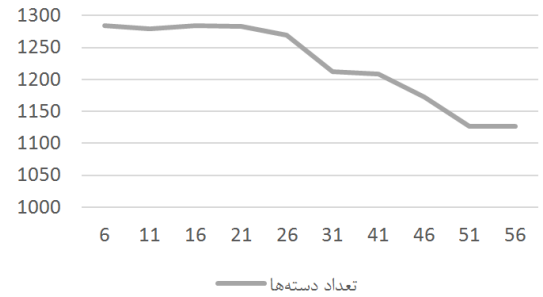
شکل ۲۴: مقایسه دقت چهار الگوریتم PADC، ADC، K-Means و MR-DBSCAN.

شاخص RAND چهار الگوریتم است. همان طور که ملاحظه می‌گردد، الگوریتم پیشنهادی از الگوریتم پایه و K-MEANS دقیق‌تر عمل می‌کند. همچنین دقت دو الگوریتم MR-DBSCAN و PADC نزدیک به یکدیگر هستند و البته در برخی بخش‌ها، MR-DBSCAN دقیق‌تر است که علت آن وجود مقاومت در مقابل نویز در این الگوریتم است. با تنظیم دقیق دو پارامتر ε و minpts می‌توان دقت این الگوریتم را افزایش داد. هر چند که تنظیم دقیق این دو پارامتر، خود یکی از معضلات الگوریتم‌های DBSCAN و MR-DBSCAN است.

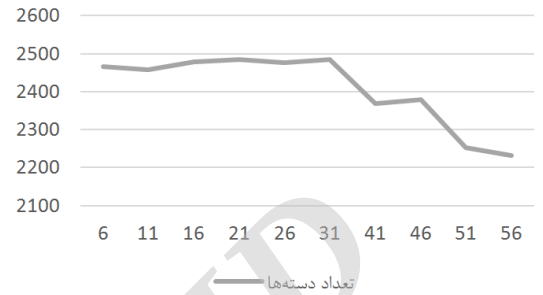
دقت هر دو الگوریتم ADC و PADC تا ۱۰۰۰۰ الگو، بسیار مشابه می‌باشد. علت این امر، عدم تغییر در ساختار اصلی الگوریتم ADC است و صرفاً بر سرعت اجرای آن تمرکز شده است. محور عمودی این نمودار، شاخص RAND و محور افقی، تعداد الگوها است. شایان ذکر است که به خاطر محدودیت الگوریتم اصلی، نتایج مقایسات با بیش از ۱۰۰۰۰ الگو، موجود نیست.

یکی از پارامترهای الگوریتم PADC، تعداد حداکثر همسایه یک الگو است. پس از تعیین فاصله هر الگو با دیگر الگوها، لازم است عملیات مرتب‌سازی انجام شود. الگوریتم به جهت افزایش سرعت، فقط K همسایه اول را مد نظر دارد که این پارامتر می‌تواند بر دقت تأثیر بگذارد و با این پارامتر می‌توان غلظت را تشخیص داد. در [۸] شاخص فاصله با در نظر گرفتن غلظت داده‌ها محاسبه می‌گردد. پارامترهای موجود در الگوریتم ارائه‌شده در [۸] بسیار متعدد هستند که موجب پیچیدگی فرایند خوشه‌بندی می‌گردد. اگر مقدار K به سمت P میل کند، تعداد دسته‌ها کاهش و هرچه به ۱ نزدیک باشد، تعداد دسته‌ها افزایش خواهد یافت. بنابراین تعیین صحیح این پارامتر، بسیار اهمیت دارد. با توجه به هر مجموعه داده، انتخاب این عدد، متفاوت است. می‌توان در شکل ۲۵ و ۲۶ دید که با افزایش این پارامتر، تعداد دسته‌ها کاهش می‌یابد. در این نمودارها محور عمودی، تعداد دسته‌ها و محور افقی، متغیر K می‌باشد.

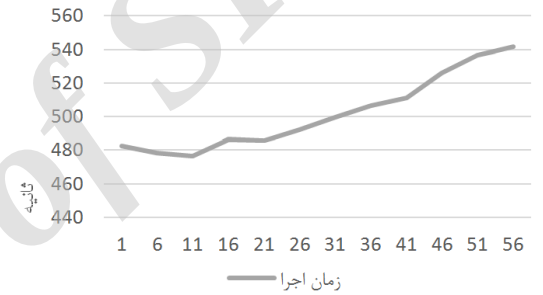
شایان ذکر است که با افزایش مقدار این پارامتر، با توجه به تغییر در روند مرتب‌سازی، سرعت اجرا نیز تغییر خواهد کرد. در شکل ۲۷ و ۲۸ روند افزایش سرعت با افزایش پارامتر K نمایش داده شده است. همان طور که می‌توان ملاحظه نمود با افزایش پارامتر K سرعت انجام خوشه‌بندی کاسته می‌شود، لذا انتخاب درست این پارامتر بسیار مهم است. هر چند که برای هر مورد خوشه‌بندی نیازی به اصلاح این پارامتر نیست اما تعیین دقیق آن علاوه بر دقت، بر سرعت نیز تأثیر زیادی دارد. با توجه به این که هر اجرای فرایند خوشه‌بندی، مستقل از دیگری و به صورت موازی انجام می‌گیرد، می‌توان از kهای مختلف به جهت کشف غلظت مجموعه داده استفاده نمود. تعداد بالای دسته‌های ایجادشده، نشان‌دهنده کم‌بودن K و تعداد کم آن، نشانگر زیادبودن K می‌باشد. در دو مجموعه



شکل ۲۵: تأثیر پارامتر K بر تعداد دسته‌ها در مجموعه داده No-show −۱۰k.



شکل ۲۶: تأثیر K بر تعداد دسته‌ها در مجموعه داده Power consumption −۲M.



شکل ۲۷: تأثیر پارامتر K بر زمان اجرا در مجموعه داده No-show −۱۰k.

داده مورد ارزیابی این مقاله، برای مقدار K، حدود ۵ مناسب است. به طور تجربی استنتاج می‌شود مقدار مناسب برای K برابر با تعداد خصوصیت‌های مجموعه داده است و می‌توان آن را به عنوان شروع در نظر داشت و در صورت لزوم، این مقدار را به صورت دقیق تعیین نمود.

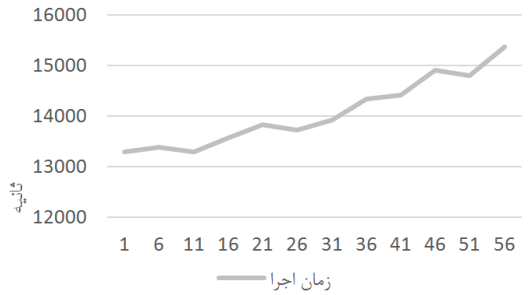
با مقایسه الگوریتم پیشنهادی و الگوریتم پایه، می‌توان دو شاخص تسریع[1] (شکل ۲۹) و مقیاس‌پذیری[2] (شکل ۳۰) را به دست آورد.

۶- نتیجه‌گیری

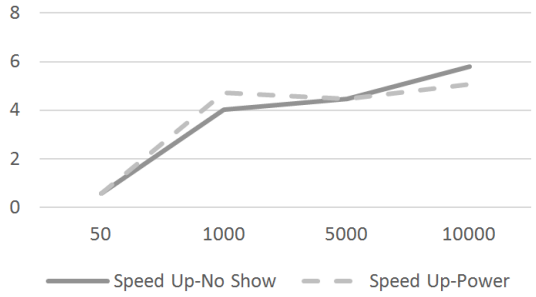
خوشه‌بندی به عنوان یکی از تکنیک‌های مهم داده‌کاوی، دارای ارزش ویژه‌ای است. کاربردهای وسیع آن در پردازش داده‌ها، این تکنیک را تبدیل به یکی از کلیدی‌ترین عناصر این حوزه کرده است [۳۴]. با توجه به رشد سریع داده‌ها و ظهور کلان‌داده‌ها، نیاز به توسعه الگوریتم‌های داده‌کاوی، همگام با این حوزه، کاملاً ملموس است.

در این مقاله، راهکاری برای خوشه‌بندی کلان‌داده‌ها با نوع مرکب پیشنهاد گردیده است. راهکار پیشنهادی، مقیاس‌پذیر و پویا است و می‌تواند نقصان داده‌ها[3] را جبران نماید. این راهکار شامل سه بخش پیش‌پردازش، الگوریتم خوشه‌بندی داده‌های مرکب و اصلاح داده‌ها است.

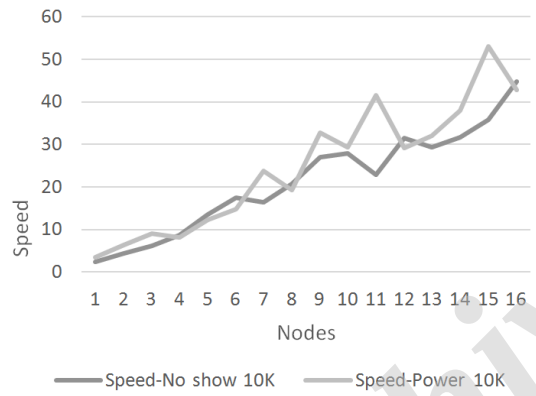
الگوریتم پیشنهادی، نسخه اصلاح‌شده الگوریتم ADC است که برای



شکل ۲۸: تأثیر پارامتر K بر زمان اجرا در مجموعه داده Power consumption −۲M با دوگره پردازشی.



شکل ۲۹: افزایش سرعت الگوریتم پیشنهادی در مقایسه با الگوریتم پایه.



شکل ۳۰: نمودار مقیاس‌پذیری الگوریتم پیشنهادی با افزایش تعداد گره‌های پردازشی.

آن امکان پردازش کلان‌داده‌ها را فراهم می‌کند. بهبود سرعت و حفظ دقت، اهداف توسعه راهکار پیشنهادی بوده است. این توسعه مرتبه الگوریتم را وابسته به تعداد گره‌های پردازشی کرده و با افزایش این گره‌ها، سرعت آن افزایش خواهد یافت.

معیار فاصله استفاده‌شده در این الگوریتم از چگالی داده‌ها و ویژگی‌های مفهومی آنها چشم‌پوشی می‌نماید. لذا پیشنهاد می‌شود به جهت بهبود عملکرد الگوریتم، معیار فاصله آن اصلاح گردد. استفاده از درخت معنا [۱] و در نظر گرفتن تراکم داده‌ها [۲۴]، راهکارهای پیشنهادی برای اصلاح معیار فاصله هستند. همچنین محاسبه فاصله برای خصوصیت‌های عددی به خاطر اجرای فرایند نرمال‌سازی، مرتبه زمانی بالا و نیاز به بهبود دارد.

راهکار پیشنهادی در بخش اصلاح نقصان داده‌ها برای تخمین مقدار، از میانگین وزن‌دار الگوی‌های مجاور استفاده می‌کند. توسعه این بخش به وسیله تکنیک‌های کلاس‌بندی [۳۵] پیشنهاد می‌گردد.

مراجع

[1] C. C. Hsu and Y. P. Huang, "Incremental clustering of mixed data based on distance hierarchy," *Expert Syst. Appl.*, vol. 35, no. 3, pp. 1177-1185, Oct. 2008.

1. Speed Up
2. Scale Up
3. Miss Value

[2] J. Liang, X. Zhao, D. Li, F. Cao, and C. Dang, "Determining the number of clusters using information entropy for mixed data," *Pattern Recognit.*, vol. 45, no. 6, pp. 2251-2265, Jun. 2012.
[3] A. Foss, M. Markatou, B. Ray, and A. Heching, "A semiparametric method for clustering mixed data," *Mach. Learn.*, vol. 105, no. 3, pp. 419-458, Dec. 2016.
[4] F. Cao, *et al*., "An algorithm for clustering categorical data with set-valued features," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 29, no. 10, pp. 4593-4606, Oct. 2017.
[5] F. Noorbehbahani, S. R. Mousavi, and A. Mirzaei, "An incremental mixed data clustering method using a new distance measure," *Soft Comput.*, vol. 19, no. 3, pp. 731-743, Mar. 2015.
[6] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable K-means ++," in *Proc. VLDB Endow.*, vol. 5, pp. 622-633, Mar. 2012.
[7] Y. Kim, K. Shim, M. S. Kim, and J. Sup Lee, "DBCURE-MR: an efficient density-based clustering algorithm for large data using MapReduce," *Inf. Syst.*, vol. 42, pp. 15-35, Jun. 2014.
[8] E. Gungor and A. Ozmen, "Distance and density based clustering algorithm using Gaussian kernel," *Expert Syst. Appl.*, vol. 69, pp. 10-20, 1 Mar. 2017.
[9] Z. He, X. Xu, and S. Deng, "Scalable algorithms for clustering large datasets with mixed type attributes," *Int. J. Intell. Syst.*, vol. 20, no. 10, pp. 1077-1089, Oct. 2005.
[10] X. Cui, P. Zhu, X. Yang, K. Li, and C. Ji, "Optimized big data K-means clustering using MapReduce," *J. Supercomput.*, vol. 70, no. 3, pp. 1249-1259, Dec. 2014.
[11] A. Hadian and S. Shahrivari, "High performance parallel k-means clustering for disk-resident datasets on multi-core CPUs," *J. Supercomput.*, vol. 69, no. 2, pp. 845-863, Aug. 2014.
[12] T. Kanungo, D. M. Mount, N. S. Netanyahu, C. D. Piatko, R. Silverman, and A. Y. Wu, "An efficient k-means clustering algorithm: analysis and implementation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 24, no. 7, pp. 881-892, Jul. 2002.
[13] S. A. Ludwig, "MapReduce-based fuzzy c-means clustering algorithm: implementation and scalability," *Int. J. Mach. Learn. Cybern.*, vol. 6, no. 6, pp. 923-934, Dec. 2015.
[14] S. Fosso Wamba, S. Akter, A. Edwards, G. Chopin, and D. Gnanzou, "How 'big data' can make big impact: findings from a systematic review and a longitudinal case study," *Int. J. Prod. Econ.*, vol. 165, pp. 234-246, Jul. 2015.
[15] J. Han, J. Pei, and M. Kamber, *Data Mining: Concepts and Techniques*, Elsevier Science, 2011.
[16] T. Wangchamhan, S. Chiewchanwattana, and K. Sunat, "Efficient algorithms based on the k-means and chaotic league championship algorithm for numeric, categorical, and mixed-type data clustering," *Expert Syst. Appl.*, vol. 90, pp. 146-167, 30 Dec. 2017.
[17] B. Y. J. Dean and S. Ghemawat, "MapReduce: a flexible data processing tool," *Commun. ACM*, vol. 53, no. 1, pp. 72-77, Jan. 2010.
[18] T. White, Hadoop: The Definitive Guide, vol. 54, 2012.
[19] W. Zhao, H. Ma, and Q. He, "Parallel K-means clustering based on MapReduce," in *Lecture Notes in Computer Science*, vol. LNCS5931, pp. 674-679, 2009.
[20] F. Barcelo-Rico and J. L. Diez, "Geometrical codification for clustering mixed categorical and numerical databases," *J. Intell. Inf. Syst.*, vol. 39, no. 1, pp. 167-185, Dec. 2011.
[21] H. S. M. Coxeter, P. Du Val, H. T. Flather, and J. F. Petrie, *The Fifty-Nine Icosahedra*, New York, NY: Springer New York, 1982.
[22] A. Likas, N. Vlassis, and J. J. Verbeek, "The global k-means clustering algorithm," *Pattern Recognit.*, vol. 36, no. 2, pp. 451-461, Feb. 2003.
[23] Y. He, *et al*., "MR-DBSCAN: an efficient parallel density-based clustering algorithm using MapReduce," in *Proc. of the Int. Conf. on Parallel and Distributed Systems, ICPADS'11*, pp. 473-480, Tainan, Taiwan, 7-9 Dec. 2011.
[24] M. Ester, H. P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," in *Proc. of the 2nd Int. Conf. on Knowledge Discovery and Data Mining, , PAKDD'96*, pp. 226-231, Portland, OR, US, 2-4 Aug. 1996.
[25] Z. Huang, "Clustering large data sets with mixed numeric and categorical values," in *Proc. 1st Pacific-Asia Conf. Knowl. Discov. Data Mining, PAKDD'97*, pp. 21-34, 1997.
[26] M. A. Ben Haj Kacem, C. E. Ben N'Cir, and N. Essoussi, "MapReduce-based k-prototypes clustering method for big data," in *Proc. of the IEEE Int. Conf. on Data Science and Advanced Analytics, DSAA'11*, pp. 473 - 480, Tainan, Taiwan, 7-9 Dec. 2011.
[27] V. V. Ayuyev, A. Thura, N. N. Hlaing, and M. B. Loginova, "The quick dynamic clustering method for mixed-type data," *Autom. Remote Control*, vol. 73, no. 12, pp. 2083-2088, Dec. 2012.
[28] C. M. Ayuyev, V.V., Aung, Z.Y., and Thein, *The Domain Compensation Method for Incomplete Information in a Database*, Tr. Mosk. Gos. Tekhn, pp. 57-64, 2007.
[29] S. Hassani, *Mathematical Physics: A Modern Introduction to Its Foundations*, 2nd Edition, 2013.
[30] L. Wang, *et al*., "Cloud computing: a perspective study," *New Gener. Comput.*, vol. 28, no. 2, pp. 137-146, Apr. 2010.
[31] E. Achtert, H. P. Kriegel, and A. Zimek, "ELKI: a software system for evaluation of subspace clustering algorithms," in *Ludäscher B., Mamoulis N. (eds) Scientific and Statistical Database Management, SSDBM 2008. Lecture Notes in Computer Science,* vol 5069*,* pp. 580-585, 2008.
[32] UCI Machine Learning Repository, *Individual Household Electric Power Consumption Data Set*, [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption. [Accessed: 30-Apr-2018].
[33] R. Andraka, "A survey of CORDIC algorithms for FPGA based computers," in *Proc. ACMSIGDA 6th Int. Symp. F. Program. Gate Arrays, FPGA'98,* pp. 191-200, Monterey, California, US, 22-25 Feb. 1998.
[34] A. Fahad, *et al*., "A survey of clustering algorithms for big data: taxonomy and empirical analysis," *IEEE Trans. Emerg. Top. Comput.*, vol. 2, no. 3, pp. 267-279, Sept. 2014.
[35] Z. Liu, Q. Pan, J. Dezert, and A. Martin, "Adaptive imputation of missing values for incomplete pattern classification," *Pattern Recognit.*, vol. 52, pp. 85-95, Apr. 2016.

محسن محمودی فارغ التحصیل مقطع کارشناسی ارشد در رشته مهندسی کامپیوتر- نرم‌افزار از دانشگاه تربیت دبیر شهید رجایی می‌باشد. وی تحصیلات خود را در مقطع کارشناسی مهندسی کامپیوتر- نرم‌افزار در سال ۱۳۹۵ در دانشگاه زنجان به عنوان استعداد درخشان به اتمام رسانده است. ایشان در طی دوران تحصیل خود در دوره کارشناسی و کارشناسی ارشد سوابقی نظیر دستیار آموزشی در کلاس‌های متعدد و همچنین توصیه‌نامه بسیاری از اساتید مطرح را دارا می‌باشند. همچنین لازم به ذکر است سوابق علمی، تحقیقاتی ایشان در شرکت‌های خصوصی و دولتی ایران، متعدد می باشد. زمینه‌های تحقیقاتی نام‌برده متنوع بوده و شامل موضوعاتی مانند داده‌کاوی، پردازش داده‌ها، سیستم‌های تصمیم‌یار و سیستم‌های همکارانه می‌باشد.

نگین دانشپور استادیار دانشکده مهندسی کامپیوتر دانشگاه تربیت دبیر شهید رجایی می‌باشد. نام‌برده تحصیلات خود را در مقطع کارشناسی مهندسی کامپیوتر- سخت‌افزار در سال ۱۳۷۸ با کسب رتبه اول در دانشگاه شهیدبهشتی، و کارشناسی ارشد مهندسی کامپیوتر- نرم‌افزار در سال ۱۳۸۱ در دانشگاه صنعتی امیرکبیر به پایان رسانده است، و در سال ۱۳۸۹ دکتری خود در رشته مهندسی کامپیوتر- نرم‌افزار را از دانشگاه صنعتی امیرکبیر اخذ کرده است. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: پایگاه داده تحلیلی، سیستم‌های تصمیم‌یار، پیش پردازش داده‌ها، و داده‌کاوی.