دانشگاه صنعتی امیرکبیر

(پلی تکنیک تهران)

دانشکده مهندسی کامپیوتر و فناوری اطلاعات

پایان‌نامه کارشناسی

گرایش نرم‌افزار

عنوان

طراحی و پیاده‌سازی نرم‌افزاری برای تشخیص ناهنجاری در داده‌های جاری

نگارش

ریحانه شاه‌محمدی

استاد راهنما

دکتر امیرحسین پی‌براه

اسفند ۱۳۹۴

صفحه فرم ارزیابی و تصویب پایان نامه- فرم تأیید اعضاء کمیته دفاع

به نام خدا

تعهدنامه اصالت اثر

تاریخ:

اینجانب ریحانه شاه‌محمدی متعهد می شوم که مطالب مندرج در این پایان نامه حا صل کار پژوهشی اینجانب تحت نظارت و راهنمایی ا ساتید دانشگاه صنعتی امیرکبیر بوده و به د ستاوردهای دیگران که در این پژوهش از آنها ا ستفاده شده ا ست مطابق مقررات و روال متعارف ارجاع و در فهر ست 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.

ریحانه شاه‌محمدی

امضا

تشکر و قدردانی

با سپاس فراوان از جناب آقای دکتر پی‌براه که با حسن اعتماد به اینجانب، در همه مراحل انجام این پروژه، با راهنمایی‌ها و توصیه‌های مناسب همواره مرا همراهی نموده‌اند.

چکیده

امروزه در هر لحظه حجم زیادی داده تولید می‌شود. در بسیاری از موارد لازم است که داده‌های تولید شده به سرعت پردازش شوند. داده‌هایی که چنین ویژگی‌هایی دارند در مدل داده جاری دسته‌بندی می‌شوند. به عنوان نمونه‌ای از این موارد می‌توان به نظارت بر داده‌های مربوط به علائم حیاتی بیماران و یا تشخیص تقلب‌های بانکی اشاره کرد. در چنین شرایطی باید عملیات پردازش داده‌ها به صورت بلادرنگ انجام شود و داده‌هایی که نشان‌دهنده‌ی رفتار غیرطبیعی سیستم هستند به سرعت شناسایی شوند. شناسایی داده‌های غیرنرمال، تشخیص ناهنجاری نام دارد که یک فرآیند داده‌کاوی است.

هدف این پروژه تشخیص داده‌های ناهنجار موجود در یک جریان داده است. داده‌ی مورد استفاده در اینجا جریان داده‌ی توییت‌های ارسالی شبکه اجتماعی Twitter است که به صورت به‌هنگام از رابطه کاربری نرم‌افزاری آن دریافت می‌شود. سپس الگوریتم خوشه‌بندی k-means روی آن‌ها اعمال شده و داده‌ها را به چند خوشه تقسیم می‌کند. در نهایت داده‌ای که در فاصله زیادی از مرکز خوشه مربوط به خود قرار دارد، به عنوان داده‌ی ناهنجار شناسایی می‌شود.

واژه‌های کلیدی:

تشخیص ناهنجاری، داده‌کاوی، داده جاری، جریان داده، داده‌های ناهنجار

فهرست عناوین

صفحه

فهرست اشکال

صفحه

۱

فصل اول

مقدمه

مقدمه

فرآیند تشخیص ناهنجاری[1] یک عمل داده‌کاوی است که در آن هدف پیداکردن داده‌هایی است که از الگوی رفتاری نرمال تبعیت نمی‌کنند. ناهنجاری، در اصطلاح داده‌ی پرت[2] نیز نامیده می‌شود. چنین داده‌هایی براثر خطاهای مکانیکی، تغییراتی در رفتار سامانه، خطاهای انسانی و یا رفتار متقلبانه[3] ایجاد می‌شوند. تشخیص آن‌ها می‌تواند خطاهای موجود در سامانه و یا تقلب‌ها را به‌موقع اطلاع دهد و مانع خسارات احتمالی ناشی از آن شود.

امروزه این فرآیند در حوزه‌های وسیعی کاربرد دارد، مانند:

- تشخیص تراکنش‌های تقلبی که با قصد سرقت از بانک و یا حساب مشتری‌ها صورت می‌گیرند.
- شناسایی دسترسی‌های غیرمجازی که به یک شبکه‌ی کامپیوتری می‌شود.
- نظارت بر عملکرد یک شبکه برای تشخیص گلوگاه‌های ایجادشده در آن.
- شناسایی ویژگی‌های جدید در تحلیل تصاویر ماهواره‌ای.
- تشخیص ساختار مولکولی جدید در تحقیقات دارویی.[۱]

در بسیاری از این موارد، عمل تشخیص ناهنجاری باید برای جریانی از داده‌ها[4] انجام شود.

منظور از جریان داده‌ها، مجموعه‌ای از داده‌ها است که به‌طور پیوسته توسط یک منبع داده تولید می‌شوند و ذخیره‌سازی همه‌ی داده‌های آن در یک پایگاه داده عملی نیست. هر داده‌ی عضو این مجموعه، یک داده‌ی جاری[5] نامیده می‌شود.

مشخصه‌های گفته‌شده برای داده‌های جاری، تشخیص ناهنجاری برای آن‌ها را با مشکلاتی روبه‌رو می‌سازد که در زیر چند نمونه از آن‌ها آورده شده است.

---

[1] Anomaly detection

[2] Outlier

[3] Fraudulent

[4] Data stream

[5] Streaming data

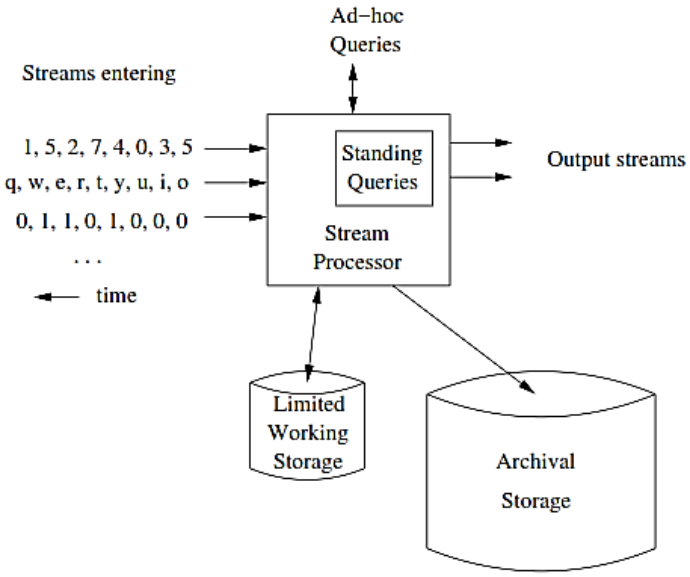
- داده‌های جاری، گذرا[6] هستند؛ یعنی برای پردازش آن‌ها محدودیت زمانی وجود دارد و مدت‌زمان پردازش، باید حداقل باشد. به همین جهت بسیاری از روش‌های پردازش سنتی، روی آن‌ها قابل‌اعمال نیست.
- داده‌های جاری، بی‌شمار هستند. در هیچ لحظه‌ای تعداد داده‌ها مشخص نیست و هیچ‌گاه همه‌ی داده‌ها را باهم در اختیار نداریم. ازاین‌رو، نمی‌توان در پردازش آن‌ها از روش‌هایی که همه‌ی عناصر مجموعه‌ی داده را هنگام پردازش نیاز دارند، استفاده کرد.
- نرخ ورود داده‌های جاری به سامانه پردازش مشخص نیست. ممکن است در یک بازه‌ی زمانی خاص، داده‌ها با نرخ بسیار زیادی به سامانه وارد شوند و پیش از اتمام پردازش یک مجموعه داده، مجموعه‌ی دیگری وارد سامانه شود. در این شرایط ممکن است بخشی از اطلاعات از دست برود. [۳]

بنابراین تشخیص ناهنجاری برای داده‌های جاری به‌سادگیِ داده‌های سنتی نیست.

هدف از این پروژه پیاده‌سازی نرم‌افزاری برای تشخیص ناهنجاری‌های موجود در داده‌های جاری در حجم بزرگ است.

در شکل ۱-۱ ، یک نمای کلی از سامانه‌ای برای پردازش داده‌های جاری در حجم بزرگ، آمده است.

[6] Transient



شکل ۱-۱ نمای کلی یک سامانه پردازش داده‌های جاری[۸]

همان‌طور که در شکل آمده است، جریانی از داده‌ها وارد سامانه می‌شوند و بخشی از آن‌ها که برای پردازش موردنیاز است، در یک پایگاه داده محدود نگهداری می‌شوند. سپس پردازش‌های لازم روی این داده‌ها انجام‌شده و نتایج حاصل از آن در قالب یک جریان داده خروجی تولید می‌شوند.[۸]

در این پروژه چند الگوریتم تشخیص ناهنجاری برای پیام‌های ارسالی در شبکه اجتماعی Twitter، در چارچوب محاسباتی Storm پیاده‌سازی می‌شوند.

در بخش دوم، انواع روش‌های تشخیص ناهنجاری، به‌ویژه روش‌های مورداستفاده در این پروژه، معرفی می‌شوند. پس‌ازآن داده‌های جاری و ابزار موجود برای پردازش آن‌ها معرفی می‌شود. در این بخش، عناصر چارچوب محاسبات استورم و نحوه‌ی کارکرد آن به‌تفصیل آمده است. درنهایت، مروری به کارهای مرتبط با این حوزه انجام‌شده است.

در بخش سوم، یک دید کلی از سامانه طراحی‌شده معرفی می‌شود و به همراه آن اجزای مختلف سامانه توضیح داده می‌شوند. همچنین نحوه‌ی پیاده‌سازی سامانه شرح داده می‌شود.

در بخش چهارم، نتایج آزمایش‌های انجام‌شده بر روی سامانه آورده شده است. در این بخش عملکرد سامانه مورد ارزیابی قرار می‌گیرد.

در بخش آخر، جمع‌بندی، نتایج و کارهای آینده برای این پروژه مطرح می‌شوند.

۲

فصل دوم

مفاهیم پایه و پیشینه

۱.۲ داده‌ی جاری

منظور از داده‌ی جاری، داده‌ای است که یک منبع به‌طور پیوسته آن را تولید می‌کند و باید بی‌درنگ[۷] پردازش شود. در بسیاری از سامانه‌هایی که امروزه در دنیا استفاده می‌شوند ، بخش‌هایی وجود دارد که موظفند به‌طور مداوم داده‌های ورودی را پردازش کند و یا باید بتوانند اتفاقاتی که در لحظات خاصی در سامانه رخ می‌دهد را تشخیص دهند. داده‌های ورودی این دست مسائل، داده‌ی جاری تلقی می‌شوند. به مجموعه‌ای از داده‌های جاری، یک جریان داده گفته می‌شود.[۴]

۱.۱.۲ سامانه‌های مدیریت جریان داده[۸]

در مسائل حوزه‌ی داده‌های جاری، حجم نامحدودی داده وجود دارد و نمی‌توان همه‌ی داده‌ها را در سامانه ذخیره کرد. از طرفی برای آن‌که مفهوم جاری‌بودن رعایت شود، باید پردازش داده‌ها به‌سرعت صورت گیرد. ازاین‌رو احساس نیاز به سبک جدیدی از محاسبات به وجود می‌آید.

سامانه‌های مدیریت پایگاه داده[۹] سنتی: اول این‌که نیاز دارند که داده پیش از پردازش در پایگاه داده ثابت باشد و شاخص گذاری شده باشد، دوم این‌که فارغ از آن‌که داده در چه زمانی وارد سامانه شده باشد، هنگام درخواست کاربر، داده را پردازش می‌کنند. هر دو مورد ذکرشده با نیازمندی‌های برنامه‌های حوزه‌ی پردازش جریان اطلاعات[۱۰] در تناقض هستند. به‌عنوان‌مثال، فرض کنید سامانه‌ای می‌خواهیم که با توجه به وضعیت حس‌گرهای دود و دما از آتش‌گرفتن یک ساختمان مطلع شود. ازیک‌طرف باید در لحظه‌ای که میزان دود موجود در فضا و دمای آن از حد معینی بیشتر شد، سامانه زنگ را به صدا دربیاورد. از طرفی اگر در لحظه‌ای مقدار اندازه‌گیری شده توسط حس‌گرها نشان‌دهنده‌ی آتش‌سوزی نباشد، نیازی به ذخیره‌سازی آن‌ها در سامانه نیست.[۳]

---

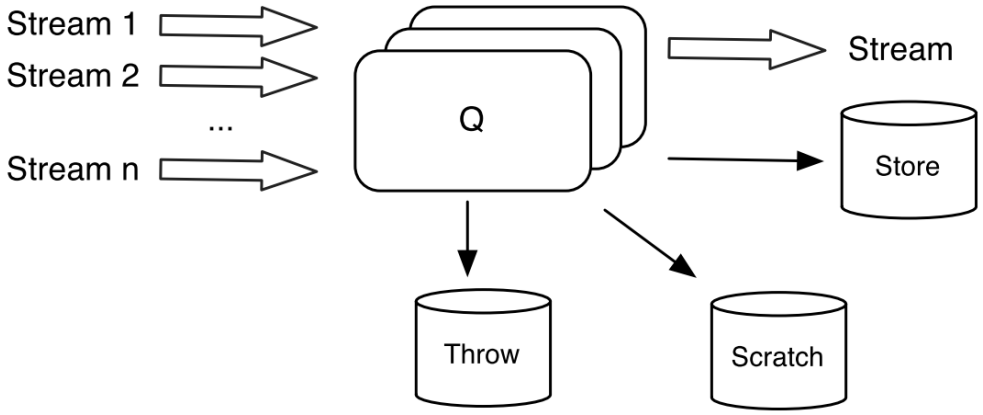
[7] Real-time

[8] Data Stream Management System (DSMS)

[9] DataBase Management System (DBMS)

[10] Information Flow Processing (IFP)

طرح‌های مختلفی برای سامانه‌های مدیریت داده‌های جاری پیشنهاد شده است که در شکل۲-۱، یک مدل رایج از آن آورده شده است.



شکل ۲-۱ معماری یک سامانه مدیریت داده‌های جاری[۳]

در این شکل سامانه به‌عنوان مجموعه‌ای از پرس‌وجو[11]ها، یک یا چند جریان ورودی و چهار نوع خروجی ممکن برای آن، مدل‌سازی شده است. این خروجی‌ها عبارت‌اند از:

جریان: عناصری که به‌عنوان پاسخ تولید می‌شوند و پس از تولید تغییر نمی‌کنند، این جریان را ایجاد می‌کنند.

مخزن[12]: بخشی از پاسخ که ممکن است تغییر کند یا حذف شود جزو این بخش محسوب می‌شود. دو بخش جریان و مخزن، پاسخ پرس‌وجویی که در حال اجرا در سامانه است را تشکیل می‌دهند.

چرک‌نویس[13]: این بخش حافظه‌ی در حال کار سامانه است؛ یعنی ممکن است بخشی از داده را ذخیره کند ولی محتوی آن، بخشی از جواب مسئله نیست.

[11] Query

[12] Store

[13] Scratch

پرتاب[۱۴]: یک نوع سطل بازیافت است که برای دور انداختن چندتایی[۱۵]های غیرضروری استفاده می‌شود.[۳]

درواقع یک سامانه پردازش جریان، واسطی را فراهم می‌کند که بتوان به‌وسیله‌ی آن عملیات موردنظر را روی جریان داده انجام داد. از میان سامانه‌هایی که برای پردازش جریان تولیدشده‌اند، Storm یکی از مطرح‌ترین سامانه‌های موجود است که در این فصل به توضیح اجزا و نحوه‌ی عملکرد آن می‌پردازیم.

۲.۱.۲ سامانه مدیریت جریان داده Storm

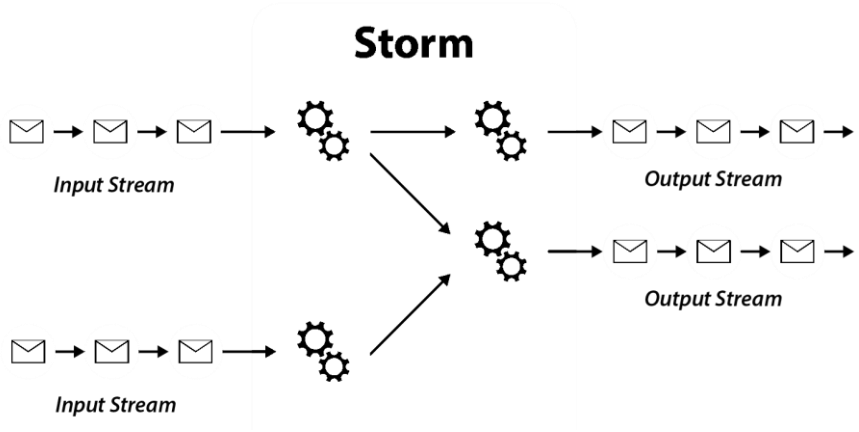
Storm یک چارچوب محاسباتی توزیع‌یافته و تحمل‌پذیر دربرابر خطا برای محاسبات بی‌درنگ در حجم وسیع است که بخش عمده‌ی آن به زبان برنامه‌نویسی کلوژر[۱۶] نوشته‌شده است. این پروژه ابتدا توسط Nathan Marz و گروهش در شرکت BackType ایجاد شد. پس از مدتی Twitter از این پروژه استفاده کرد و کد آن به‌صورت متن‌باز عرضه شد. Storm از ماه سپتامبر سال ۲۰۱۴، به‌عنوان یک پروژه سطح بالای شرکت Apache معرفی‌شده است.[۸]

Storm می‌تواند حجم نامحدودی از داده‌های جاری را بی‌درنگ و به‌صورت قابل‌اطمینان پردازش کند. همچنین قابلیت پشتیبانی از همه‌ی زبان‌های برنامه‌نویسی را دارد و می‌تواند با صف‌بندی‌ها و پایگاه داده‌های مختلف تجمیع شود.

[14] Throw

[15] Tuple

[16] Clojure



شکل ۲-۲ نمای کلی سامانه محاسباتی Storm [۸]

در ادامه عناصر اصلی سامانه محاسباتی Storm، معرفی‌شده‌اند:

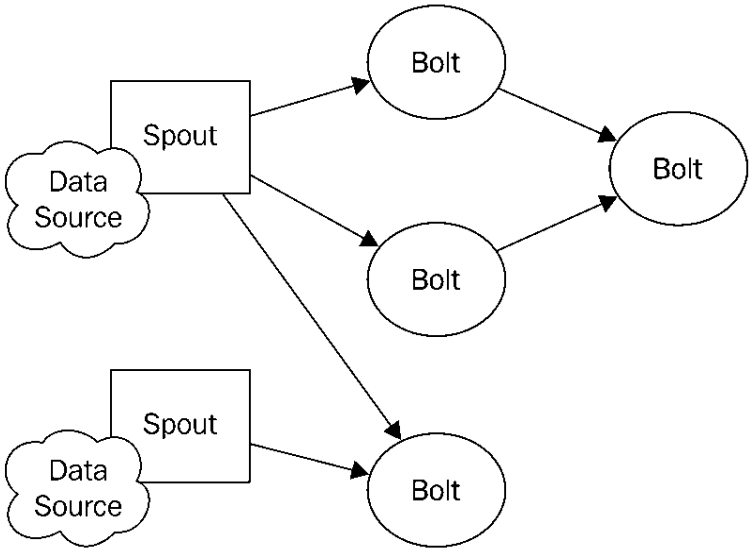
- **جریان**: در این چارچوب محاسباتی، مهم‌ترین مفهوم انتزاعی، جریان داده است. ساختار داده‌ی اصلی در Storm چندتایی است. یک چندتایی، لیستی از مقادیر نام‌گذاری‌شده (زوج‌های کلید-مقدار) است و منظور از جریان داده، دنباله‌ی نامحدودی از چندتایی‌ها است.
- **توپولوژی**: در Storm، ساختار محاسبات توزیع یافته توپولوژی نام دارد. هر توپولوژی شامل یک یا چند جریان داده، تعدادی اسپاوت[۱۷](تولیدکننده‌ی جریان) و تعدادی بلت[۱۸](عملگر) است که با گروه‌بندی جریان‌ها، به شکل یک گراف جهت‌دار بدون دور[۱۹] به هم متصل می‌شوند. توپولوژی‌ها برخلاف کارهای دسته‌ای[۲۰]، نقطه‌ی شروع و پایان ندارند و تا زمانی که کشته نشوند و یا آن‌ها را متوقف نکنیم، به‌طور مستمر اجرا می‌شوند. شکل ۲-۳، مثالی از یک توپولوژی را نشان می‌دهد.[۸]

[17] Spout

[18] Bolt

[19] Directed Acyclic Graph(DAG)

[20] Batch job



شکل ۲-۳ مثالی از یک توپولوژی در Storm [۸]

- **اسپاوت**: محل اصلی ورود داده به یک توپولوژی در Storm، اسپاوت‌ها هستند. درواقع اسپاوت‌ها آداپتورهایی هستند که به یک منبع داده متصل می‌شوند، داده را به شکل چندتایی درآورده و آن‌ها را به‌عنوان جریان در توپولوژی منتشر می‌کنند. ازآنجاکه نقش اسپاوت‌ها تنها تأمین داده به فرم مطلوب برای توپولوژی است و هیچ منطق مربوط به کسب‌وکار خاصی در آن‌ها اعمال نمی‌شود، می‌توان از یک اسپاوت پیاده‌سازی شده در توپولوژی‌های مختلفی استفاده کرد.
- **بلت**: می‌توان هر بلت را به‌عنوان یک عملگر و یا تابع در محاسبات در نظر گرفت. بلت‌ها هر تعداد جریان داده را به‌عنوان ورودی می‌گیرند، پردازشی که برای آن‌ها تعریف‌شده را روی جریان‌ها اجرا می‌کنند و یک یا چند جریان را به‌عنوان خروجی در سامانه منتشر می‌کنند.
- **گروه‌بندی جریان**: بخشی از تعریف یک توپولوژی، مشخص‌کردن جریان‌های ورودی هر بلت است. در Storm هنگام تعریف هر بلت، با گروه‌بندی جریان‌ها مشخص می‌کنیم که کدام جریان‌ها را به‌عنوان ورودی بگیرد. هشت نوع گروه‌بندی مختلف برای جریان‌ها در Storm پیاده‌سازی شده است که موارد پرکاربرد آن در زیر آمده است:

۱. گروه‌بندی مخلوط[21]: چندتایی‌ها به‌طور تصادفی بین وظایف بلت‌ها تقسیم می‌شوند. در این گروه‌بندی تضمین می‌شود که تعداد یکسانی از چندتایی‌ها به هر بلت می‌روند.

[21] Shuffle grouping

۲. گروه‌بندی عرصه‌ها[۲۲]: جریان بر اساس عرصه‌هایی که در گروه‌بندی مشخص می‌شوند، تقسیم می‌شود. به‌عنوان‌مثال، اگر جریان بر اساس عرصه‌ی نام کاربری گروه‌بندی شود، چندتایی‌هایی که در آن‌ها عرصه‌ی نام کاربری یکسان است، به وظیفه یکسانی می‌روند.
۳. گروه‌بندی همه[۲۳]: در این نوع گروه‌بندی، همه‌ی داده‌های موجود در جریان به همه‌ی بلت‌ها می‌روند.
۴. گروه‌بندی سراسری[۲۴]: همه‌ی داده‌های جریان به بلتی می‌روند که کوچک‌ترین شناسه را دارد.
۵. گروه‌بندی هیچ[۲۵]: در حال حاضر، این گروه‌بندی مشابه گروه‌بندی مخلوط عمل می‌کند؛ اما هدف اصلی گروه‌بندی هیچ این است که بلت‌هایی با این نوع گروه‌بندی در همان نخی[۲۶] اجرا شوند که اسپاوت یا بلتی که منبع آن‌ها را تأمین می‌کند، اجرا می‌شود.
۶. گروه‌بندی مستقیم[۲۷]: در این گروه‌بندی همان عنصری که داده را منتشر می‌کند، مشخص می‌کند که هر چندتایی به کدام بلت برود.[۸]

در شکل ۲-۴، نمایی از برخی گروه‌بندی‌های Storm آمده است.

[22] Fields grouping

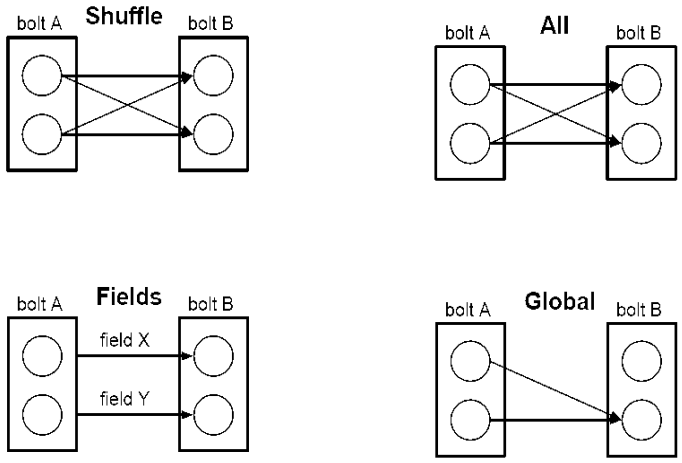
[23] All grouping

[24] Global grouping

[25] None grouping

[26] Thread

[27] Direct grouping

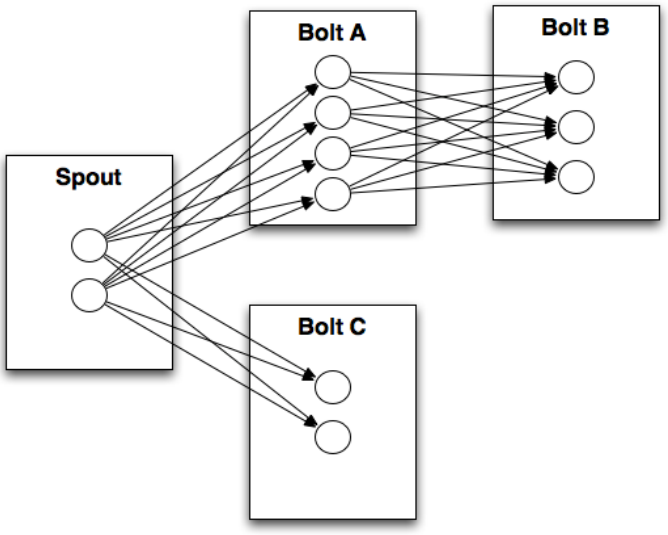


شکل ۲-۴ نمایی از گروه‌بندی‌های Storm [۸]

علاوه بر گروه‌بندی‌هایی که در Storm پیاده‌سازی شده، کاربر می‌تواند بنا به نیاز گروه‌بندی‌های جدیدی برای خود تعریف کند.

- **وظیفه‌ها**: هر اسپاوت یا بلت به‌عنوان چندین وظیفه در یک خوشه[۲۸] اجرا می‌شوند. هر وظیفه معادل اجرای یک نخ است و گروه‌بندی‌هایی که در توپولوژی تعریف می‌شوند، مشخص می‌کنند که چندتایی‌ها چطور از یک مجموعه وظیفه به یک مجموعه وظیفه‌ی دیگر منتقل شوند. در Storm می‌توان هنگام تعریف اسپاوت و بلت، تعداد نخ‌هایی که در نظر داریم برای آن ایجاد شود را مشخص کرد. این یکی از روش‌های تعیین میزان موازی‌سازی در Storm است. شکل ۲-۵، مثالی از وظیفه‌ها و ارتباط آن‌ها در Storm است.

[28] Cluster



شکل ۲-۵ مثالی از وظیفه‌ها و ارتباط آن‌ها در Storm [۸]

- **کارگر**[۲۹]**ها**: منظور از فرآیند کارگر، یک ماشین مجازی جاوا[۳۰]ی فیزیکی است که زیرمجموعه‌ای از وظایف تعریف‌شده در یک توپولوژی را اجرا می‌کند. توپولوژی‌ها را می‌توان روی یک یا چند فرآیند کارگر اجرا کرد. به‌عنوان مثال، اگر ۵۰ فرآیند کارگر داشته باشیم و وظایف یک توپولوژی را به‌گونه‌ای تقسیم کنیم که هر کارگر ۶ وظیفه داشته باشد، میزان موازی‌سازی کلی سامانه ۳۰۰، خواهد بود. Storm وظایف را تا حد ممکن به‌طور مساوی بین کارگرها تقسیم می‌کند.[۸]

۱.۲.۱.۲ معماری Storm

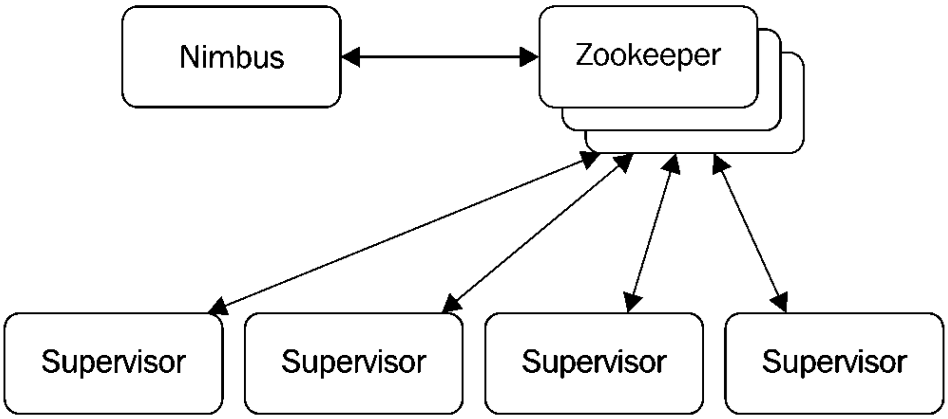
برای درک این‌که Storm چگونه کار می‌کند، لازم است معماری آن را معرفی کنیم. Storm هم مثل سایر روش‌های داده‌ی حجیم به فرم خوشه مستقر می‌شود. منظور از خوشه در این حوزه، مجموعه‌ای از چند کامپیوتر متصل‌به‌هم است که با یکدیگر کار می‌کنند و از دید ناظر خارجی، می‌توان آن را یک سامانه واحد دانست که عملیات مشخصی را انجام می‌دهد. معماری خوشه‌های Storm از نوع راهبر-پیرو[۳۱] است. یک خوشه در Storm، یک گره راهبر (به نام Nimbus) و یک یا چند گره پیرو (به نام Supervisor)

[29] Worker

[30] Java Virtual Machine(JVM)

[31] Master-Slave

دارد. علاوه بر این گره‌ها، Storm به یک نمونه از نوع Zookeeper نیاز دارد و این نمونه می‌تواند از یک یا چند گره از نوع Zookeeper تشکیل‌شده باشد. در شکل ۲-۶، یک نمای کلی از ترکیب این عناصر در Storm آمده است.[۵]



شکل ۲-۶ نمایی کلی از ترکیب عناصر Storm

۲.۲.۱.۲ خوشه‌ی Storm

فرآیندهای Nimbus و Supervisor، شبح[۳۲]هایی هستند که خود Storm آن‌ها را فراهم می‌کند و نیازی نیست که حتماً آن‌ها را در یک ماشین جدا اجرا کرد. درواقع، می‌توان با اجراکردن فرآیندهای Nimbus، Supervisor و Zookeeper روی یک ماشین، یک شبه‌خوشه[۳۳]ی تک‌گره‌ای[۳۴] ایجاد کرد.

برای استقرار یک توپولوژی بر یک خوشه، کد توپولوژی به همراه اطلاعات پیکربندی آن، به Nimbus ارسال می‌شود. مسئولیت اصلی فرآیند Nimbus، مدیریت، هماهنگی و نظارت بر اجرای توپولوژی در خوشه است. به‌علاوه، این فرآیند توپولوژی را مستقر می‌کند، وظایف را به گره‌های کارگر منتسب می‌دهد و در صورت عدم موفقیت در انجام یک وظیفه توسط یک گره، آن وظیفه را مجدداً به یک گره واگذار می‌کند.[۵]

---

[32] Daemon

[33] Pseudo-Cluster

[34] Single node

Supervisorها، گره‌های کارگر هستند و همواره وظیفه‌ای که Nimbus به آن‌ها تخصیص می‌دهد را انجام می‌دهند. یک گره کارگر، فرآیند مربوط به یک ماشین مجازی جاوا است که زیرمجموعه‌ای از یک توپولوژی را اجرا می‌کند.

Zookeeper با استفاده از مجموعه کوچکی از موارد ساده و اولیه، خدمتی برای نگهداری اطلاعات متمرکز در یک محیط توزیع یافته فراهم می‌کند. Storm از Zookeeper برای هماهنگی اطلاعات مربوط به حالت سامانه بین Nimbus و Supervisor استفاده می‌کند. Zookeeper همه‌ی اطلاعات مربوط به وضعیت سامانه (مثلاً نحوه تخصیص منابع به کارگرها و یا سالم یا خراب بودن هر کارگر) را در هرلحظه ذخیره می‌کند و در صورت خرابی Nimbus یا Supervisorها، می‌توان با استفاده از اطلاعات موجود در Zookeeper، آن‌ها را بازیابی کرد و سامانه را مجدداً به راه انداخت.[۵]

در Storm، مشخصه‌هایی پیاده‌سازی شده است که کارایی و قابلیت اطمینان آن را تضمین می‌کند. همچنین، روی تحمل‌پذیری خطا تمرکز شده و تدابیری برای مواجه با خطا در نظر گرفته‌شده است. Storm مکانیسمی دارد که می‌تواند تضمین کند که هر چندتایی به‌طور کامل پردازش شود: به این صورت که اگر مشخص شود که یک چندتایی در یک جزء، پردازش نشده است، به‌طور خودکار دوباره از جزء قبلی فرستاده می‌شود.[۵]

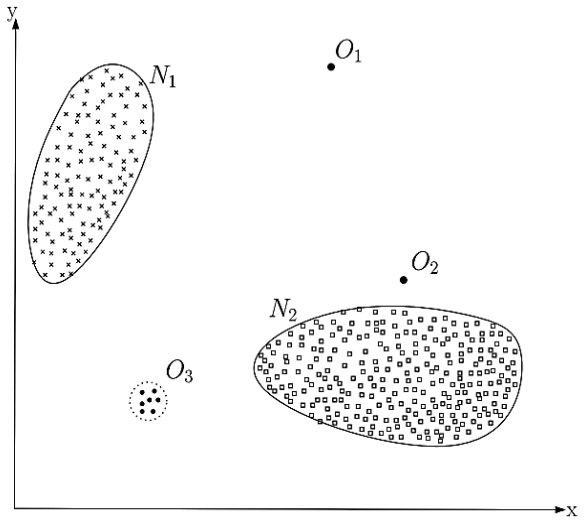
درمجموع، مفاهیم و معماری Storm به‌گونه‌ای تعریف شده است که می‌توان از آن به‌عنوان یک سامانه محاسباتیِ جریان مناسب نام برد. Storm نحوه‌ی تخصیص منابع سطح پایین و نحوه‌ی کنترل جریان را از کاربر مخفی می‌کند و به کاربر این امکان را می‌دهد که بدون درگیرشدن با جزئیات غیرضروری، با استفاده از رابط‌های انتزاعی تعریف‌شده، برنامه موردنظر خود را بسازد.

۳.۱.۲ سامانه‌های مدیریت جریان داده دیگر

علاوه بر Storm، چارچوب‌های دیگری نیز برای پردازش داده‌های جاری مانند Yahoo! S4، Tigon و Pulsar ، تولید شده‌اند؛ اما Storm ویژگی‌هایی دارد که باعث می‌شود گزینه‌ی مناسب‌تری برای پردازش جریان باشد. به‌عنوان‌مثال می‌توان پشتیبانی از همه زبان‌های برنامه‌نویسی، امکان تضمین پردازش حداقل یک‌بار هر قلم داده و وجود جامعه‌ی فعال کاربران آن را نام برد.

۲.۲ ناهنجاری چیست؟

منظور از ناهنجاری یا داده‌ی پرت، داده‌ای است که با الگوی رفتاری داده‌ی نرمال مطابقت ندارد. در شکل ۲-۷، ناهنجاری‌های موجود در یک مجموعه داده‌ی دوبعدی نشان داده‌شده‌اند. مجموعه‌ی داده در دو ناحیه N1 و N2 نرمال است و داده‌هایی که در بخش O1، O2 و O3 هستند، داده‌ی پرت تلقی می‌شوند.[۷]



شکل ۲-۷ مثالی از ناهنجاری‌ها در یک مجموعه داده‌ی دوبعدی[۷]

۱.۲.۲ تشخیص ناهنجاری

تشخیص ناهنجاری، یک فرآیند داده‌کاوی است که در آن داده‌هایی که از الگوی رفتاری داده‌ی نرمال تبعیت نمی‌کنند، شناسایی می‌شوند. امروزه این فرآیند کاربردهای زیادی در حوزه‌های متنوع دارد که می‌توان آن‌ها را تحت قالب زیر دسته‌بندی کرد:

- **تشخیص تقلب**[۳۵]: تشخیص عملکردهای متقلبانه‌ای که با استفاده از کارت‌های اعتباری صورت می‌گیرند.

[35] Fraud

- پردازش بار برنامه کاربردی: تشخیص برنامه‌ها و یا کاربرانی که برای انجام تقلب بار زیادی را به سامانه تحمیل می‌کنند.
- **تشخیص نفوذ**[۳۶]: تشخیص نفوذ غیرمجاز به شبکه‌های کامپیوتری.
- نظارت بر فعالیت‌ها: تشخیص تقلب‌هایی که با استفاده از موبایل صورت می‌گیرند با نظارت بر فعالیت کاربران آن.
- **کارایی شبکه**: نظارت بر کارایی شبکه‌های کامپیوتری، مثلاً تشخیص گلوگاه‌های شبکه.
- **عیب‌شناسی**[۳۷]: نظارت بر فرآیندها برای تشخیص عیب‌هایی که ممکن است در بخش‌های مختلف سامانه وجود داشته باشد. مانند نظارت بر عملکرد موتورها، مولدها و خط لوله‌ها در یک شاتل فضایی.
- **تشخیص نقص‌های ساختاری**: نظارت بر خطوط تولید برای تشخیص اجراهایی که محصول خرابی تولید می‌کنند، مثلاً نظارت بر عمل شکافت پرتوها.
- **تحلیل تصاویر ماهواره‌ای**: تشخیص ویژگی‌های جدید و یا شناسایی ویژگی‌هایی که به اشتباه طبقه‌بندی شده‌اند.
- **تقسیم‌بندی حرکت**: تشخیص بخشی از تصویر که مستقل از زمینه حرکت می‌کند.
- **نظارت بر شرایط پزشکی**: مثلاً نظارت بر ضربان قلب.
- **تحقیقات دارویی**: تشخیص ساختارهای مولکولی جدید.
- **تشخیص اتفاقات جدید در متون**: مثلاً تشخیص یک گزارش خبری که در رابطه با آن متون بسیاری منتشرشده است.
- **تشخیص ورودی‌های غیرمنتظره در پایگاه داده**: مثلاً ورودی‌هایی که به علت خطا یا تقلب واردشده‌اند.

عوامل بروز ناهنجاری در داده‌ها عبارتند از: خطای انسانی، خطای ابزاری، انحراف طبیعی موجود در جمعیت، رفتار متقلبانه، تغییر در رفتار سامانه‌ها و یا خطای سامانه. نحوه‌ی برخورد سامانه تشخیص ناهنجاری با داده‌های پرت به حیطه‌ی کاربرد آن بستگی دارد. مثلاً اگر داده‌ی پرت یک خطای تایپی در یک سند باشد که به علت اشتباه واردکردن حروف رخ داده است، می‌توان خطا را به کاربر نمایش داد تا

[36] Intrusion

[37] Fault diagnosis

آن را تصحیح کند یا با بررسی مشخصات ظاهری یک جمعیت بزرگ، احتمالاً نمونه‌هایی خواهیم داشت که قد آن‌ها اختلاف زیادی با بقیه دارد. این داده به‌عنوان یک مقدار طبیعی شناسایی می‌شود و در دسته‌بندی‌ها باید مدنظر قرار گیرد. در برخی موارد نیز، تشخیص ناهنجاری در سامانه‌های حساس به امنیت[۳۸] استفاده می‌شود و به‌محض وقوع ناهنجاری باید مدیر سامانه را از آن مطلع ساخت. هنگام مواجهه با چنین مواردی می‌توان داده‌ی غیر نرمال را جدا از سایر داده‌ها در سامانه نگهداری کرد و از آن برای ساخت مدلی برای شناسایی سایر ناهنجاری‌ها استفاده کرد.[۱]

۱.۲.۲.۲ چالش‌ها

مسئله تشخیص ناهنجاری همواره با چالش‌هایی روبه‌رو بوده است که می‌توان آن‌ها را در قالب زیر دسته‌بندی کرد:

- تعریف یک ناحیه نرمال به‌طوری‌که همه‌ی انواع داده‌های نرمال را دربرگیرد، بسیار مشکل است.
- در اغلب مواقع، داده‌های نرمال تغییر می‌کنند و یک الگوی ثابت، نمی‌تواند برای تفکیک داده‌های نرمال و غیر نرمال مؤثر باشد.
- در بسیاری از موارد، نمی‌توان مرز دقیق و واضحی بین داده‌ی نرمال و غیر نرمال تعیین کرد. بنابراین، داده‌ای که اختلاف کمی با داده‌های نرمال دارد، به همان اندازه که ممکن است نرمال باشد، ممکن است غیر نرمال باشد.
- مشخص‌کردن ویژگی‌هایی که داده‌ی غیر نرمال در آن ویژگی‌ها متمایز است، اغلب کار ساده‌ای نیست.
- در مواردی که ناهنجاری ناشی از رفتار متقلبانه است، فرد متقلب همواره سعی می‌کند داده‌های غیر نرمال را با الگوی داده‌های نرمال تطبیق دهد و به‌این‌ترتیب عمل تشخیص ناهنجاری بسیار مشکل خواهد شد.
- گاهی داده به‌طور طبیعی شامل عناصری است که تفاوت زیادی با سایر داده‌ها دارد و این داده باید به‌حساب آید. در فرایند تشخیص ناهنجاری، برای این‌گونه داده‌ها باید دقت زیادی در تفکیک این داده‌ها از داده‌های پرت داشته باشیم.[۴]

[38] Safety Critical

با در نظر داشتن موارد ذکرشده در بالا، به دست آوردن یک قاعده برای مسئله شناسایی ناهنجاری‌ها در حالت کلی بسیار مشکل است. درواقع راهکارهایی که برای حل این مسئله استفاده می‌شوند، هرکدام تنها در حوزه خاصی کاربرد دارند و برای هر مسئله، باید راه‌حلی با توجه به ویژگی‌های آن مسئله‌ی خاص ارائه شود.

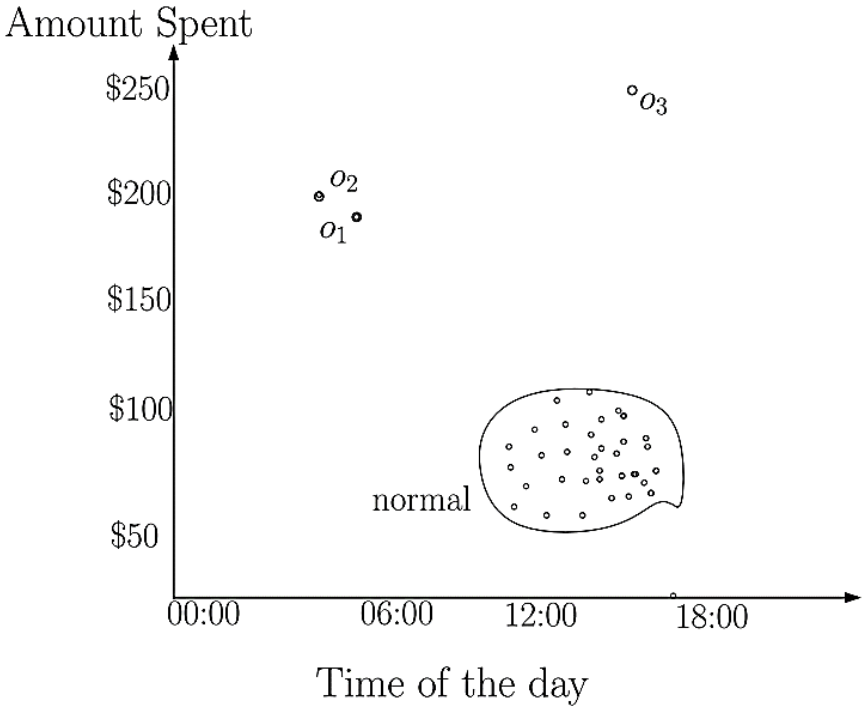
۲.۲.۲ انواع ناهنجاری

بسته به این‌که الگوریتم تشخیص ناهنجاری در چه موردی استفاده می‌شود، ممکن است تعریف داده‌ی پرت در آن، ازنظر ارتباطی که با دیگر عناصر دارد، متفاوت باشد. بنابراین، ناهنجاری‌های موجود در یک مجموعه داده را می‌توان با توجه به نوع ترکیب و ارتباط آن‌ها با سایر داده‌ها در سه دسته‌بندی کلی قرارداد. این دسته‌بندی‌ها عبارت‌اند از:

- **نوع-۱**. در یک مجموعه داده، داده‌ای که تفاوت زیادی با سایر داده‌ها دارد، ناهنجاری است. این نوع ناهنجاری به‌اصطلاح ناهنجاری نقطه‌ای[۳۹] یا عمومی نامیده می‌شود. این ساده‌ترین نوع ناهنجاری است و تمرکز اغلب روش‌های تشخیص ناهنجاری روی این نوع است. روش‌هایی که برای تشخیص داده‌ی پرت نقطه‌ای استفاده می‌شوند، داده‌ای را ناهنجاری به‌حساب می‌آورند که در صفت‌های مورد بررسی، تفاوت چشم‌گیری با بقیه‌ی داده‌های موجود در مجموعه داده داشته باشد.

  به‌عنوان مثال، در تشخیص تقلب در کارت‌های اعتباری، هر شیء داده یک تراکنش مربوط به یک کارت اعتباری است. با فرض این‌که دو صفتِ زمان و مقدار خرج شده در نظر گرفته شوند، شکل ۲-۸، می‌تواند یک طرح نمونه از فضای دوبعدی داده‌ها باشد. سطح منحنی، داده‌های نرمال را نشان می‌دهد. سه تراکنش o1، o2 و o3 خارج از مرز منحنی داده‌های نرمال هستند و به همین علت به‌عنوان داده‌ی پرت نوع-۱ شناسایی می‌شوند. با شناسایی سریع این ناهنجاری می‌توان از زیان‌های احتمالی جلوگیری کرد.

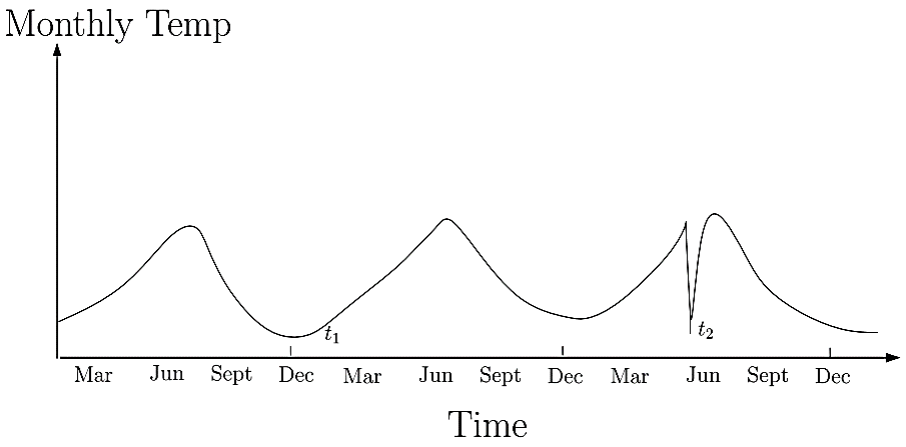
[39] Point Anomaly



شکل ۲-۸ مثالی از ناهنجاری نوع-۱ در یک مجموعه داده‌ی دوبعدی

- **نوع-۲**. گاهی یک رفتار نرمال، در یک زمینه[40]ی خاص ناهنجاری محسوب می‌شود. مثلاً درصورتی‌که میزان دمای تهران در یک روز تابستانی زیر صفر باشد، غیر نرمال است (در این مثال دمای تهران به‌عنوان زمینه‌ی مسئله در نظر گرفته‌شده است). شکل ۲-۹، این ناهنجاری را نشان می‌دهد.

[40] Context



شکل ۲-۹ مثالی از ناهنجاری نوع-۲ برای دمای اندازه‌گیری شده در مواقع مختلف سال

بنابراین یک شیءِ داده اگر با توجه به زمینه خاصی که مدنظر است، با سایر داده‌ها متفاوت باشد، یک ناهنجاری نوع-۲ محسوب می‌شود. به ناهنجاری نوع-۲، ناهنجاری زمینه[۴۱]ای یا شرطی[۴۲] نیز گفته می‌شود. بنابراین، برای این قبیل مسائل، باید زمینه‌ی موردنظر برای تشخیص ناهنجاری هم مشخص شود. در این نوع تشخیص ناهنجاری، داده‌ها با دو گروه صفت مختلف تعریف می‌شوند که عبارت‌اند از:

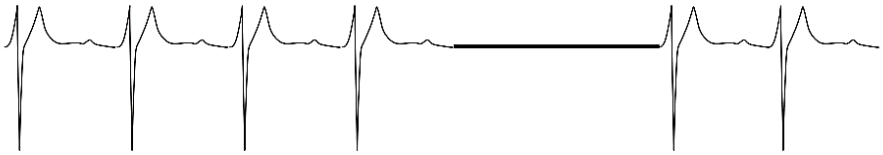
- صفات زمینه‌ای: صفاتی که زمینه‌ی مسئله را تعریف می‌کنند. در مثال بالا، تاریخ و مکان، صفات زمینه‌ای هستند. چون با توجه به این دو صفت، محدوده دمای خاصی را به انتظار داریم.
- صفات رفتاری: صفاتی که مشخصه‌های غیر زمینه‌ای مسئله را تعریف می‌کنند و از این صفات برای تشخیص این‌که یک داده ناهنجاری است یا خیر، استفاده می‌شود. در مثال بالا، میزان دما، رطوبت و فشار هوا می‌توانند صفات رفتاری باشند و اگر از محدوده‌ای که درزمینه‌ی موردنظر مسئله تعریف می‌شود، خارج شوند، به‌عنوان داده‌ی پرت شناخته می‌شوند.

- **نوع-۳**. این ناهنجاری موقعی رخ می‌دهد که یک زیرمجموعه از داده‌ها، نسبت به سایر داده‌ها دورافتاده باشد. ممکن است هرکدام از داده‌های موجود در این زیرمجموعه از داده‌ها، ناهنجاری نباشند، اما وقوع همزمان آن‌ها با یکدیگر، یک ناهنجاری به‌حساب می‌آید. این نوع ناهنجاری،

[41] Contextual Anomaly

[42] Conditional Anomaly

ناهنجاری جمعی[۴۳] نام دارد. در شکل زیر، نمودار ضربان قلب یک فرد آورده شده است. بخش مسطحی که در آن مشاهده می‌شود، علی‌رغم این‌که مقادیر همه‌ی نقاط آن در محدوده‌ی طبیعی هستند، یک ناهنجاری جمعی به‌حساب می‌آید.



شکل ۲-۱۰ مثالی از ناهنجاری نوع-۳ در نمودار ضربان قلب یک انسان

در حوزه‌ی تشخیص تقلب کارت‌های اعتباری نیز، برخی عملکردها در حوزه ناهنجاری جمعی قرار می‌گیرند. فرض کنید شخصی در مدت‌زمان یک ساعت چندین تراکنش از یک پمپ‌بنزین داشته باشد. اگرچه هرکدام از این تراکنش‌ها یک تراکنش نرمال به‌حساب می‌آیند، اما چنین دنباله‌ای از تراکنش‌ها یک ناهنجاری تلقی می‌شود.[۷]

به‌طورکلی ناهنجاری‌های نوع-۱ عمومی‌تر هستند و در هر مجموعه داده‌ای این ناهنجاری‌ها قابل‌تعریف هستند. اما ناهنجاری‌های نوع-۲ و ۳ اگر لازم باشد، برای هر مسئله‌ی خاص، با توجه به ویژگی‌های آن تعریف شوند. یک مجموعه داده می‌تواند چند نوع ناهنجاری داشته باشد و علاوه بر آن، یک داده ممکن است متعلق به بیش از یک نوع ناهنجاری باشد. برای تشخیص ناهنجاری زمینه‌ای لازم است از قبل اطلاعاتی در رابطه با داده‌ها داشته باشیم تا صفات زمینه‌ای آن‌ها را تعریف کنیم. تشخیص ناهنجاری جمعی هم برای مدل‌سازی رابطه‌ی بین اشیاء نیاز به اطلاعات قبلی دارد. البته تشخیص ناهنجاری نوع-۲ و ۳ معنادارتر است. روش‌هایی وجود دارد که می‌توان ناهنجاری نوع-۱ و ناهنجاری نوع-۳ را با توجه به زمینه‌ی مسئله به‌صورت ناهنجاری نوع-۲ تعریف کرد. به‌علاوه، می‌توان با ایجاد تغییراتی درداده، مثلاً متراکم سازی اشیاء داده، به کمک الگوریتم‌های تشخیص ناهنجاری نوع-۱، ناهنجاری نوع-۲ و ۳ را شناسایی کرد.[۷]

[43] Collective Anomaly

۳.۲.۲ روش‌های تشخیص ناهنجاری

روش‌های زیادی برای تشخیص ناهنجاری وجود دارد که می‌توان آن‌ها را طبق دو دیدگاه دسته‌بندی کرد. دیدگاه اول، روش‌ها را بر مبنای در دسترس بودن یا نبودن یک مجموعه از داده‌ها و این‌که این داده‌ها برچسب -چه نرمال و چه غیر نرمال- دارند یا خیر، دسته‌بندی می‌کند. بر این اساس روش‌های موجود به سه دسته‌ی تشخیص ناهنجاری نظارت‌شده، نیمه‌نظارت‌شده و نظارت‌نشده تقسیم می‌شوند. راه دیگر دسته‌بندی این روش‌ها بر مبنای نحوه‌ی جداسازی داده‌های پرت از سایر داده‌هاست. طبق این رویکرد، روش‌های تشخیص ناهنجاری در سه دسته‌ی متدهای آماری[۴۴]، مبتنی بر نزدیکی[۴۵] و مبتنی بر خوشه‌بندی[۴۶] قرار می‌گیرند.[۷]

۱.۲.۲.۲ تشخیص ناهنجاری نظارت‌شده، نیمه‌نظارت‌شده و نظارت‌نشده

تفاوت اصلی این سه روش، در برچسب داشتن یا نداشتن داده‌ی آموزشی است.

- **تشخیص ناهنجاری نظارت‌شده**: این روش هنگامی استفاده می‌شود که هر نمونه‌ی داده‌ی آموزشی، از قبل به‌عنوان نرمال یا غیر نرمال برچسب‌گذاری شده باشد. ابتدا با استفاده از داده‌ی آموزشی، یک مدل برای داده‌های نرمال و غیر نرمال ساخته می‌شود. سپس هر داده‌ی ورودی با این مدل مقایسه می‌شود و در صورت تطابق به‌عنوان داده‌ی هم‌نوع، و در غیر این صورت، به‌عنوان داده‌ای از نوع مخالف تعیین می‌شود.[۷]
- **تشخیص ناهنجاری نیمه نظارت‌شده**: در این روش ابتدا مجموعه‌ای از داده‌ها داریم که تنها بخش کمی از آن‌ها برچسب‌گذاری شده‌اند. در اینجا، با استفاده از داده‌های دارای برچسب و داده‌های بدون برچسبِ نزدیک به آن‌ها، مدلی ساخته می‌شود و مشابه روش قبل، داده‌های ورودی با استفاده از این مدل طبقه‌بندی می‌شوند.[۷]

تشخیص ناهنجاری بدون نظارت: این روش زمانی استفاده می‌شود که هیچ داده‌ی برچسب‌گذاری شده‌ای نداریم. در اینجا طبقه‌بندی داده‌ها با این فرض صورت می‌گیرد که داده‌های نرمال از یک الگوی

---

[44] Statistical

[45] Proximity-based

[46] Clustering-based

تکراری تبعیت می‌کنند. همه‌ی داده‌های نرمال الزاماً مشابه هم نیستند و در یک گروه قرار نمی‌گیرند؛ بلکه ممکن است در چندین گروه قرار داشته باشند که هر گروه الگو و ویژگی‌های متمایزی دارد. به این‌ترتیب، ناهنجاری، داده‌ای است که کمترین شباهت را به گروه‌ها داشته باشد.[۷]

۲.۲.۲.۲ تشخیص ناهنجاری آماری، مبتنی بر نزدیکی و مبتنی بر خوشه‌بندی

- **تشخیص ناهنجاری آماری**: مبنای این‌گونه روش‌ها این فرض است که داده‌ها طبق یک مدل آماری تولید می‌شوند و داده‌ای که طبق آن مدل نباشد، غیر نرمال است.[۷]
- **تشخیص ناهنجاری مبتنی بر نزدیکی**: روش‌های مبتنی بر نزدیکی با این فرض انجام می‌شوند که داده‌ی پرت، فاصله‌ی زیادی از نزدیک‌ترین همسایه‌هایش دارد. دقت تشخیص این روش، به تابعِ فاصله‌ی تعریف‌شده وابسته است.[۷]
- **تشخیص ناهنجاری مبتنی بر خوشه‌بندی**: در این روش فرض می‌شود که داده‌های نرمال می‌توانند در خوشه‌های بزرگ و متراکمی قرار گیرند، درحالی‌که ناهنجاری‌ها در فاصله‌ی بسیار زیادی از مرکز این خوشه‌ها قرار دارند و یا در خوشه‌های کوچک و کم‌تراکمی هستند. [۷]

امروزه روش‌های متنوعی برای تشخیص ناهنجاری‌ها وجود دارد که هرکدام مزایا و معایبی دارند. با توجه به مواردی که در هر مسئله‌ی تشخیص ناهنجاری اهمیت دارد، می‌توان روشی مناسب برای آن مسئله انتخاب کرد.

۲.۲.۴ الگوریتم خوشه‌بندی k-means

الگوریتم k-means یک الگوریتم خوشه‌بندی مبتنی بر مرکز و بدون‌نظارت است. این الگوریتم بسیار ساده است و هزینه کمی برای سیستم دارد.

به‌ازای یک مجموعه داده D = {x1,x2,x3 ...xn} شـامل n عضـو که هر عضـو D یک بردار حقیقی dبعدی اسـت، الگوریتم خوشـه‌بندی k-means داده را به k بخش C1, C2 . . .Ck تقسـیم می‌کند به طوری که Ci ⊂ D و به ازای $1 \leq i,j \leq k$ داریم Ci ∩Cj = Φ. هر خوشه با مرکزش(c_i) نشان داده می‌شود. مراکز خوشه‌ها را می‌توان به روش‌های مختلفی انتخاب کرد، از جمله میانگین اعضای خوشه و یا میانه آن‌ها. در این الگوریتم برای سـنجش فاصـله دو نقطه روش‌های مختلفی بسـته به کاربرد مسـاله استفاده می‌شود که یکی از متداول‌ترین معیارها فاصله اقلیدسی دو نقطه است.[۷]

روش کار الگوریتم به این صورت است که ابتدا k نقطه به صورت تصادفی به عنوان مراکز اولیه خوشه‌ها انتخاب می‌کند. نقطه‌های دیگر هر کدام در خوشه‌ای می‌روند که فاصله کمتری از مرکز آن دارند. در این مرحله مرکز جدید هر خوشـه با توجه به اعضـای آن مشـخص می‌شـود. اکنون مجددا داده‌ها با توجه به مراکز جدید، خوشه‌بندی می‌شوند. این عملیات تا جایی تکرار می‌شود که خوشه‌ها تغییری نکنند.

در الگوریتم۱ این رویه آمده است.

الگوریتم۱ خوشه‌بندی k-means

Input:

k: The number of clusters;

D: A data set containing n objects;

Output:

A set of k clusters;

1: Arbitrarily choose k objects from D as the initial cluster centers;

2: while Cluster means change do

3: (Re) assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;

4: Calculate the mean value of the objects for each cluster to update their means;

5: end while;

۵.۲.۲ تعیین داده‌ی ناهنجار

الگوریتم‌های تشخیص ناهنجاری خروجی‌های مختلفی دارند و با توجه به این‌که خروجی آن‌ها از چه نوعی باشد، ناهنجاری را به شکل متفاوتی مشخص می‌کنند. به همین علت هنگام انتخاب الگوریتم باید به خروجی آن توجه شود. به‌طورمعمول خروجی الگوریتم‌های تشخیص ناهنجاری در دودسته‌ی زیر قرار می‌گیرند:

- **روش‌های برچسب‌گذاری**: در این روش‌ها روی داده‌ها برچسب نرمال و یا غیر نرمال گذاشته می‌شود و درنهایت، دو مجموعه داده‌ی کلی خواهیم داشت که یکی از آن‌ها مجموعه‌ی داده‌های پرت است. مزیت این روش این است که همه‌ی داده‌های پرت را در کنار هم خواهیم داشت و می‌توانیم یک الگوی کلی برای آن‌ها تعریف کنیم. اشکال این روش این است که تفاوتی بین داده‌های پرت متفاوت قائل نمی‌شود و هیچ رتبه‌بندی مشخصی برای آن‌ها ارائه نمی‌دهد. در اغلب موارد، از ناهنجاری‌هایی که اختلاف کمی از داده‌های طبیعی دارند، چشم‌پوشی می‌کنیم. اما با برچسب‌گذاری امکان تصمیم‌گیری در اینژ موارد را نداریم.
- **روش‌های امتیازدهی**: این روش‌ها، به هر داده‌ی ناهنجار با توجه به میزان اختلافش از الگوی داده‌ی طبیعی یک امتیاز می‌دهند. بنابراین، خروجی آن‌ها لیست مرتبی از داده‌های پرت است. در هر سامانه با توجه به نیازی که داریم، ممکن است تعداد زیادی از داده‌های این لیست را به‌عنوان ناهنجاری در نظر بگیریم و یا تعداد کمی از آن‌ها که امتیازشان از حدآستانه مشخصی بیشتر است را انتخاب کنیم. درهرصورت، در تعیین یک داده به‌عنوان ناهنجاری قدرت تصمیم‌گیری بیشتری داریم. اما مشکلی که وجود دارد این است که پیدا کردن یک حدآستانه برای امتیاز داده‌های پرت کار ساده‌ای نیست.

۶.۲.۲ امتیاز ناهنجاری

با اســتفاده از الگوریتم خوشــه‌بندی k-means، داده‌ها به k بخش تقســیم می‌شــوند. داده‌ای که فاصــله زیادی از مرکز خو شه دا شته با شد به عنوان ناهنجاری شنا سایی می شود. با فرض این که O داده‌ای در

خوشـــه‌ای با مرکز C_o باشـــد و L_{C_O} میانگین فاصـــله داده‌های همان خوشـــه باشـــد، با اســـتفاده از رابطه $dist(O, C_o)/ L_{C_O}$ می‌توان مشخص کرد هر داده چقدر از متوسط داده‌های یک خوشه دورتر است. این رابطه امتیاز ناهنجاری یک داده را اندازه می‌گیرد. هرچه امتیاز ناهنجاری یک داده بی شتر با شد، آن داده با احتمال بیشتری ناهنجاری است.

۳.۲ تشخیص ناهنجاری در داده‌های جاری

مسئله‌ی تشخیص ناهنجاری در داده‌های جاری با داده‌های غیر جاری متفاوت است و نمی‌توان همان شیوه‌ی تشخیص ناهنجاری برای داده‌های غیر جاری را برای داده‌های جاری به کاربرد. مشخصه‌های داده‌های جاری، تشخیص ناهنجاری برای آن‌ها را با مشکلاتی روبه‌رو می‌سازد، این مشکلات عبارتند از:

- داده‌های جاری گذرا[47] هستند؛ یعنی برای پردازش آن‌ها محدودیت زمانی وجود دارد و مدت زمان پردازش، باید حداقل باشد. به همین جهت بسیاری از روش‌های پردازش سنتی روی آن‌ها قابل اعمال نیست.
- داده‌های جاری بی‌شمار هستند. در هیچ لحظه‌ای تعداد داده‌ها مشخص نیست و هیچ‌گاه همه‌ی داده‌ها را باهم در دست نداریم؛ بنابراین هر الگوریتم یادگیری آفلاینی که به ذخیره‌سازی کل داده‌ها برای تحلیل نیاز دارد، با مشکل کمبود حافظه مواجه می‌شود.
- در جریان داده‌ها اغلب داده‌ی پرت به‌ندرت اتفاق می‌افتد. به همین علت نمی‌توان از الگوریتم‌های طبقه‌بندی‌کننده‌ای که به داده‌ی آموزشی با همه‌ی برچسب‌های موجود (نرمال و غیر نرمال) نیاز دارند، استفاده کرد.
- داده‌های جاری ممکن است پس از مدت زمانی تغییر یابند و مدلی که برای تشخیص داده‌های غیر نرمال درست‌شده است باید در هر بازه‌ی زمانی با داده‌ها سازگار شود تا همواره دقت تشخیص بالایی داشته باشیم.

[47] Transient

- نرخ ورود داده‌های جاری به سامانه پردازش مشخص نیست. ممکن است در یک بازه‌ی زمانی خاص داده‌ها با نرخ بسیار زیادی به سامانه وارد شوند و پیش از اتمام پردازش یک مجموعه داده، مجموعه‌ی دیگری وارد سامانه شود. در این شرایط ممکن است بخشی از اطلاعات از دست برود.[۴]

بنابراین برای ارائه‌ی یک روش مناسب باید به مسائل گفته‌شده توجه کافی داشت.

۴.۲ کافکا

کافکا یک واسطه‌گر پیام است که به زبان Scala نوشته و به صورت متن‌باز عرضه شده است. کافکا ابتدا توسط Linkedin توسعه یافت و سپس بنیاد نرم‌افزار آپاچی[۴۸] برقرار شد. کافکا یک بستر برای کنترل به‌هنگام داده‌ی لاگ پایدار، توزیع‌یافته، کم‌تاخیر و با نرخ بالا است. در کافکا تولیدکننده‌ها پیام‌ها را در خوشه‌ی کافکا منتشر می‌کنند و مصرف‌کننده‌هایی که عضو یک موضوع هستند، پیام‌های مربوط به آن موضوع را دریافت می‌کنند.[۹]

معماری کافکا با دیگر سامانه‌های پیام‌رسانی بسیار متفاوت است. کافکا یک سامانه‌ی نقطه به نقطه است و هر نقطه در آن یک واسطه[۴۹] نام دارد. واسطه‌ها فعالیتشان را با کمک یک نمونه Zookeeper هماهنگ می‌کنند.[۹]

در ادامه، عناصر مهم خوشه‌ی کافکا معرفی می‌شوند.

۱.۴.۲ تولید کننده

در کافکا بخشی به نام تولیدکننده، پیام‌ها را در قالب عنوان‌های مختلف منتظر می‌کنند. هر عنوان صفی از پیام‌ها است که می‌تواند توسط چندین مصرف‌کننده مصرف شود. کافکا این امکان را می‌دهد که هر

[48] Apache Software Foundation

[49] Broker

عنوان به چند بخش تقسیم شود و خواندن و نوشتن‌ها به‌طور موازی در بخش‌های مختلف انجام شود. به این ترتیب موازی‌سازی انجام می‌شود. داده‌ی هر بخش از یک عنوان در پوشه متفاوتی ذخیره می‌شود و هر کدام از این پوشه‌ها می‌توانند در دیسک‌های متفاوتی باشند. بنابراین محدودیت انجام عملیات ورودی/خروجی روی یک دیسک برطرف می‌شود. همچنین می‌توان دو بخش از یک عنوان را در واسطه‌های متفاوتی قرار داد. هرکدام از پیام‌های یک بخش، شماره‌ترتیب یکتایی به نام آفست دارند.[۹]

۲.۴.۲ تکثیر

در کافکا می‌توان به‌منظور تحمل‌پذیری سامانه در برابر خطا، بخش‌ها را تکثیر کرد. کافکا کنترل تکثیر بخش‌ها را به‌طور خودکار انجام می‌دهد و تضمین می‌کند که هر کپی در واسطه متفاوتی باشد. کافکا به ازای هر بخش یک واسطه به عنوان رهبر انتخاب می‌کند و همه‌ی دستورات خواندن و نوشتن به رهبر می‌روند. این ویژگی از نسخه‌ی 0.8.0 به بعد به کافکا اضافه شده است.[۹]

۳.۴.۲ مصرف‌کننده‌ها

یک مصرف‌کننده محدوده‌ای از پیام‌ها را از یک واسطه می‌خواند. هر مصرف‌کننده یک شناسه گروه دارد. مصرف‌کننده‌هایی که شناسه‌ی گروه مشابهی دارند از لحاظ منطقی یک مصرف‌کننده‌ی واحد را شکل می‌دهند. هرکدام از پیام‌های یک عنوان، به یکی از مصرف‌کننده‌های یک گروه منتقل می‌شود. پیام‌ها پس از مصرف‌شدن توسط یک مصرف‌کننده پاک نمی‌شوند، به همین علت هر مصرف‌کننده می‌تواند با سرعت متفاوتی پیام‌ها را مصرف کند. در واقع، مسئولیت مدیریت تعداد پیام‌های مصرفی، برعهده‌ی خود مصرف‌کننده است.[۹]

مصرف‌کننده‌ها براساس آفست پیام‌های دریافت شده، تشخیص می‌دهند که چه بخشی از جریان را پردازش کرده‌اند. اگر یک مصرف‌کننده بخواهد پیام‌هایی که قبلا پردازش کرده را مجددا دریافت کند، کافی است در حین دریافت پیام‌ها از کافکا شماره‌ی آفست را کاهش دهد.

۳

فصل سوم

طراحی و پیاده‌سازی سیستم

طراحی و پیاده‌سازی سیستم

در این فصل ساختار کلی سیستم و جزئیات مولفه‌های در نظر گرفته شده برای آن و همین‌طور اطلاعاتی در رابطه با نحوه پیاده‌سازی سیستم بیان می‌شود.

۱.۳ معماری سیستم

چارچوب انتخاب شده برای این سیستم Apache Storm است. Storm امکان پردازش داده‌های جاری به صورت توزیع‌یافته را فراهم می‌کند. به‌علاوه انتزاع سطح بالایی از داده مثل اسپاوت و بلت دارد که کار کردن با داده‌های جاری را ساده می‌کند.

این سیستم از دو واحد پردازش مجزا تشکیل شده است. یک واحد برای انجام پیش‌پردازش داده خام ورودی و یک واحد برای خوشه‌بندی داده‌ها و تولید داده‌های ناهنجار به عنوان خروجی در نظر گرفته شده است. هر واحد پردازشی در Storm به صورت یک بلت پیاده‌سازی می‌شود. بلت پیش‌پردازش‌کننده، جریان داده ورودی را دریافت می‌کند و عملیاتی که به عنوان پیش‌پردازش داده تعریف شده است را روی آن اعمال کرده و داده‌ی پیش‌پردازش‌شده را منتشر می‌کند. این داده را بلت مربوط به خوشه‌بندی دریافت می‌کند و داده‌هایی که به عنوان ناهنجاری تشخیص داده را به خروجی می‌برد.

۲.۳ جریان داده ورودی

در Storm داده‌ها از یک منبع خارجی گرفته شده و به وسیله‌ی یک اسپاوت در توپولوژی منتشر می‌شوند. انتشار داده به صورت یک جریان انجام می‌شود. هر جریان دنباله‌ای از چندتایی‌ها با شِمای تعریف شده است. در این پروژه از کافکا به عنوان منبع تامین جریان داده استفاده شده است. یک اسپاوت برای دریافت عناوین کافکا تنظیم می‌شود تا چندتایی‌های متناظر هر عنوان را بگیرد.

در این پروژه داده‌ی خام ورودی، توییت‌هایی هستند که در لحظه در شبکه اجتماعی Twitter منتشر می‌شوند. Twitter یک رابط برنامه‌نویسی نرم‌افزار برای داده‌های جاری دارد که امکان استفاده از جریان توییت‌ها را به توسعه‌دهنده می‌دهد.

۳.۳ واحد پیش‌پردازش

این واحد مخصوص انجام عملیاتی است که باید روی داده‌ی خام انجام شود تا آماده خوشه‌بندی گردد. نمونه‌ای از این عملیات نرمال‌سازی داده‌ها، تجمیع داده‌ها و یا فیلتر کردن داده‌های هدف برای پردازش مراحل بعد است. در این پروژه از هر شیء توییت، صفاتی که مورد نیاز است استخراج می‌شود و هر توییت به عنوان یک چندتایی در شکل‌گیری جریان تولیدی این واحد نقش دارد.

۴.۳ واحد خوشه‌بندی

الگوریتم انتخاب شده برای خوشه‌بندی الگوریتم بدون‌نظارت k-means است. k-means یک الگوریتم ساده است که می‌تواند عملیات خوشه‌بندی را با سرعت و دقت مناسبی انجام دهد. پس از انجام خوشه‌بندی با استفاده از روابط مطرح شده در فصل۲، میزان ناهنجاری هر داده مشخص می شود و در صورتی که داده به عنوان داده‌ی پرت شناسایی شود، به عنوان خروجی این واحد تولید می‌شود.

در این پروژه این بخش با مدل پنجره لغزان عمل می‌کند. در هر لحظه لیستی شامل آخرین ۶۰ داده‌ی وارد شده به این بخش وجود دارد که عمل خوشه‌بندی روی آن داده‌ها انجام می‌شود. با ورود داده‌ی جدید، قدیمی‌ترین داده‌ی موجود در لیست حذف می شود و عمل خوشه‌بندی مجددا انجام می شود. در صورتی که داده‌ی جدید ناهنجاری باشد، به خروجی می‌رود.

الگوریتم استفاده شده در زیر آمده است.

الگوریتم ۲ تشخیص ناهنجاری با استفاده از الگوریتم خوشه‌بندی k-means

Algorithm 2 K-means based anomaly detection algorithm

Input:

k: The number of clusters;

D: A data set containing n objects;

t: the anomaly score to define anomalies

Output:

A set of identified anomalies;

1: Arbitrarily choose k objects from D as the initial cluster centers;

2: while Cluster means change do

3: (Re) assign each object to the cluster to which the object is the most similar, based on the mean value of the objects in the cluster;

4: Calculate the mean value of the objects for each cluster to update their means;

5: end while;

6: Retrieve the average distance of all the objects from their centroid in each cluster;

7: Calculate the anomaly score, that is, every object's distance from centroid divided by the average distance in this cluster;

8: If the anomaly score is higher than t, output this object.

۵.۳ واحد خروجی

در واحد خروجی بسته به این که سیستم در چه جایی و با چه کاربردی استفاده می‌شود، می‌توان عملکردهای متفاوتی تعریف کرد. به عنوان نمونه‌ای از این عملکردها می‌توان از نوشتن خروجی در فایل، انتشار در کافکا یا یک سامانه حافظه نهان توزیع‌یافته مثل Memcached و یا ارسال پیام به مدیریت، می‌توان نام برد. در این سیستم داده‌هایی که به عنوان ناهنجاری شناخته می شوند در یک فایل نو شته می‌شوند.

۴

فصل چهارم

ارزیابی و جمع‌بندی

ارزیابی و جمع‌بندی

در این فصل ابتدا سیستم پیاده‌سازی‌شده ارزیابی می‌شود و سپس نتیجه‌ی کار بیان شده است. در نهایت کارهایی برای انجام روی این پروژه در آینده ارائه شده است.

۱.۴ هدف پروژه

هدف از این پروژه پیاده‌سازی روشی است که ناهنجاری را با دقت مناسبی تشخیص دهد. الگوریتم استفاده شده در آن، یک الگوریتم متداول از نوع نظارت‌نشده به نام k-means است. این الگوریتم در بستر پردازش جریان داده Storm پیاده‌سازی شده است.

۲.۴ روند ارزیابی

برای ارزیابی دقت این سیستم از فاکتورهای رایج ارزیابی الگوریتم‌های تشخیص ناهنجاری استفاده شده است، که در زیر معرفی می‌شوند.

مثبت واقعی (TP): تعداد داده‌هایی که در واقع ناهنجاری بوده‌اند و سیستم هم به عنوان داده‌ی ناهنجار شناخته است.

منفی واقعی (TN): تعداد داده‌هایی که در واقع نرمال بوده‌اند و سیستم هم به عنوان داده‌ی نرمال شناخته است.

مثبت نادرست (FP): تعداد داده‌های نرمالی که سیستم به اشتباه به عنوان ناهنجاری شناخته است.

منفی نادرست (FN): تعداد داده‌های ناهنجاری که سیستم شناسایی نکرده است.

با استفاده از رابطه زیر، می‌توان دقت یک الگوریتم تشخیص ناهنجاری را تعیین کرد.

$$Accuracy = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

۳.۴ نتایج ارزیابی

برای ارزیابی این سیستم از توییت‌های شبکه اجتماعی Twitter استفاده شده است. Twitter یک رابط برنامه‌نویسی نرم‌افزار دارد که می‌توان جریانی از توییت‌های ارسال شده در هر لحظه را به‌صورت بلادرنگ از آن دریافت کرد. به‌طور متوسط در هر ثانیه در Twitter ۶هزار توییت ارسال می‌شود و رابط برنامه‌نویسی نرم‌افزارِ جریانیِ Twitter حدود یک درصد توییت‌ها را نمونه‌گیری می‌کند و در اختیار سیستم قرار می‌دهد.

در این سیستم به ازای ۳۰۰نمونه داده دقت ۷۶درصد به دست آمده است.

با توجه به نتیجه به دست آمده می‌توان الگوریتم k-means را یک الگوریتم مناسب برای تشخیص ناهنجاری دانست. چرا که ضمن سادگی و سرعت عملکرد بالا، دقت مناسبی هم دارد. البته برای کاربردهایی که حساسیت بالایی به ناهنجاری دارند، می‌توان این الگوریتم را در ترکیب با الگوریتم‌های دیگر استفاده کرد تا دقت بیشتری حاصل شود.

۴.۴ جمع‌بندی و کارهای آینده

در فصول پیشین، ابتدا داده‌های جاری و بستر Storm برای پردازش آن‌ها معرفی شد. سپس انواع روش‌های تشخیص ناهنجاری و کاربردهای آن‌ها در پردازش داده‌های جاری بیان شد. به علاوه الگوریتم k-means به عنوان یک الگوریتم مناسب در تشخیص ناهنجاری با جزئیات شرح داده شد. در ادامه اطلاعات طراحی و پیاده سازی سیستمی برای تشخیص ناهنجاری در داده‌های جاری مطرح شده و در نهایت این سیستم ارزیابی شد.

بنابراین، در این پروژه نشان داده شد که با استفاده از الگوریتم k-means در بستر Storm می‌توان داده‌های ناهنجار موجود در یک جریان داده را با دقت مناسبی تشخیص داد. به این ترتیب بسته به این که سیستم در چه جایی استفاده شود، می‌تواند از خسارات احتمالی ناشی از ناهنجاری‌ها جلوگیری کند.

به عنوان کارهای آینده برای این پروژه، می‌توان الگوریتم‌های دیگری در کنار k-means استفاده کرد تا تشخیص ناهنجاری با دقت بیشتری انجام شود. هچنین می‌توان از روش‌هایی برای شناسایی صفات

مناسب‌تر برای خوشه‌بندی، استفاده کرد. به علاوه می‌توان یک رابط برنامه‌نویسی نرم‌افزار برای این سیستم تولید کرد تا در نرم‌افزارهای دیگر قابل استفاده باشد.

منابع و مراجع

[۱] V. Hodge and J. Austin, "A Survey of Outlier Detection Methodologies", *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85-126, 2004.

[۲] V. Chandola, A. Banerjee and V. Kumar, "Anomaly detection: A survey", *ACM Computing Surveys (CSUR)*, vol. 41, no. 3, p. 15, 2009.

[۳] G. Cugola and A. Margara, "Processing flows of information", *CSUR*, vol. 44, no. 3, pp. 1-62, 2012.

[۴] H. C. M. ANDRADE, B. GEDIK and D. S. TURAGA, *Fundamentals of Stream Processing: Application Design, Systems, and Analytics*. New York: Cambridge University Press, 2014.

[۵] A. Jain and A. Nalya, *Learning Storm*. Packt Publishing Ltd., 2014.

[۶] P. Goetz and B. O'Neill, *Storm Blueprints: Patterns for Distributed Real-time Computation*. Packt Publishing Ltd., 2014.

[۷] J. Han, Data Mining: Concepts and Techniques. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005.

[۸] 2016. [Online]. Available: http://kafka.apache.org/.

[۹] Storm.apache.org, "Apache Storm", 2016. [Online]. Available: http://storm.apache.org/.