دانشکده مهندسی کامپیوتر

بررسی الگوریتم‌های یادگیری ماشین برای جریان‌های متنی

گزارش سمینار کارشناسی ارشد در رشته مهندسی کامپیوتر
گرایش هوش مصنوعی و رباتیک

وحید خرازی

استاد راهنما

دکتر بهروز مینایی بیدگلی

آبان ۹۵


بسم الله الرحمن الرحيم

تقدیم به:

پدر و مادر عزیزم که پیغمبر نگاهشان همیشه مرا چشم و چراغ خواهد بود.

قدردانی

سپاس خداوندگار حکیم را که با لطف بی‌کران خود، آدمی را زیور عقل آراست.

در آغاز وظیفه خود می‌دانم از زحمات بی‌دریغ استاد راهنمای خود، جناب آقای دکتر مینایی، صمیمانه تشکر و قدردانی کنم که قطعاً بدون راهنمایی‌های ارزنده ایشان، این مجموعه به انجام نمی‌رسید.

وحید خرازی

آبان ۹۵

چکیده

امروزه ما با منابع تولید داده‌های جریانی روبه‌رو هستیم که داده‌ها را با سرعت بالایی تولید می‌کنند. مدل‌کردن داده‌های حجیم، پیوسته، سریع و متغییر در زمان با توجه به محدودیت‌های زمانی و منابع محاسباتی نیازمند الگوریتم‌های یادگیری است که با یک‌بار مشاهده‌ی داده و با رعایت محدودیت حافظه اصلی، به صورت بلادرنگ پاسخگو باشند. همچنین به دلیل ماهیت متغییر داده‌ها در این فضا، این الگوریتم‌ها باید توانایی تشخیص رانش مفهوم را داشته باشند. به الگوریتم‌هایی که با داده‌های نامحدود، پویا و گذرا کار می‌کنند «الگوریتم‌های یادگیری داده‌های جریانی» گفته می‌شود. حال اگر منبع تولید این داده‌ها، یک تولید کننده محتوای متنی باشد مساله به جهت نفرین ابعاد در داده‌های متنی، وقوع بیشتر رانش مفهوم و دشواری‌های پردازش زبان‌های طبیعی، پیچیده‌تر می‌شود.

در سال‌های اخیر، هم از نظر تئوری و هم از نظر عملی، الگوریتم‌های یادگیری ماشین و داده‌کاوی، تمرکز خود را بر روی مجموعه‌های داده‌ای ایستا، یکجا، همجنس، پایدار، معین، محدود و در نهایت مدل‌های ایستا معطوف کرده‌اند در حالی که امروزه، بسیاری ابزارهای کاربردی، حجم بسیار زیادی از داده‌های جریانی را با سرعت بالایی تولیید می‌کنند. داده‌کاوی و یادگیری از جریان‌های داده‌ای، به ویژه داده‌های متنی که ابعاد بالایی دارند، یکی از موضوعات داغ تحقیقاتی است که می‌تواند کاربردی باشد.

واژگان کلیدی: یادگیری ماشین، داده های جریانی، متن کاوی، پردازش متن

فهرست مطالب

فهرست تصاویر

فهرست جداول

فصل ۱

مقدمه

روش‌های سنتی یادگیری ماشین، معمولا روی مخزن‌های داده ساکن و ایستا تمرکز کرده‌اند، ولی توسعه فناوری باعث شده است که داده‌های جریانی تولید شوند و راهی که مردم داده را ذخیره و پردازش می‌کردند تغییر کند[۱۷] . امروزه بسیاری از سازمان‌ها، حجم زیادی از داده را با سرعت بالا تولید می‌کنند. به عنوان مثال، هر روز گوگل بیشتر از سه و نیم میلیون جستجو را پاسخ می‌دهد[۱] ، ماهواره‌های ناسا بیشتر از ۴ ترابایت عکس تولید می‌کنند[۲] و فروشگاه والمارت بیشتر از ۲۰ میلیون تراکنش را ثبت می‌کند[۳] . مساله‌ی جدید تحقیقاتی داده‌کاوی و یادگیری ماشین این است[۲۰] : «چگونه ما می‌توانیم مجموعه‌ی داده‌ی نامحدودی که سریع تولید می‌شود و در طول زمان تغییر می‌کند را ضمن در نظر گرفتن محدودیت‌های زمانی و منابع محاسباتی مدل کنیم؟».

این مجموعه‌های داده، بسیار بزرگ‌تر از آن هستند که روی حافظه‌ی اصلی جا شوند و باید روی حافظه‌های جانبی ذخیره بشوند. پس دسترسی تصادفی به این داده‌ها، که معمولا در روش‌های سنتی فرض می‌شد که امکان پذیر است، این‌جا بسیار هزینه‌بر است. یک هدف کاوش داده‌های جریانی، ساختن یک فرایند یادگیری است که به صورت خطی با افزایش تعداد نمونه‌ها رشد کند[۱] . به هر حال از آنجایی که داده‌ها به شکل پیوسته در گذر هستند، مدلی که ما از روی داده‌های قدیمی می‌سازیم شاید به درد داده‌های جدید نخورد و باید اثر داده‌های منقضی‌شده از بین برود. این فکر که مدل را با داده‌های جدید دوباره آموزش دهیم، ناکارامد و

[۱] http://www.internetlivestats.com/google-search-statistics/

[۲] http://data.nasa.gov/about/

[۳] http://wikibon.org/blog/big-data-statistics/

جدول ۱-۱: مقایسه بین روش‌های یادگیری سنتی و روش‌های یادگیری در محیط داده‌های جریانی[۲۰]

محدودیت / رویکرد	سنتی	داده های جریانی
تعداد مشاهده داده	چندبار	یک بار
زمان	نامحدود	بلادرنگ
حافظه	نامحدود	محدود
تعداد مفهوم	یک مفهوم	چندین مفهوم
تعداد مفهوم	دقیق	تقریبی

ناکافی است. پس یک هدف کاوش داده‌های جریانی می‌تواند این باشد که چطور مدل را به صورت افزایشی و با دیدن نمونه‌های جدید بروز رسانی کنیم[۲۰] .

داده‌های جریانی با شاخصه‌هایی نظیر حجم نامتناهی، با اهمیت بودن ترتیب یا زمان وقوع و تغییرات پویا شناخته می‌شوند. مثلا گوگل، میلیون‌های جستجو را روزانه پردازش می‌کند که هر کدامشان در یک زمان مشخص انجام شده‌اند و این جستجوها در طول زمان و براساس موضوعات داغ روز تغییر می‌کنند. جدول ۱-۱ تفاوت شاخصه‌ها بین الگوریتم‌های جریانی و سنتی را نشان می‌دهد. به عبارت دیگر، اگر بخواهیم محدودیت‌های الگوریتم‌ها را در فضای داده‌های جریانی برشمریم باید از چهار محدودیت عمده نام ببریم[۱]:

- یک‌بار مشاهده داده:[۴] بر خلاف روش‌های یادگیری سنتی که می‌توانستند مجموعه‌ی داده را مکررا و چند بار پیمایش کنند، الگوریتم‌ها در داده‌های جریانی فقط یک‌بار می‌توانند داده را مشاهده کنند و برگشت به عقبی وجود ندارد. دلیل این موضوع هم آن است که خواندن/نوشتن روی حافظه‌های جانبی بسیار هزینه‌برتر از حافظه اصلی است. این محدودیت زیادی است و اگر به الگوریتم‌ها اجازه دهیم به جای یک داده، چندین داده را ببینند و اجازه داشته باشند که چند نمونه را، البته فقط در کوتاه مدت ذخیره کنند، اوضاع کمی بهتر می‌شود. به عنوان مثال یک الگوریتم می‌تواند یک دسته از نمونه‌ها را برای پردازش‌های داخلی ذخیره کند اما در نهایت باید با گرفتن یک دسته جدید از داده، داده‌های گذشته را پاک کند.

- پاسخ بلادرنگ:[۵] بسیاری از کاربردهای داده‌های جریانی نظیر پیش‌بینی بازار بورس، به پاسخ بلادرنگ نیاز دارند. میزان زمان مورد نیاز برای پردازش داده‌ها و رسیدن به تصمیم باید کم باشد.

[۴] Single-pass

[۵] Real-time Response

- محدودیت حافظه: حجم داده‌ای که با آن مواجه هستیم خیلی بزرگ است و حتی ممکن است نامحدود باشد و ما می‌توانیم تنها تعداد کمی یا خلاصه‌ای از داده‌های جریانی را ذخیره کنیم و ممکن است دیگر به اصل داده دسترسی نداشته باشیم. با این وضع، روش‌های تخمینی هم می‌توانند قابل قبول باشند.
- تشخیص رانش مفهوم: رانش مفهوم زمانی است که الگوها(یا توزیع داده) در طی زمان تغییر پیدا می‌کند.

حال اگر منبع تولید جریان داده، منبعی باشد که متن تولید می‌کند مساله از جهاتی بغرنج‌تر می‌شود. پردازش و مدل‌کردن داده‌های متنی به دلیل غیر ساخت‌یافته‌بودن [۶] ، خطاهای سطح بالا [۷] و وجود در فرمت‌های گوناگون بسیار پیچیده‌تر از سایر داده‌هاست[۲۰] .نفرین ابعاد [۸] یکی دیگر از معضلات کار با داده‌های متنی است[۴]. در برخی از کاربردها، تعداد ویژگی‌هایی که از یک متن استخراج می‌شود چیزی بیش‌تر از دویست‌هزار ویژگی است. بنابراین لازم است که الگوریتم‌هایی در برابر داده‌های جریانی متنی به کار گرفته شوند که قابلیت سازگاری با این تعداد بسیار زیاد ویژگی را داشته باشند و یا این که با روش‌هایی نظیر پیش‌پردازش، انتخاب ویژگی و یا کاهش ابعاد، این داده‌ها را طوری به ورودی الگوریتم‌ها داد که محدودیت‌های زمانی و حافظه‌ای رعایت شود. مضاف بر همه‌ی این مشکلات، در داده‌های متنی تعداد بسیار بیشتری از رویدادهای تعجب‌آور و تغییر موضوعات شگفت‌انگیز مشاهده می‌شود. این موضوع باعث می‌شود که کاوش و یادگیری در داده‌های متنی یک وظیفه دلهره‌آور باشد[۲۰] .

در این فصل به معرفی فضای کار در الگوریتم‌های یادگیری برای داده‌های جریانی پرداختیم، به علاوه مشکلات پیش‌رو زمانی که منبع داده‌ی ما یک منبع تولید کننده متن باشد را معرفی کردیم. در فصل ۲ ابتدا به مرور ادبیات یادگیری در داده‌های جریانی خواهیم پرداخت و مفاهیمی نظیر منبع داده، مفهوم و رانش‌مفهوم را به شکل ریاضی تعریف می‌کنیم و سپس رویکردهای مختلف در برابر داده‌های جریانی را معرفی می‌کنیم. در این فصل علاوه بر ارایه تعاریف اولیه، مروری بر نرم‌افزارهای داده‌کاوی و یادگیری در فضای جریانی خواهیم داشت و مخزن‌های داده‌ی دردسترس را جهت تحقیقات و آزمایش‌های آتی معرفی خواهیم کرد. در فصل ۳ به معرفی الگوریتم‌های یادگیری ماشین در فضای داده های جریانی خواهیم پرداخت. این الگوریتم‌ها عموما نسخه‌های توسعه‌یافته از الگوریتم‌های سنتی یادگیری ماشین هستند و با چالش‌ها و محدودیت‌های داده‌های جریانی

[۶] Unstructured
[۷] High Level of Noise
[۸] Curse of Domensionality

سازگار شده‌اند. به طور کلی در این فصل الگوریتم‌هایی را معرفی خواهیم کرد که تغییر یافته الگوریتم‌های سنتی درخت تصمیم، یادگیرنده‌های بیز، شبکه‌های عصبی، ماشین‌های بردار پشتیبان و مجمع‌های رده‌بندی هستند. نهایتا در فصل ۴ و پایانی این گزارش، ابتدا به جمع‌بندی روش‌های یادگیری ارایه‌شده، مقایسه و نمایش نسبت الگوریتم‌ها با پدران سنتیشان می‌پردازیم، سپس مزایا و معایب هر کدام را مطرح می‌کنیم و مشخص خواهیم کرد که کدام روش‌ها می‌توانند برای منابع جریانی داده‌های متنی به کار گرفته شوند. در نهایت نیز به جمع‌بندی کار و معرفی تحقیقات آینده، نظیر انتخاب پویای ویژگی برای داده‌های جریانی خواهیم پرداخت.

فصل ۲

مروری بر ادبیات

در فصل پیش، به معرفی فضای یادگیری در جریان‌های داده‌ای پرداختیم و ویژگی‌های این الگوریتم‌ها و محیط اجرای آن‌ها را بررسی کردیم. همچنین مقایسه‌ای از محدودیت‌های این الگوریتم‌ها با الگوریتم‌های سنتی ارایه دادیم. در این فصل به مقدمات اصلی کار می‌پردازیم و تعاریفی نظیر مفهوم، منبع داده، رانش مفهوم و پنجره‌ی زمانی را مطرح می‌کنیم. به علاوه رویکردهای کلی به الگوریتم‌های یادگیری محیط داده‌های جریانی را نیز بررسی خواهیم کرد و در نهایت نرم‌افزارها و داده‌های مورد استفاده در فضای تحقیقاتی را معرفی خواهیم کرد.

۲-۱ تعاریف اولیه

در این بخش به معرفی چند تعریف اولیه نظیر منبع داده، مفهوم، رانش مفهوم می‌پردازیم.

۲-۱-۱ جریان داده

یک جریان داده [۱] به شکل یک دنباله از نمونه‌های داده تعریف می‌شود[۱۴] و آن را با نماد DS نشان می‌دهیم: $DS = \{x_1, x_2, ..., x_t, ...\}$ که در این رابطه x_i برابر i امین داده‌ی مشاهده شده است. هر نمونه داده‌ی x_i یک برچسب دارد و با $y_i \in Y = \{y_1, y_2, ..., y_c\}$ نشان داده می‌شود.

[۱] Data Stream

۲-۱-۲ مفهوم

یک مفهوم[۲] یا یک منبع داده، توسط احتمال پیشین برای رده‌ها و تابع توزیع احتمال رده‌ی آن ها تعریف می‌شود[۱۴]:

$$S = \{(P(y_1), P(X|y_1)); ...; (P(y_c), P(X|y_c))\}$$

که در این رابطه y بیانگر رده‌ها است. روش‌های سنتی یادگیری ماشین و داده‌کاوی با یک مفهوم کار می‌کند، و این بدان معنی است که تابع توزیع احتمال مجموعه آموزش و مجموعه آزمون یکسان است. این در حالی است که داده‌های جریانی پویا هستند و تعداد زیادی مفهوم دارند. یک مثال از این تغییر مفهوم را می‌توانید حمله‌ی هکری در یک شبکه‌ی کامپیوتری را در نظر بگیرید[۲۰] . توضیح مفهوم داده‌ی «معمولی» با داده‌ی «حمله» در طی زمان تغییر می‌کند. همیشه حمله‌ها در حال انجام هستند ولی انواع و روش‌های آن تغییر می‌کند. یک جریان داده‌ای مانند DS شامل یک مجموعه k عضوی از منبع داده را در نظر بگیرید که در آن منابع داده با S_i نشان داده می‌شود و توزیع این منابع شناخته شده است($i = 1, ..., k$). در زمان t یک یا چند منبع داده فعال وجود دارد. فرض کنید $w_i(t) \in [0, 1]$ برابر با تاثیر منبع داده S_i در زمان t باشد. توزیع داده‌ی جریان داده‌ی DS در زمان t توسط رابطه زیر مشخص می‌شود[۱۴]:

$$\sum^{DS}(t) = \{w(1)S_1, ..., w(k)S_k\}$$

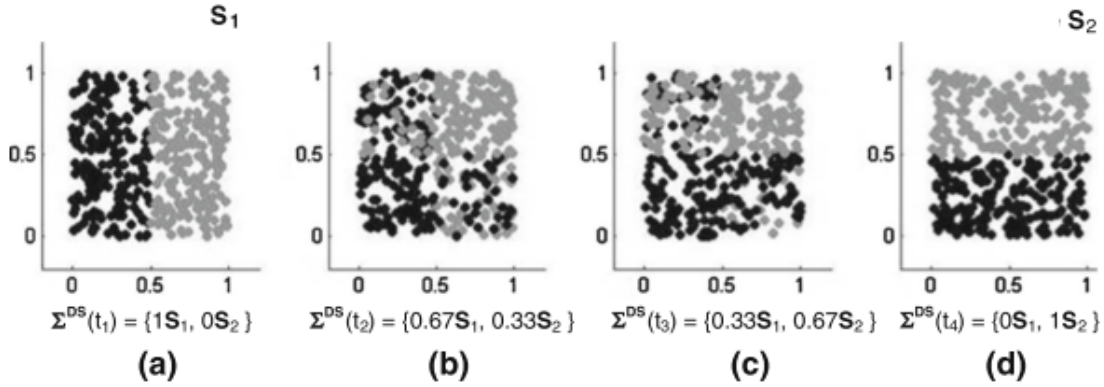
۲-۱-۳ رانش مفهوم

اگر فرض کنیم که یک منبع داده DS شامل k منبع جریانی مستقل S با وزن اثر w باشد، اگر در دو زمان t_1 و t_2 رابطه زیر برقرار باشد می گوییم رانش مفهوم [۳] رخ داده است.

$$\sum^{DS}(t_1) \neq \sum^{DS}(t_2) \qquad (۲-۱)$$

[۲] Concept

[۳] Concept Drift



شکل ۲-۱: مثالی از تغییر تدریجی منبع S_1 به منبع S_2. در این شکل کلاس y_1 با رنگ خاکستری و کلاس y_2 با رنگ مشکی مشخص شده است[۱۹].

در تصویر ۲-۱ می‌توانید یک نمونه از رانش مفهوم که بصری‌سازی شده است را طی زمان‌های مختلف مشاهده کنید.

بر اساس رابطه ۲-۱ هر نوع تغییری(حتی کوچک‌ترین تغییرات) در منبع جریان، یک رانش مفهوم حساب می‌شود. بنابراین رانش‌های مفهوم را به دو دسته اساسی تقسیم می‌کنیم [۷]:

رانش مفهوم مجازی

اگر رانش مفهوم تأثیر مستقیمی بر حدود تصمیم‌گیری نداشته باشد(احتمال پیشین داده‌ها تغییر نکند) اما رانش بر روی تابعِ چگالی احتمال اثر بگذارد، گوییم که رانش مفهوم مجازی [۴] رخ داده است.

رانش مفهوم حقیقی

اگر رانش مفهوم علاوه بر تغییر تابع چگالی احتمال، تأثیر مستقیم بر حدود تصمیم‌گیری بگذارد و احتمال‌های پیشین را تغییر دهد، گوییم رانش مفهوم حقیقی [۵] رخ داده است. روش‌های یادگیری در محیط‌های جریان‌های داده‌ای باید به گونه‌ای باشند که با رانش مفهوم حقیقی سریعا سازگار شوند.

[۴] Virtual Concept Drift

[۵] Real Concept Drift

۲-۲ پنجره‌های زمانی

از آن‌جایی که داده‌های جریانی امکان دارد بینهایت باشند، تنها مقدور است که بخشی از تمام داده‌ی جریانی را پردازش کنیم. این بخش مورد توجه از داده توسط یک پنجره زمانی از نمونه‌های داده تعریف می‌شود.

$$W[i,j] = (x_i, x_{i+1}, ..., x_j) \qquad (۲-۲)$$

بر اساس این تعریف، انواع مختلفی از پنجره زمانی بیان می‌شود: پنجره نقطه عطفی [۶] ،پنجره کشویی [۷] ،پنجره محو شونده [۸] و پنجره یک‌بر [۹].

۲-۲-۱ پنجره نقطه عطفی

در پنجره نقطه عطفی، ما به تمام داده جریانی از زمان شروع نمونه اول تا زمان فعلی نمونه t_c توجه می‌کنیم: پنجره به شکل $W[1, c]$ تعریف می‌شود[۸]. استفاده از پنجره نقطه عطفی بدین معناست که تمامی تراکنش‌ها در پنجره به شکل یکسانی برای ما اهمیت دارند؛ هیچ تفاوتی داده‌های گذشته و حال وجود ندارد. به طور مداوم که داده‌های جریانی تغییر می‌کنند، مدل با استفاده از داده قدیمی نمونه‌ها ساخته می‌شود که حتی ممکن است که با داده‌های جدید ناسازگار باشند. برای تاکید بیشتر روی داده‌های جدیدتر می‌توان از پنجره‌های کشویی، محوشونده و یک بر استفاده کرد.

۲-۲-۲ پنجره کشویی

در پنجره کشویی که با $W[t_c - w + 1, t_c]$ نشان داده می‌شود ما تنها به w تراکنش اخیر توجه می‌کنیم و بقیه داده‌های جریانی نادیده گرفته می‌شوند[۲۱]. نتیجه کاوش وابسته به اندازه پنجره است که w بیانگر آن است. اگر w خیلی بزرگ باشد و رانش مفهوم رخ دهد، پنجره ممکن است شامل داده‌ها و اطلاعات منقضی شده باشد و دقت مدل کاهش پیدا می‌کند. اگر w کوچک باشد، پنجره ممکن است شامل داده‌های ناکارامد باشد

[۶] Landmark Window
[۷] Sliding Window
[۸] Fading Window
[۹] Tilted Window

و مدل دچار بیش‌برازش گردد. کارهای قبلی یک مقدار ثابت را برای اندازه پنجره کشویی در نظر می‌گرفتند که توسط کاربران و با آزمایش تعیین می‌شد. اخیرا چند کار [۵] برای ارایه پنجره کشویی انعطاف پذیر ارایه شده است که در آن اندازه پنجره بر اساس دقت مدل تغییر می‌کند؛ زمانی که دقت زیاد باشد اندازه پنجره بزرگ می‌شود و زمانی هم که دقت کم شود، پنجره کوچک می‌شود.

۲-۲-۳ پنجره محوشونده

در پنجره‌های محو شونده، هر نمونه از داده‌های جریانی با یک وزن که با زمان رسیدن آن نمونه در ارتباط است شناسایی می‌شود؛ بنابراین داده‌های جدیدتر می‌رسند وزن بیشتری از داده‌های قدیمی‌تر دارند[۱۶]. با استفاده از پنجره محو شونده، می‌توان تاثیر(اهمیت) داده‌های قدیمی منقرض شده را کاهش داد تا در نتیجه کاوش تاثیری نداشته باشند. معمولا از یک تابع نمایی نزولی مانند تابع زیر برای پنجره زمانی استفاده می‌شود:

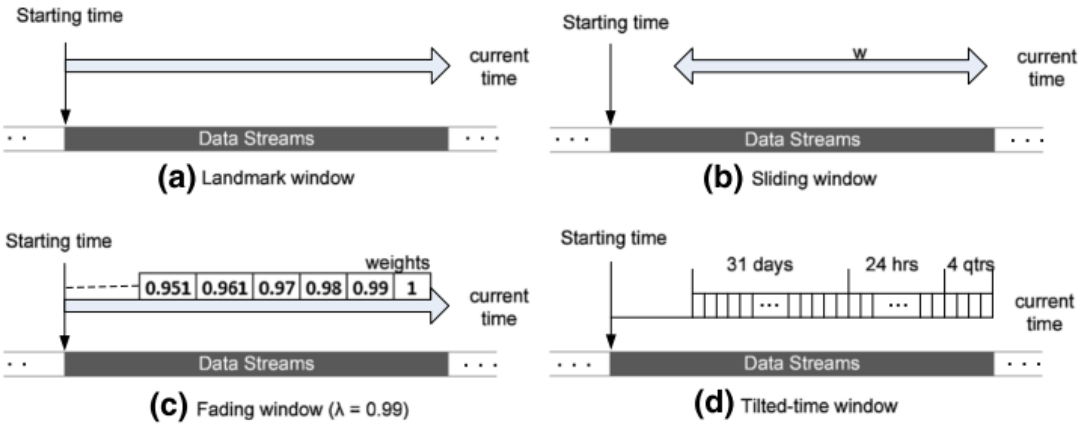
$$f(\triangle t) = \lambda^{\triangle t} (۰ < \lambda < ۱)$$

که در این تابع $\triangle t$ قدمت(سن) نمونه داده‌ی جریانی و برابر با تفاوت زمانی فعلی و زمان مشاهده داده است. پنجره محو شونده نیاز به انتخاب یک پارامتر محوشوندگی مانند λ دارد که معمولا یک عدد در بازه $[۰/۹۹, ۱]$ برای کاربردهای واقعی انتخاب می‌شود[۲۰].

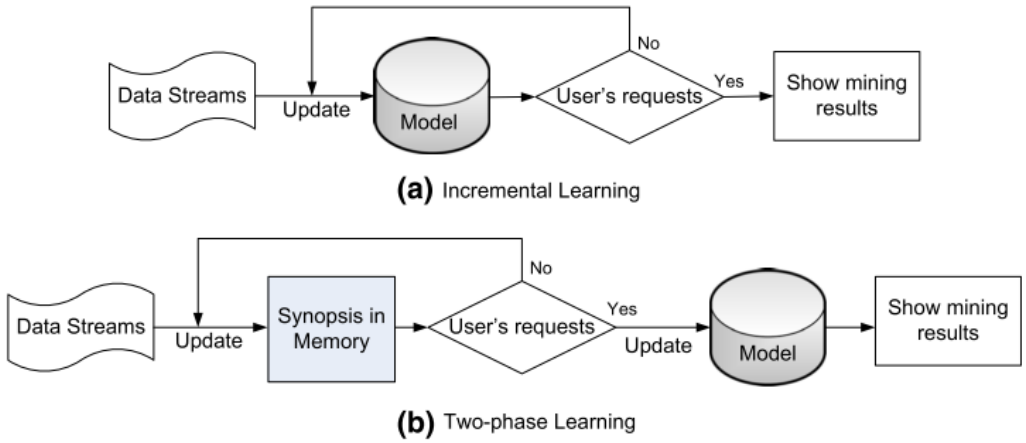
۲-۲-۴ پنجره یک‌بر

پنجره یک‌بر یک نوع پنجره زمانی شبیه پنجره محو شونده و کشویی است[۳]. این پنجره در سطوح مختلف ریزدانگی که با توجه به تاخر داده شکل گرفته‌اند اعمال می‌شود. پنجره زمانی یک‌بر تقریبا تمام مجموعه داده را ذخیره کرده و یک تعادل بین فضای مورد نیاز برای ذخیره‌سازی و دقت برقرار می‌کند. البته این مدل می‌تواند پس از اجرای طولانی غیر پایدار باشد.

تصویر ۲-۲[۲۰] چهار نوع مختلف پنجره زمانی را نشان می‌دهد. برای پنجره محو شونده λ برابر با ۰/۹۹ تنظیم شده است و وزن نمونه‌ها کاهش می‌یابد. برای پنجره یک‌بر، ما جهار ربع اخیر یک ساعت را ذخیره کرده‌ایم، سپس ۲۴ ساعت گذشته و ۳۱ روز اخیر نمایش داده شده‌اند.



شکل ۲-۲: پنجره‌های مختلف زمانی



شکل ۳-۲: الف- رویکرد افزایشی به یادگیری. ب- رویکرد دو گامی

۲-۳ رویکردهای محاسباتی

براساس انواع مختلف پنجره زمانی، دو رویکرد محاسباتی مختلف برای پردازش داده‌های جریانی وجود دارد.

۲-۳-۱ یادگیری افزایشی

یادگیری افزایشی[۱۰] یک رویکرد محاسباتی برای داده‌های جریانی است[۲۴]. در این رویکرد، مدل به صورت افزایشی تغییری می‌کند تا با تغییرات در داده‌هایی که در حال آمدن هستند، سازگار شوند. دو طرح کلی برای بروزرسانی مدل وجود دارد: بروزرسانی با هر نمونه داده و بروزرسانی با پنجره. به عنوان نمونه ستریت [۱۱] و همکارانش [۲۴]یک مجمع از رده‌بندها برای داده‌های جریانی توسعه دادند که یک پنجره از داده‌های ورودی را می‌گیرد و مدل را با تنظیم وزن‌های هر رده‌بند یا جایگزین کردن رده‌بندهای قدیمی با نمونه‌های جدیدشان سازگار می‌کند. شکل ۲-۳ ـ الف رویکرد یادگیری افزایشی را نمایش می‌دهد. مزیت این رویکرد این است که نتیجه برای هر نمونه(یا پنجره‌ای از نمونه‌ها) در دسترس است ولی به منبع محاسباتی بیشتری نیاز است.

۲-۳-۲ یادگیری دوگامی

یادگیری دوگامی[۱۲]، که با نام یادگیری آنلاین ـ آفلاین نیز شناخته می‌شود یک رویکرد محاسباتی برای داده‌های جریانی است[۳]. ایده بنیادی این است که کاوش بر داده‌ها را به دو بخش تقسیم کنیم. در گام اول(گام آنلاین)، چکیده‌ای از داده در لحظه ایجاد می‌شود. در گام دوم(گام آفلاین)، بر مبنای درخواست کاربر، پردازش روی چکیده‌هایی که در گام قبل ایجاد شده بود انجام می‌شود. به عنوان مثال، آگراول [۱۳] و همکارانش[۲]، یک روش آنلاین ـ آفلاین برای خوشه‌بندی داده‌های جریانی ارایه کرده‌اند که در بخش آنلاین، خلاصه‌ای از اطلاعات آماری داده جریانی به صورت برخط گردآوری می‌شود. در طول بخش آفلاین، از این داده‌ها برای خوشه بندی سطح بالا استفاده می‌شود. رویکرد یادگیری دوگامی در شکل نشان داده شده است. این روش قابلیت پردازش داده های جریانی را با سرعت بیشتری داد منتها محدودیت آن این است که کاربر باید تا فراهم شدن نتیجه، منتظر بماند. شکل ۲-۳ ـ ب رویکرد یادگیری آنلاین ـ آفلاین را نمایش می‌دهد.

[۱۰] Incremental Learning
[۱۱] Street
[۱۲] Two-Phase Learning
[۱۳] Aggarwal

۲-۴ اعتبار سنجی

در روش‌های سنتی یادگیری ماشین با مقدار داده‌ی محدود، فرآیند اعتبارسنجی بر استفاده‌ی بیشینه از داده تمرکز داشت. جداسازی [۱۴]، اعتبار سنجی متقابل [۱۵] و تک‌گذاری [۱۶] روش‌های استاندارد اعتبار سنجی هستند. در روش جداسازی به شکل تصادفی مجموعه‌ی داده به دو زیر مجموعه، یکی برای آموزش و دیگری برای آزمایش تقسیم می‌شود؛ نسبت‌های رایج برای تقسیم داده به مجموعه آموزش و آزمایش نصف و یک‌سوم است. در روش اعتبار سنجی متقابل با k دسته، داده به k مجموعه‌ی مساوی و مستقل از نمونه‌ها تقسیم می‌شد و یکی از این زیر مجوعه‌ها برای آزمایش و باقی آن‌ها($k-1$)برای آموزش ادغام می‌شد؛ در این روش فرآیند اعتبارسنجی k بار و هر بار با یک زیر مجموعه برای آزمایش انجام می‌شود. روش تک‌گذاری نیز یک نوع اعتبار سنجی متقابل است که در آن k با تعداد کل داده‌ها برابر است.

در محیط جریان‌های داده‌ای، از آن‌جایی که داده می‌تواند نامحدود باشد، اعتبار سنجی بر ارزیابی مدل در صحنه‌های مختلف متمرکز است[۲۰]. یک روش شناخته‌شده رسم منحنی یادگیری با ذخیره‌سازی کارایی مدل در طول زمان است؛ این منحنی نشان خواهد داد که چقدر مدل پس از مشاهده‌ی داده‌های بیشتر، بهتر شده است و مدل چگونه با رانش مفهوم سازگار می‌شود. یک الگوریتم برتری خود را به سایر روش‌ها زمانی نشان می‌دهد که منحنی یادگیریش بیشتر مواقع از سایر الگوریتم‌ها بالاتر باشد.

روش جداسازی و روش پی‌درپی [۱۷] دو روش پر کاربرد برای اعتبارسنجی جریان‌های داده‌ای هستند[۲۰]. در روش جداسازی، نمونه‌های داده به چانک‌های مختلفی تقسیم می‌شوند؛ از هر چانک داده ابتدا برای آزمایش و سپس برای بروزرسانی مدل استفاده می شود. روش جداسازی زمانی که رانش مفهوم رخ داده، ترجیح داده می‌شود چون این روش اجازه می‌دهد که مدل با آخرین تغییرات داده سازگار شود. روش پی‌درپی(یا آزمایش_سپس_آموزش [۱۸] به شکل لایه لایه) یک روش دیگر برای ارزیابی جریان‌های داده‌ای است[۶]. هر نمونه داده ابتدا و پیش از این‌که برای آموزش به صورت افزایشی استفاده شود، برای آزمایش استفاده می‌شود. این روش می‌تواند یک حالت خاص برای روش جداسازی در نظر گرفته شود اگر اندازه چانک برابر با یک باشد. مزیت این روش این است که نیازی به تعریف از پیش اندازه چانک نیست ولی متاسفانه کارایی این

[۱۴] Hold-out
[۱۵] Cross-Validation
[۱۶] Leave-One-Out
[۱۷] Prequential
[۱۸] Test-Then-Train

الگوریتم در زمان محدود مبهم است چراکه اشتباهات اولیه مدل به سرعت در طول زمان کاهش می‌یابد.

معیارهای ارزیابی

به طور عمومی معیارهای ارزیابی روش‌های سنتی یادگیری ماشین، می‌تواند برای ارزیابی یادگیرنده‌ها در جریان‌های داده نیز استفاده شود. برای رده‌بندی داده‌ها، صحت [۱۹] و تابع ضرر ۱ − ۰ دو معیار رایج هستند. برای ارزیابی مداوم کارایی رده‌بندی در داده‌های جریانی، گیم [۲۰] و همکارانش [۹] یک روش ضرر پی‌درپی که با انواع مختلف پنجره کار می‌کند را ارایه کرده‌اند. برای از یادبردن پی‌درپی خطا اثبات شده است که همگرایی خطای بیز در داده‌های ایستا، روش را برای تشخیص رانش کارا می‌سازد. این روش می‌تواند به سادگی روی سایر معیارهای کارایی نیز اعمال شود.

۲-۵ نرم‌افزارهای کاربردی

نرم‌افزارهای مختلف کاربری و متن‌باز برای تحقیقات دانشگاهی در حوزه‌ی یادگیری و داده‌کاوی در محیط‌های پویای جریان‌های داده‌ای وجود دارد.

WEKA

شناخته‌شده‌ترین ابزار داده‌کاوری در محیط دانشگاهی است. این ابزار شامل یک مجوعه از الگوریتم‌های پردازی داده، رده‌بندی، رگرسیون، خوشه‌بندی، قواعد باهم‌آیی و بصری سازی داده است[۱۰].

MOA

نرم افزار MOA [۶] یک فریمورک متن‌باز محبوب برای داده کاوی در داده های جریانی است که توسط دانشگاه وایکیتو نیوزلند [۲۱] و بر مبنای چهارچوب WEKA توسعه داده شده است. در این نرم افزار که توسط زبان جاوا پیاده سازی شده، تعداد زیادی ابزار مناسب جهت پیاده سازی و تست الگوریتم های یادگیری و خوشه بندی و تشخیص رانش مفهوم وجود دارد. برای نمونه الگوریتم‌های درخت تصمیم خیلی سریع و

[۱۹] Accuracy

[۲۰] Gama

[۲۱] University of Waikato, New Zealand

مجمع رده‌بند ها که در ادامه توضیح داده می‌شوند در این چهارچوب موجود است. این ابزار تعدادی تولیدکننده [۲۲] داده جریانی مانند مفهوم‌های SEA ،ابرصفحه‌ی دوران کننده [۲۳] و STAGGER را فراهم کرده است.

Rapid-Miner

یکی دیگر از ابزارهای متن‌باز برای داده‌کاوی است[۱۲]. RapidMiner بسیار قدرتمندتر از WEKA است و تمام الگوریتم‌های WEKA و سایر الگوریتم‌های پیشرفته را دارد. این ابزار خلاقانه‌تر است و این قابلیت را داد که پروسه داده‌کاوی را به شکل یک دنباله از عملگرها تعریف کرد. همچنین RapidMiner ابزارهای بیشتری را بصری‌سازی فراهم کرده است.

۶-۲ مخزن‌های داده

داده‌های واقعی بسیار کمی برای ارزیابی روش‌های یادگیری در داده‌های جریانی وجود دارد[۲۰]. یک دلیل این موضوع می‌تواند این باشد که محققان روش‌های سنتی داده‌کاوی و یادگیری ماشین معمولا داده‌های خود را به قدری کوچک نگه‌می‌داشتند تا با روش‌های یادگیری دسته‌ای [۲۴] سازگار باشد.

یک دلیل دیگر می‌تواند مساله حریم شخصی در انتشار داده‌های که خیلی بزرگ هستند باشد. محققان معمولا از داده‌های خصوصی برای اثبات سیستم‌هایشان استفاده می‌کنند که نمی‌توانند آن‌ها را منتشر کنند. برای غلبه بر این کمبود، بعضی از مجموعه داده‌های ترکیب شده با تعداد نامحدود داده ساخته شده‌اند. برای مثال تولید کننده درخت تصادفی، تولید کننده مفهوم SEA و ابرصفحه‌های چرخنده تولید شده‌اند. این تولیدکننده‌های داده در چهارچوب MOA پیاده‌سازی شده‌اند[۶]. بعضی از مخزن‌های داده با مجموعه‌های داده‌ی بزرگ نیز برای ارزیابی روش‌های یادگیری در داده‌های جریانی استفاده می شود:

[۲۲] Generator
[۲۳] Rotating Hyperplane
[۲۴] Batch Learning

UCI Machine Learning Repository

یک مخزن آنلاین معتبر[۲۵] و شناخته شده برای آزمایش و آنالیز الگوریتم‌های یادگیری ماشین است. سه مجموعه داده پر استفاده در چندین مقاله برای ارزیابی جریان‌ها Covertype Forest و Poker-Hand و Elec-tricity هستند[۲۰].

KDD Cup Center

رقابت‌های سالانه‌ی داده‌کاوی و کاوش دانش توسط ACM Special Interest Group[۲۶] سازمان‌دهی می‌شود. معمولا داده‌هایی که برای این رقابت تولید می‌شود می‌تواند منبع خوبی برای ارزیابی الگوریتم‌های یادگیری ماشین در داده‌های جریانی باشد.

[۲۵] http://www.ics.uci.edu/ mlearn/mlrepository.html

[۲۶] http://www.kdd.org/

فصل ۳

روش‌های یادگیری در داده‌های جریانی

رده‌بندی فرایند یافتن یک مدل عمومی از داده‌های گذشته است به طوری که بتوان آن مدل را روی داده‌های جدید اعمال کرد. رده‌بندی از دو گام تشکیل شده است: گام یادگیری(آموزش) و گام تست. در بخش یادگیری سیستم تلاش می‌کند که یک مدل از مجموعه‌ی داده‌های نمونه‌ی مجموعه آموزشی یاد بگیرد؛ در گام تست از این مدل برای برچسب‌زنی به داده‌هایی که برچسب زده‌نشده‌اند استفاده می‌شود. در ادبیات داده‌های جریانی، تعداد زیادی الگوریتم رده‌بندی مانند درخت‌های تصمیم، رده‌بند بیزین، ماشین‌های بردار پشتیبان، k نزدیک‌ترین همسایگی و روش‌هایی که مجمعی از رده‌بندها را می‌سازند وجود دارد. در این فصل به بررسی این روش‌ها می‌پردازیم و روش کار هر کدام را به شکل مختصر توضیح می‌دهیم.

۳-۱ درخت هافدین

درخت هافدین یک رده‌بند بر مبنای درخت تصمیم برای داده‌های جریانی است[۹]. روش‌های سنتی درخت تصمیم، برای انتخاب خصیصه تقسیم نیاز به چندین بار پویش داده دارند که این موضوع در محیط داده‌های جریانی، عملا نشدنی است. سایر روش‌های رده‌بندی داده‌های جریانی نیز معمولا کاستی‌هایی مانند موارد زیر دارند:

- حساسیت زیاد به ترتیب نمونه‌ها
- کارایی کم. در بعضی موارد آن‌های از الگوریتم‌های یادگیری دسته‌ای کندترند.

درخت هافدین که یکی از روش‌های جدید یادگیری برای جریان‌هاست چالش‌های زیر را حل می‌کند:

- عدم قطعیت در زمان یادگیری. زمان یادگیری در درخت هافدین برای هر نمونه ثابت است و این بدان معنیست که درخت هافدین برای کاوش در داده‌های جریانی مناسب است.
- زمانی که تعداد نمونه کافی برای ساخت درخت پویش شود، نتیجه درخت در روش هافدین تقریبا مشابه(یا برابر) با درخت‌های ساخته شده با روش‌های مرسوم یادگیری دسته‌ای است.

برای تعامل با چالش پویش چندباره داده‌ها از حد هافدین برای انتخاب یک خصیصه تقسیم بهینه، پس پویش تعداد کافی نمونه استفاده می‌شود. فرض کنید N تعداد مشاهدات مستقل از یک متغییر تصادفی r باشد که این متغییر تصادفی در محدود R و با میانگین $\overline{r}$ است. حد هافدین تضمین می کند که میانگین درست r حداقل از $\overline{r} - \epsilon$ با احتمال $\delta - ۱$ بزرگ‌تر باشد. در این رابطه δ پارامتری است که کاربر انتخاب می‌کند.

$$P(E[r] \geq (\overline{r} - \epsilon)) \geq ۱ - \delta, \epsilon = \sqrt{\frac{R^۲ ln(\frac{۱}{\delta})}{۲N}}$$

چیزی که حد هافدین را جالب توجه می‌کند، امکان گرفتن نتیجه‌های مشابه بدون در نظر گرفتن توزیع احتمالی‌ست که مشاهدات را تولید می‌کند. به عنوان نمونه اگر اختلاف بهره اطلاعاتی بین دو خصیصه A و B (که می‌دانیم بهره اطلاعاتی خصیصه A بیشتر از خصیصه B است) برابر با ۰/۳ و $e = ۰/۱$ باشد این معنی را می‌دهد که در آینده، تفاوت بین بهره اطلاعاتی A و B حداقل $۰/۲ = ۰/۱ - ۰/۳$ است. به عبارت دیگر با احتمال $\delta - ۱$، هنگامی که تفاوت بهره اطلاعاتی مشاهده شده بزرگ‌تر از ϵ باشد یک خصیصه در مقایسه با دیگر خصیصه‌ها قالب است.

فرض کنیم $G(x_i)$ یک معیار اکتشافی برای انتخاب کردن خصیصه جداسازی باشد و پس از مشاهده N نمونه، x_a و x_b اولین و دومین خصیصه برتر برای خصیصه جداسازی باشند. با این مفروضات، در درخت هافدین، متغییر r از رابطه $r = \triangle G = G(X_a) - G(X_b)$ تعیین می‌شود. اگر $\overline{r} = \triangle \overline{G}$ $\overline{G}(X_a) - \overline{G}(X_b)$ باشد که در آن ϵ از رابطه بالا محاسبه شده است، می‌گوییم با اطمینان $\delta - ۱$ این تفاوت بزرگتر از $۰ < \overline{r} - \epsilon$ است. بنابراین خصیصه x_a برای جداسازی انتخاب می‌شود.

نویسندگان الگوریتم درخت هافدین را توسط یک یادگیرنده با نام «درخت تصمیم خیلی سریع»(VFDT) پیاده‌سازی کرده‌اند. این پیاده‌سازی شامل بهبودهایی برای استفاده‌های خاص، مانند راهبرد محدود کردن

گره، راهندازی سریع با استفاده از یادگیرنده‌های مبتنی بر RAM و قابلیت بازپویش نمونه‌های قبلی زمانی که سرعت گذر داده کم است می‌باشد.

الگوریتم درخت هافدین، یک الگوریتم با دقت بالاست که می‌تواند بسیار خوب با مجموعه‌ی داده‌های بزرگ کار کند، ولی این الگوریتم نمی‌تواند با مساله رانش مفهوم در داده‌های جریانی تعامل کند[۲۰]. الگوریتم CVFDT [۱۳] یک تعمیم از الگوریتم درخت هافدین است که برای تعامل با رانش مفهوم در داده‌های جریانی سازگار شده است. CVFDT ، بعضی آماره‌های کارآمد را برای بررسی کردن اعتبار تصمیم‌های قبلی در کنار هر گره درخت نگهداری می‌کند. زمانی که یک داده وارد می‌شود، به طور مداوم این آماره‌های قرارگرفته در کنار گره‌های درخت بروزرسانی می‌شوند. با کمک پنجره‌های جابه‌جا شونده روی داده، این الگوریتم اثر داده‌هایی که از پنجره خارج هستند را در آماره‌های هر گره نادیده می‌گیرد. این پویش دوره‌ای گره‌های درخت باعث تشخیص رانش مفهوم می‌شوند. اگر رانش مفهوم ظاهر شده باشد، CVFDT به صورت موازی شاخه‌های جایگزین را با انتخاب بهترین خصیصه‌های جدید و حذف شاخته‌های قدیمی، اگر دقتشان کم باشد، گسترش می‌دهد.

۲-۳ رده‌بند بیزین

سیدل [۱] و همکارانش [۲۴] یک روش جالب برپایه نمایه‌سازی [۲] رده‌بند ارایه داده‌اند که درخت بیز نامیده می‌شود. درخت بیز از یک درخت مخلوط ـ گوسین سلسله مراتبی [۳] جهت نمایانکردن تمام مجموعه‌ی داده استفاده می‌کند. هر گره از درخت شامل آماره‌های از نمونه‌های داده مثل مستطیل محدودکننده کمینه [۴] ، تعداد نمونه‌های داده، جمع خطی و مجموع مربعات تمام داده است. برای حل مساله‌ی رده‌بندی چند کلاسه، یک درخت بیز برای هر کلاس ساخته می‌شود.برای هر داده‌ی ست مانند x ، الگوریتم سعی می‌کند که نزدیک‌ترین مجموعه گره‌ها را که مرز نامیده می‌شوند پیدا کند. احتمال تعلق x به کلاس c_i توسط رابطه زیر محاسبه می‌شود:

[۱] Seidl
[۲] Index-Based
[۳] Hierarchical Gaussian-mixture Tree
[۴] Minimum Bounding Rectangle

$$P(c_i|x) = \left(\sum_{e_s \in E_i} \frac{n_{e_s}}{n} g(x, \mu_{e_s}, \sigma_{e_s}) \right) \star P(c_i)/P(x)$$

که در این رابطه e_s یک گره درخت در مخموعه مرز E_i است. n_{e_s} و μ_{e_s} و σ_{e_s} به ترتیب، تعداد نمونه‌های داده،مرکز و انحراف گره e_s هستند. برای مشخص کردن برچسب نمونه‌ی x ، احتمال برای تمام کلاس‌ها حساب می‌شود و کلاسی که بیشترین احتمال را داشته‌باشد به عنوان کلاس نمونه x اعلام می‌شود. درخت بیز یک رده‌بند است که پس از مدت کمی از شروع، می‌تواند نتیجه و تصمیم داشته باشد و سپس آن دقت‌ها را با انتخاب مرزهای دقیق‌تر افزایش دهد.

۳-۳ شبکه عصبی

لیت [۵] و همکارانش [۱۵] روش شبکه‌های عصبی داده‌دانه تغییر پذیر [۶] (eGNN) را جهت رده‌بندی داده‌های جریانی ارایه داده‌اند. دو گام در eGNN وجود دارد؛ در گام اول، eGNN از نرون‌های T-S برای ساختن ریزدانه‌های اطلاعات برای داده‌هایی که درحال آمدن هستند، استفاده می‌کند؛ در گام دوم، شبکه عصبی روی این ریزدانه‌های اطلاعاتی، به جای داده‌ی اصلی، ساخته می‌شود. یک ریزدانه مرتبط با برچسب یک کلاس به شکل یک تابع عضویت سه‌گوش [۷] تعریف می شود که بعدا جهت تطبیق با داده‌های جدید تغییر می‌کند.

وزن‌ها با یک ثابت زوال [۸] کاهش پیدا می‌کنند؛ این پردازش کمک می‌کند که درجه اهمیت ریزدانه‌های منقضی شده کاهش پیدا کند. به شکل پایه، eGNN از نمونه‌های کلاس در فرم ریزدانه‌های اطلاعاتی جهت انجام رده‌بندی استفاده می‌کند. زمانی که داده‌های تست می‌آیند یک نرون بیشینه‌گیری، بهترین پیش‌بینی ریزدانگی را انتخاب می‌کند و برچسب آن را به نمونه، جهت پیشبینی کلاسش، اختصاص می‌دهد. eGNN می‌تواند در مساله رده‌بندی در محیط‌های که دایما تغییر می‌کند استفاده شود اما نیاز به زمان طولانی آموزش، هنوز یک محدودیت برای شبکه‌های eGNN به منظور کار با داده‌های بسیار بزرگ است. البته، در قسمت آزمایش‌های مقاله اصلی، مجموعه‌های داده‌ی کوچک برای ارزیابی این روش استفاده شده است؛ سپس

[۵] Leite

[۶] Evolving Granular Neural Network

[۷] Triangular membership function

[۸] Decay

نویسنده، این روش را جهت کارایی بهتر و کار به شکل نیمه نظارتی [۹] گسترش داده‌است تا با مجموعه‌ی داده‌هایی که کاملا برچسب گذاری نشده‌اند هم کار کند.

۳-۴ ماشین‌های بردار پشتیبان

ماشین‌های بردار پشتیبان [۱۰] کارایی خود را در بسیاری از مساله‌های یادگیری ماشین با داده‌های ایستا نشان داده‌اند[۲۰]. البته استفاده از ماشین‌های بردار پشتیبان برای کاربردهایی با مقیاس بزرگ بسیار پرهزینه است. اگر N تعداد نمونه‌های داده باشد، این الگوریتم از پیچیدگی زمانی $O(N^3)$ و از پیچیدگی حافظه‌ای $O(N^2)$ است. برای کار کردن با داده‌های بسیار بزرگ، تی‌سنگ [۱۱] و همکارانش، الگوریتم ماشین بردار هسته [۱۲] (CVM) را ارایه داده‌اند تا پیچیدگی الگوریتم ماشین بردار پشتیبان را کاهش دهند. این الگوریتم از حداقل گوی نزدیک(MEB) برای این کار استفاده می‌کند. یک MEB ابر کره [۱۳] (کره با بیش از سه بعد) است که نماینده یک مجموعه از نمونه‌های داده کنار آن است[۲۵] . این الگوریتم ابتدا MEB های نماینده را که تخمین خوبی از داده اصلی باشند، پیدا می‌کند. سپس مساله بهینه‌سازی پیدا کردن محدوده‌ی بیشینه بر روی این مجموعه‌های MEB اعمال می‌شود.

ری [۱۴] و همکارانش [۲۳] الگوریتم دیگری با نام StreamSVM که یک بهبود از الگوریتم CVM است را برای کار با داده‌های جریانی ارایه کرده‌اند. در این الگوریتم، یک MEB شعاع انعطاف پذیری است، هرزمان نمونه‌های جدید آموزش اضافه می‌شوند، این شعاع افزایش می‌یابد. این الگوریتم زمانی که تخمین‌ها درست باشند با بهینه‌ترین روش‌ها قابل رقابت است، اما StreamSVM هنوز قابلیت تشخیص رانش مفهوم در داده‌های جریانی را ندارد.

[۹] Semi-Supervised
[۱۰] Support Vector Machine
[۱۱] Tsang
[۱۲] Core Vector Machine
[۱۳] Hypersphere
[۱۴] Rai

۳-۵ رده‌بند نزدیک‌ترین همسایگی

در مورد خوشه بندی جریان‌های داده‌ای(که مورد بحث این گزارش نیست)، یک روش خوشه‌بندی با نام CluStream وجود دارد. این روش از تعریفی تحت عنوان ریز_خوشه [۱۵] برای خوشه‌بندی داده‌ها استفاده می‌کند. یک ریز_خوشه برای تعدادی داده، شامل آماره‌هایی نظیر مجموع مربعات نقاط، مجموع نقاط، مجموع مربعات زمان رخداد نقاط، مجموع خطی زمان رخداد و تعدا نمونه‌هاست. این روش خوشه‌بندی به جای ذخیره کل داده‌های خوشه، در گام آنلاین ریز_خوشه‌های آن‌ها که حجم کمتری را اشغال می‌کند و پردازش آن آسان‌تر است را ساخته و ذخیره می‌کند و در گام آفلاین به جای خوشه‌بندی کل داده‌ها، این ریز_خوشه‌ها را خوشه‌بندی می‌کند[۲۰]. روش On-Demand-Stream [۲] یک گسترش از روش CluStream است که همانند این روش، از رویکرد آنلاین_آفلاین و پنجره یک‌بر استفاده می‌کند. تفاوت ریز_خوشه در این روش با ریز_خوشه در روش CluStream این است که برچسب کلاس نیز به ریز_خوشه اضافه می‌شود و ریز_خوشه‌ها فقط شامل نمونه‌هایی از یک کلاس مشخص و یکسان هستند. در رده‌بندی آفلاین، سعی می‌شود که بهترین پنجره از نمونه‌های داده که افق زمان [۱۶] نامیده می‌شود انتخاب شود و ریز_خوشه‌ها از این افق زمانی استخراج می‌شوند. روش On-Demand-Stream از یک رده‌بندی ۱NN برای نسبت‌دادن داده‌ی آزمایش به نزدیک‌ترین ریز_خوشه استفاده می‌کند.

با الهام‌گرفتن از ایده M-tree، ژنگ و همکارانش [۲۷] یک روش ساختار Lazy-tree برای نمایه سازی ریز_خوشه‌ها پیشنهاد داده‌اند. این کمک می‌کند که پیچیدگی زمانی روش نزدیک‌ترین همسایگی اگر فرض کنیم N تعداد ریز_خوشه‌ها در Lazy-tree باشد، از $O(N)$ به $O(log(N))$ کاهش یابد. درخت شامل سه عملگر اصلی است: جستجو، حذف و افزودن. اضافه‌کردن و حذف‌کردن برای افزودن گره‌های جدید و حذف‌کردن گره‌های منقضی شده به کار می‌رود و تضمین می‌کنند که درخت متعادل بماند. عملگر جستجو برای رده‌بندی داده‌های تست استفاده می‌شود.

[۱۵] Micro-Cluster

[۱۶] Time Horizon

۳-۶ مجمع رده‌بندها

روش‌های Bagging & Boosting برتری خود را از طریق آزمایش‌ها روی مجموعه‌ی داده‌های سنتی اثبات کرده است، نظر به این موضوع، بسیاری از محققین تلاش کرده‌اند که این روش‌ها را برای کار روی داده‌های جریانی سازگار کنند. اوزا [۱۷] و همکارانش [۲۲] روشی تحت عنوان Bagging & Boosting آنلاین ارایه کرده‌اند که یکی از اولین کارها برای سازگار کردن این روش بوده است. با دید آماری، هر نمونه از داده‌های آموزشی k بار در مجموعه‌ی آموزشی اعضای رده‌بند با احتمال زیر مشاهده می‌شود:

$$P(k) = \begin{pmatrix} N \\ k \end{pmatrix} \left(\frac{۱}{N}\right)^k \left(۱ - \frac{۱}{N}\right)^{N-k}$$

که در رابطه k اندازه مجموعه‌ی آموزشی و N اندازه مجموعه‌ی داده‌هاست. در داده‌های جریانی می‌توان فرض کرد که تعداد نمونه‌های داده نامحدود است، لذا $N \to \infty$؛ بنابراین احتمال $P(k)$ به توزیع احتمال پوآسون میل می‌کند که رابطه آن مشابه زیر است:

$$Poisson(۱) = exp(۱)/k!$$

با این مشاهده، اوزا و همکاران روش Online Bagging را جهت نسبت دادن هر شی از داده به یک وزن متناسب با توزیع پواسون ارایه کردند. در Online Boosting وزن‌های داده‌هایی که در حال آمدن هستند و اعضای رده‌بندها بر اساس نسبت خطای اعضای رده‌بند به خطای کنونی تنظیم می‌شود.

۳-۶-۱ مجمع وزنی رده‌بندها

نظر به منقضی شدن داده‌های قدیمی، باید به توزیع داده به جای زمان رسیدن داده اعتماد کرد. بر این اساس ونگ [۱۸] و همکارانش [۲۶] یک روش دقت‌ـوزن برای مجمع رده‌بندها [۱۹] (AWE) ارایه کردند تا بتوان رانش مفهوم را در داده‌های جریانی کاوش کرد. الگوریتم با تعداد k رده‌بند که می‌توانند $RIPPER$ یا

[۱۷] Oza
[۱۸] Wang
[۱۹] Accuracy-Weighted Ensemble

C۴٫۵ یا بیزین ساده باشند ایجاد می‌شود و با همان تعداد نیز نگهداری می‌شود. با استفاده از یک تکه از نمونه‌های داده‌ای که در حال آمدن هستند یک رده‌بند جدید آموزش داده می‌شود؛ سپس مجمع با انتخاب k امین رده‌بندهای با دقت سازمان‌دهی می‌شود و وزن هر رده‌بند متناسب با دقت آن تعیین می‌شود. این روش بهتر از یک رده‌بند تنهای داده‌های جریانی مانند VFDT و CVFDT کار می‌کند ولی دقت آن، خیلی به اندازه تکه‌ی انتخاب شده و تعداد رده‌بندها (k) وابسته است.

در کاربردهای واقعی، در داده‌های جریانی ممکن است خطاهایی نظیر این که داده‌ها اشتباه برچسب خورده یا مقدار اشتباه داشته باشند دیده شود. ژنگ [۲۰] و همکارانش [۲۸] الگوریتم مجمع متراکم را برای تعامل با داده‌های دارای خطا ارایه کردند. این رویکرد ترکیبی از چهارچوب‌های افقی و عمودی مجمع‌هاست؛ چهارچوب افقی رده‌بندهای مختلفی بر روی هر چانک داده می‌سازد در حالی که چهارچوب عمودی رده‌بندهای مختلفی روی چانک داده‌ی بروز، با استفاده از الگوریتم‌های مختلف می‌سازد. چهارچوب افقی در برابر نویز مقاوم است و می‌تواند از اطلاعات گذشته استفاده کند ولی وقتی که رانش مفهومی رخ دهد(مفهوم در داده‌های جریانی به طور عمده تغییر کند) مناسب نیست. از طرف دیگر، چهارچوب عمودی می‌تواند وقتی که رانش مفهوم رخ دهد نتیجه خوبی بدهد ولی در برابر نویز حساس است. ساختن رده‌بندها روی چانک‌های مختلف داده و با استفاده از الگوریتم‌های مختلف یادگیری، می‌تواند راه خوبی برای جمع‌آوری یک مجمع از رده‌بندها باشد. این رده‌بندها در یک ماتریس با نام ماتریس رده‌بند [۲۱] جمع‌اوری می‌شوند. روش میانگین‌گیری وزنی روی این ماتریس، برای پیش‌بینی برچسب داده‌های آزمایش استفاده می‌شود. نویسنده به شکل تئوری اثبات کرده است که جمع‌آوری یک مجمع، به طور میانگین مجموع مربعات خطای کمتر یا حداکثر برابر با هر کدام از چهارچوب‌های افقی یا عمودی (به تنهایی) دارد.

نگوین [۲۲] و همکارانش [۲۱] روی مساله یادگیری داده‌های جریانی با ابعاد بالا زمانی که فقط یک زیر مجموعه از ویژگی‌ها برای فرایند یادگیری با اهمیت هستند کار کرده‌اند. در این موضوع، مفهوم ویژگی مرتبط یک مفهوم موقت و محدود به یک دوره زمانی خاص است. ویژگی‌های آموزنده [۲۳] ، ممکن است پس از مدتی نامربوط شوند در حالی که ویژگی‌های به‌درد نخور به ویژگی‌های مهم تبدیل شوند. نویسنده با ارایه مفهوم

[۲۰] Zhang
[۲۱] Classifier Matrix
[۲۲] NGUyen
[۲۳] Informative

رانش ویژگی، به طور مثال تغییر در مجموعه‌ی ویژگی‌های باارزش، و ادغام این مفهوم با وزن‌دهی در مجمع رده‌بند ها، سعی کرده است که این مساله را حل کند. روش انتخاب ویژگی‌های چند متغییره، سازگار با تکنیک پنجره کشویی است و می‌توان به کمک آن رانش‌های ویژگی را تشخیص داد. وقتی یک رانش تدریجی رخ می‌دهد، رده‌بندهای مجمع بروزرسانی می‌شوند و وزن آن‌ها با توجه به نسبت خطایشان تنظیم می‌شود. زمانی که یک رانش ویژگی رخ می‌دهد، مجمع یادگیرنده‌های که قبلا استفاده می‌شده را با رده‌بندهایی که بروز رسانی شده‌اند جایگزین می‌کند. در بروزرسانی، یادگیری با یک مجموعه‌ی جدید از ویژگی‌های باارزش انجام می‌شود. آزمایش‌ها نشان می‌دهد که برای داده‌های جریانی با ابعاد بالا، این روش موثر و کارآمد است.

۳-۶-۲ درخت‌های سازگار تصمیم مبتنی بر رویکرد یکی در برابر بقیه

درخت سازگار تصمیم مبتنی بر رویکرد یکی در برابر بقیه [۲۴] یک روش مجمع جدید برای داده‌های جریانی است[۱۱]. در این روش، مجمع k رده‌بند باینری CVFDT را یادمی‌گیرد. هر رده‌بند آموزش داده شده است که تشخیص دهد یک نمونه به یک کلاس خاص متعلق است یا به باقی کلاس‌ها. برای رده‌بندی یک داده‌ی جدید، همه رده‌بندها اجرا می‌شوند و رده‌بندی که مطمن‌ترین نتیجه را داد، به عنوان برچسب نمونه‌ی جدید انتخاب می‌شود. اطمینان رده‌بندی متناسب با کلاس غالب در برگ‌ها زمانی که نمونه تست وارد می‌شود تنظیم می‌شود. برای دستیابی به بیش‌ترین دقت‌ها، OVA تلاش می‌کند یک مجمعی از رده‌بندهای CVFDT بسازد که کمترین همبستگی خطا و بیشترین تنوع را داشته باشند. این روش به سهولت و سریع با رانش مفهوم سازگار می‌شود و فقط کافیست دو بخش رده‌بندها که مرتبط با کلاسی که تغییر کرده است، بروز رسانی شوند؛ به علاوه این روش به خوبی برای داده‌های جریانی نامتوازن کار می‌کند.

۳-۶-۳ مجمع متا_دانش

بیشترین حد ممکن پیچیدگی برای رده‌بندهای یادگیرنده، محدود به محدویت‌های زمانی جریان‌های داده‌ای در کاربردهای دنیای واقعی است. ژنگ [۲۵] و همکارانش، یک روش مجمع متا_دانش [۲۶] که بهترین و مناسب‌ترین رده‌بند ها را برای رده‌بندی داده‌های تست انتخاب می‌کند را ارایه دادند[۲۷]. یک مجمع درخت

[۲۴] Adapted One-vs-All Decision Trees (OVA)

[۲۵] Zhang

[۲۶] Meta-knowledge Ensemble

برای سازماندهی رده‌های پایه ساخته شده است و هر کدام یک وزن و یک محدوده از کل داده‌ها اختصاص داده می‌شود. مشابه RTree ، درخت مجمع‌ها سه عملگر پایه، شامل جستجو، اضافه کردن و حذف کردن دارد. ارتفاع این درخت متوازن است و پیچیدگی زمانی لگاریتمی را برای پیش‌بینی تضمین می‌کند. از عملگر افزودن برای ادغام یک رده‌بند جدید به مجمع استفاده می‌شود. زمانی که تعداد رده‌بندها در یک گره، از یک حد از پیش تعیین شده تجاوز کند، گره با این ایده که محدوده تحت پوشش هر گره کمینه شود، به دو گره تقسیم می‌شود. زمانی که ظرفیت E-tree پر شده باشد، عملگر حذف رده‌بندهای منقضی شده را حذف می‌کند و درخت ممکن است که دوباره سازماندهی شود تا متعادل بودن تضمین شود. عملگر جستجو نیز برای رده‌بندی یک نمونه مانند x استفاده می‌شود. رده‌بندهای فضای نزدیک شامل x فراخوانی می‌شوند و یک روش رای‌گیری وزنی برای تصمیم‌گیری برچسب نمونه‌ی x اجرا می‌شود.

فصل ۴

جمع‌بندی و نتیجه‌گیری

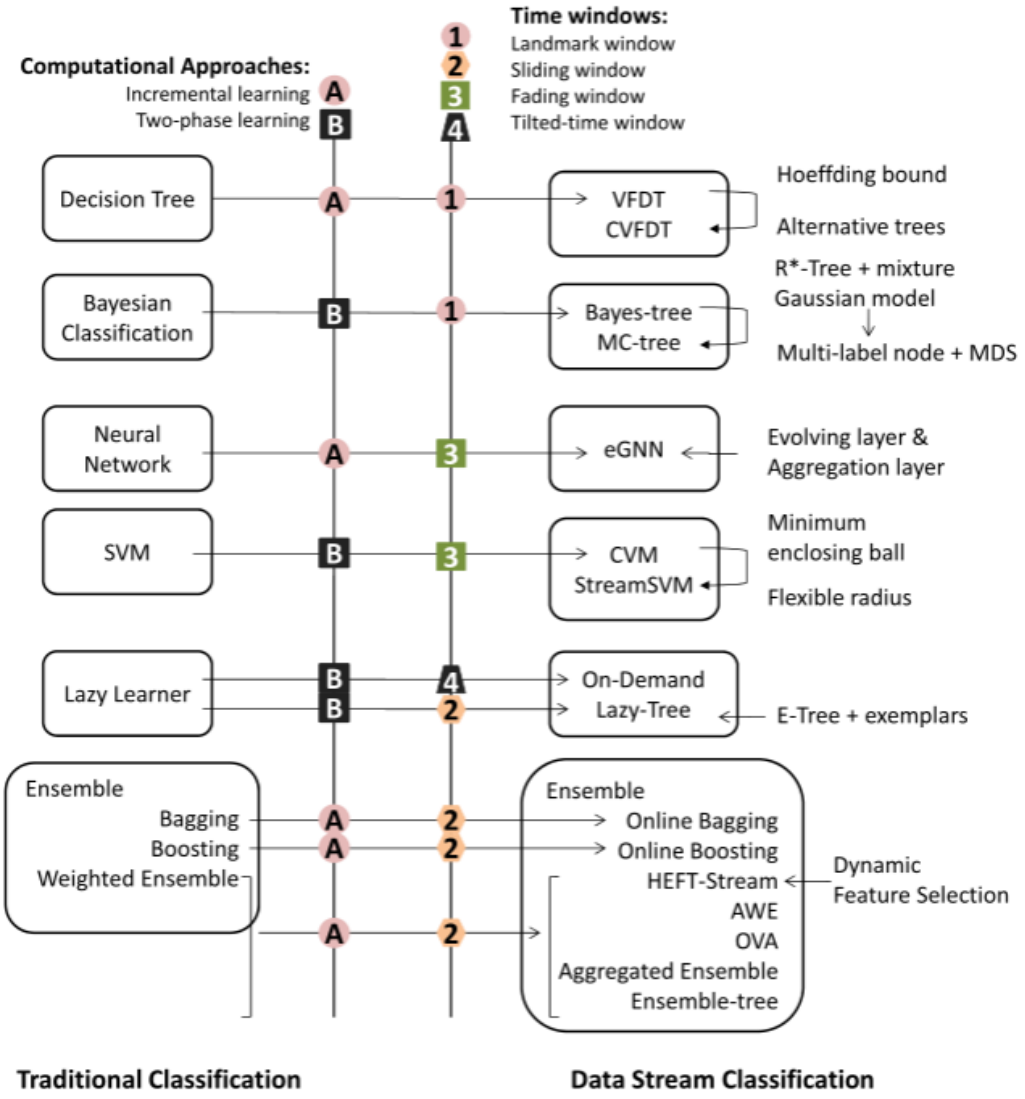
در فصل‌های گذشته به بررسی روش‌های یادگیری ماشین در فضای جریان‌های داده‌ای پرداختیم. در این فصل روش‌های معرفی شده را جمع‌بندی خواهیم کرد و مزایا و معایب هر روش را بیان می‌کنیم. به علاوه نسبت این الگوریتم‌ها را با الگوریتم‌های سنتی یادگیری ماشین بیان خواهیم کرد. سپس با توجه به ذات داده‌های متنی، مشخص خواهیم کرد که کدام یک از این روش‌ها می‌توانند در مواجهه با داده‌های متنی جریانی مورد استفاده قرار گیرند و در نهایت کارهای آینده را بیان می‌کنیم.

۴-۱ مروری بر روش‌های یادگیری

پس از معرفی که در فصل پیش از رده‌بندهایی که برای داده‌های جریانی وجود دارند، شد به جمع‌بندی این موضوع و تمام مسایل مرتبط خواهیم پرداخت.

تصویر ۴-۱ [۲۰] ارتباط بین روش‌های سنی رده‌بندی و رده‌بندی در داده‌های جریانی را نشان می‌دهد. رده‌بندهای سنتی در سمت چپ نوشته شده‌اند و رده‌بندهای داده‌های جریانی در سمت راست. برای دسته‌بندی رده‌بندهای داده‌های جریانی به دو الگو، برای رویکردهای محساباتی و پنجره‌های زمانی نیز در وسط تصویر است.

می‌توان دید که رده‌بندهای داده‌های جریانی از رده‌بندهای سنتی برگرفته شده‌اند، با این تفاوت که از رویکردهای محاسباتی مختلف و پنجره‌های زمانی متفاوتی استفاده می‌کنند. به عنوان نمونه VFDT یک گسترش از



شکل ۱-۴: مقایسه و نمایش ارتباط روش‌های سنتی یادگیری ماشین با روش‌های یادگیری در داده‌های جریانی

درخت‌های تصمیم برای داده‌های جریانی است. زمانی که به حد کافی داده موجود باشد، VFDT از حد هافدین برای ساختن گره‌های درخت استفاده می‌کند. در واقع این دنباله‌ای از رویکرد یادگیری افزایشی و پنجره نقطه عطفی است. CVFDT یک نسخه بهبودیافته از VFDT است که می‌تواند به وسیله‌ی ساختن درخت‌های جایگزین با رانش مفهوم سازگار شود. درخت بیز یک نسخه تغییر یافته از رده‌بندهای بیزین با دو فاز مختلف یادگیری و پنجره نقطه‌عطفی است. MCTree نیز بهبودی از درخت بیز با استفاده از گره‌های چند برچسبه است. eGNN یک شبکه عصبی است که برای کار با داده‌های جریانی طراحی شده‌است. CVM نیز نشات گرفته از رده‌بند SVM است که از رویکرد دوفازی یادگیری و پنجره محو شونده استفاده می‌کند. StreamSVM نیز یک نسخه توسعه‌یافته CVM است که با ساختن حداقل گوی نزدیک به شکل پویا عمل می‌کند.

الگوریتم On-Demand ، یک نسخه بهبود یافته از رده‌بندهای k-NN است که با استفاده از رویکرد یادگیری دوگامی و پنجره یک بر کار می‌کند. Lazy-Tree نیز یک نسخه بهبودیافته از رده‌بند k-NN است که با استفاده از رویکرد یادگیری دوگامی و و پنجره کشویی کار می‌کند. تعداد زیادی مجمع رده‌بندها برای کار با داده‌های جریانی طراحی شده است. Bagging و Boosting آنلاین، نسخه‌های از روش‌های سنتی Bagging و Boosting با استفاده از رویکرد یادگیری افزایشی و پنجره کشویی است.

تعداد زیادی مجمع وزنی از رده‌بندها وجود دارد که از استراتژی‌های مختلف وزن‌دهی استفاده می‌کنند. برای نمونه مجمع‌های جمع‌آوری شده که از روش میانگین وزنی(رای‌گیری) استفاده می‌کنند مانند HEFT-Stream ، AWE ، OVA ، Ensemble-tree که در این روش‌ها وزن متناسب با دقت رده‌بند تنظیم می‌شود. این روش‌های دنباله‌ای از رویکرد یادگیری افزایشی هستند و می‌توان فرض کرد که از پنجره کشویی استفاده می‌کنند و عموما اعضای رده‌بندی که دقت کمی دارند را از مجمع حذف می‌کند.

به طور خلاصه، در جدول ۴-۱ ظرفیت‌های رده‌بند های مختلف روی داده‌های جریانی، به شکل مقایسه‌ای آورده شده است. همانطور که مشهود است بسیاری از رده‌بندهای مطرح شده، عموما نمی‌توانند در آن واحد در تمام محدودیت‌های کار با داده‌های جریانی بگنجند. در این جدول در دو سطح تعامل با رانش وجود دارد: این که بتوانند با رانش مفهوم سازگار شوند و این که رانش مفهوم را رده‌بندی کنند. به علاوه مشخص شده است که کدام یک از رده‌بندها می‌توانند با داده‌های با ابعاد بالا در جریان‌های داده‌ای کار می‌کند.

همانطور که دور از انتظار نیست، تمام روش‌های رده‌بندی داده‌های جریانی، یکبار داده را مشاهده می‌کنند و با محدودیت حافظه مشکل چندانی ندارند. eGNN و CVM و StreamSVM نمی‌توانند بلادرنگ

جدول ۴-۱: [۲۰] مقایسه ظرفیت‌های الگوریتم‌های مختلف یادگیری در داده‌های جریانی

Algorithm	Bounded memory	Single-pass	Real-time response	Concept-drift adaptation	Concept-drift classification	High-dimensional data
Aggregated ensemble	√	√	√	√		
AWE	√	√	√	√		√
Bayes tree	√	√	√	√		
CVFDT	√	√	√	√		√
CVM	√	√		√		
eGNN	√	√		√		
Ensemble-tree	√	√	√	√		√
HEFT-stream	√	√	√	√	√	√
Lazy-tree	√	√	√	√		
MC-tree	√	√	√	√		
On-Demand	√	√	√	√		
Online bagging & boosting	√	√	√			
OVA	√	√	√	√		√
StreamSVM	√	√		√		
VFDT	√	√	√			√

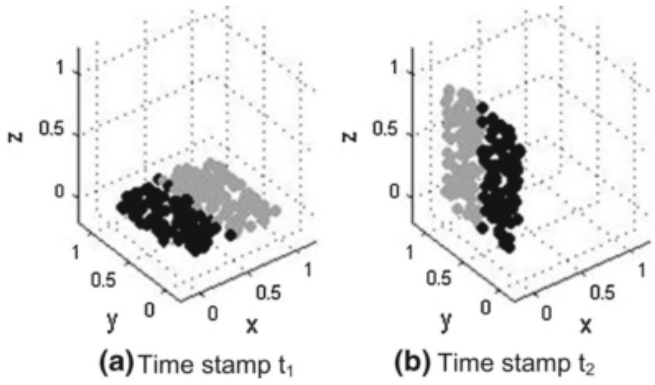
جدول ۲-۴: مزایا و محدودیت‌های روش‌های رده‌بندی برای داده‌های جریانی

روش	مزیت	محدودیت
درخت تصمیم	فهم آسان مقاوم در برابر خطا هزینه محاسباتی کم حتی با داده آموزش زیاد توانایی حذف ویژگی های تکراری	بیش برازش هنگامی که عمق درخت ها زیاد باشد
بیزین	ساده، کارا، بهینه و مقاوم در برابر خطا	پیش فرض استقلال که در دنیای واقعی معمولا نقض می شود
شبکه عصبی	به راحتی عمومی می شود می تواند مسایل پویا و غیر خطی را حل کند	عدم توانایی تفسیر مدل یادگرفته شده پیچیدگی بالا
بردار پشتبان	ارایه یک راه حل یگانه توانایی حل مسایل غیر خطی با گرفتن کرنل های متفاوت	به انتخاب کرنل وابسته است. پیچیدگی بالاست
	وقتی داده ها به یک سمت متمایل هستند هم خوب کار می کند.	برای مسایل چند کلاسه طراحی سخت است
یادگیرنده Lazy	پیاده سازی راحت	فضای ذخیره سازی زیاد
نزدیک ترین همسایگی		پیچیدگی بالا عملکرد بسیار بد برای داده های با ابعاد بالا
مجمع	صحت بالا پیاده سازی ساده	بر پایه اکتشافات و نداشتن تئوری دقیق

پاسخگو باشند و برای دادن نتیجه به زمان جهت پردازش‌هایشان احتیاج دارند. HEFT-Stream می‌تواند دو نوع رانش مفهوم(هم رانش تدریجی و هم رانش ویژگی) را مشخص کند و به خوبی با آن‌ها سازگار شود. اگر درخت تصمیم به عنوان رده‌بند پایه برای روش‌های VFDT و CVFDT و AWE و OVA و درخت مجمع انتخاب شود، این روش‌ها می‌توانند با داده‌هایی با ابعاد بالا نیز کار کنند کارکنند.

همانطور که دور از انتظار نیست، تمام روش‌های رده‌بندی داده‌های جریانی، یکبار داده را مشاهده می‌کنند و با محدودیت حافظه مشکل چندانی ندارند. eGNN و CVM و StreamSVM نمی‌توانند بلادرنگ پاسخگو باشند و برای دادن نتیجه به زمان جهت پردازش‌هایشان احتیاج دارند. HEFT-Stream می‌تواند دو نوع رانش مفهوم(هم رانش تدریجی و هم رانش ویژگی) را مشخص کند و به خوبی با آن‌ها سازگار شود. اگر درخت تصمیم به عنوان رده‌بند پایه برای روش‌های VFDT و CVFDT و AWE و OVA و درخت مجمع انتخاب شود، این روش‌ها می‌توانند با داده‌هایی با ابعاد بالا نیز کار کنند کارکنند.

هنگامی که یک رده‌بند از رویکر محاسباتی و پنجره زمانی استفاده می‌کند بعضی مزایا و معایب که در فصل‌های قبل نیز در مورد آن‌ها بحث شد، خودشان را نشان می‌دهد. به هر روی روش‌های سنتی و رده‌بندهای داده‌های جریانی هر دو مزایا و محدودیت‌هایی دارند که در جدول ۲-۴ آمده است. به عنوان مثال درخت



شکل ۲-۴: یک مثال که نشان می‌دهد چگونه ذات ویژگی‌های بااهمیت طی زمان می‌تواند تغییر کند.

تصمیم به راحتی قابل فهم است، در برابر خطا مقاوم است، کاراست و می‌تواند خصیصه‌های تکراری را حذف کند. اما اگر عمق درخت‌ها زیاد باشد دچار مشکل بیش‌برازش [۱] می‌شود. یادگیرنده Lazy می‌تواند به سادگی پیاده‌سازی شود اما این یادگیرنده حافظه زیادی مصرف می‌کند و در برابر داده‌هایی با ابعاد بالا مشکل دارد. مجمع‌ها صحت بالایی دارند و به سادگی پیاده‌سازی می‌شوند اما بیشترشان بر پایه اکتشافات هستند و پایه تئوری قوی ندارند.

۴-۲ کارهای آینده در جریان‌های متنی

از آن‌جایی که بیشتر تحقیقات در داده‌های جریانی مربوط به دهه‌ی اخیر بوده است، هنوز مسایل زیادی برای تحقیقات وجود دارد[۲۰]. با توجه موضوع این گزارش، در این بخش، تحقیقات آینده‌ای که مربوط به کاوش و یادگیری درباره داده‌های جریانی متنی است را مطرح می‌کنیم.

۴-۲-۱ انتخاب پویای ویژگی

در همه‌ی داده‌های با ابعاد بالا، که داده‌های متنی هم جز آن‌هاست تمام ویژگی‌ها(خصیصه‌ها) در فرایند با اهمیت هستند. در واقع سه نوع مختلف ویژگی وجود داد: (یک) ویژگی‌های غیر مرتبط، (دو) ویژگی‌های مرتبط اما تکراری و (سه) ویژگی‌های مرتبط و غیر تکراری. وظیفه‌ی اصلی موضوع انتخاب ویژگی، استخراج

[۱] Overfitting

مجموعه‌ی ویژگی‌های مرتبط و غیرتکراری از ویژگی‌هاست که فرایند یادگیری را با معنی‌تر و سریع‌تر کند. در ادبیات موضوع، روش‌های انتخاب ویژگی به سه دسته تقسیم می‌شوند: فیلتر، پوشش [۲] و مدل‌های نهفته [۳] [۱۸].

مدل فیلتر به عنوان یک معیار مستقل برای ارزیابی مجموعه‌ی ویژگی‌ها به کار برده می‌شود، بنابراین این روش فقط به مشخصه‌های عمومی داده اتکا می‌کند. مدل پوششی، به همراه الگوریتم‌های یادگیری اجرا می‌شود از کارایی الگوریتم‌های یادگیری برای ارزیابی ویژگی‌ها استفاده می‌کند. مدل‌های ترکیبی از مزیت دو مدل گفته شده استفاده می‌کند.

اهمیت ویژگی‌ها در داده‌های جریانی تغییر می‌کند و محدود بودن به یک بازه‌ی زمان مشخص است. ویژگی‌های که پیش‌از این باارزش فرض می‌شد ممکن است نامربوط شوند و برعکس ممکن است ویژگی‌های استفاده نشده در آینده با اهمیت شوند. بنابراین استفاده از روش‌های پویای انتخاب ویژگی برای رصدکردن تغییرات ویژگی‌ها ضروری است. شکل ۴-۲ [۲۰] ذات پویای ویژگی‌های کلیدی را نشان می‌دهد. فرض کنید داده‌های ما سه ویژگی و دو کلاس داشته باشند: نقطه‌های سیاه و قهوه‌ای نمایانگر کلاس‌های مثبت و منفی هستند. در زمان t_1، ویژگی‌های با ارزش x و y هستند چون داده روی صفحه xy جا گرفته است. پس از این که توزیع داده فرق می‌کند، ویژگی‌های با اهمیت y و z می‌شوند.

پس از بررسی تعداد زیادی از الگوریتم‌های یادگیری، مشاهده شد که تنها اندکی از آن‌ها می‌توانند با داده‌هایی با ابعاد بالا کار کنند و محدودیت‌های زیادی دارند. بعضی از رده‌بندهایی که با داده‌های با ابعاد بالا کار می‌کنند. این الگوریتم‌ها معمولا از درخت تصمیم به عنوان رده‌بند پایه استفاده می‌کنند و توانایی حذف ویژگی‌های تکراری را ندارند. HEFT-Stream تنها مجمع رده‌بندی است که هم ویژگی‌های نامرتبط و هم ویژگی‌های تکرای را حذف می‌کند و با هر نوع رده‌بندی کار می‌کند. HEFT-Stream از مدل فیلتر استفاده می‌کند و مستقل از نوع اعضای رده‌بند است. به هر حال مساله انتخاب پویای ویژگی یک مساله باز است که نیاز به تحقیقات بیشتری دارد[۲۰].

[۲] Wrapper

[۳] Embedded models

۴-۲-۲ کاوش در جریان‌های متنی

در سال‌های اخیر تعداد زیادی از نرم‌افزارهای مبتنی بر وب، حجم زیادی از جریان‌های متنی را تولید می‌کنند. به عنوان مثال، اعضای شبکه‌های اجتماعی به طور مداوم با پیام‌های متنی با یکدگیر تعامل می‌کنند. بسیاری از پرتال‌ها اخبار را بر اساس علاقه‌مندی خوانندگان، بلادرنگ به آن‌ها نشان می‌دهند و خزنده‌های وب میلیون‌ها صفحه را برای نمایه سازی ذخیره می‌کنند. کاوش در جریان‌های متنی، که با سایر وظیفه‌هایی مانند فیلترکردن هرزنامه‌ها مرتبط است.

به طور عمومی، روش‌های داده‌های جریانی با ابعاد بالا می‌توانند برای داده‌های متنی استفاده شوند، البته این موضوع باید بعد از جمع‌آوری و عملیات پیش‌پردازش شامل حذف کلمات ایست، ریشه‌یابی، نگاشت به نمایش‌هایی از قبیل کیسه‌ای از کلمات [۴] ، TF-IDF و جداسازی عبارت‌ها باشد. اما داده‌های متنی پیچیده‌تر از داده‌های با ابعاد بالا هستند، چرا که معمولا این داده‌ها غیرساخت‌یافته، شامل خطاهای سطح بالا و در فرمت‌های گوناگون هستند. به علاوه در داده‌های متنی بیشتر تغییر موضوع در زمان رخ می‌دهد. همین باعث شده است که کاوش در جریان‌های متنی یک کار دلهره‌آور باشد[۲۰].

[۴] Bag of Word

مراجع

[1] Aggarwal, C. C. Data streams: An overview and scientific applications. in *Scientific Data Mining and Knowledge Discovery*. Springer, 2009, pp. .397–377

[2] Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. A framework for clustering evolving data streams. in *Proceedings of the 29th international conference on Very large data bases-Volume 29* (2003), VLDB Endowment, pp. .92–81

[3] Aggarwal, C. C., Han, J., Wang, J., and Yu, P. S. A framework for on-demand classification of evolving data streams. *IEEE Transactions on Knowledge and Data Engineering 18*, 5 (2006), .589–577

[4] Aggarwal, C. C., and Zhai, C. *Mining text data*. Springer Science & Business Media, .2012

[5] Bifet, A. Adaptive stream mining: Pattern learning and mining from evolving data streams. in *Proceedings of the 2010 conference on adaptive stream mining: Pattern learning and mining from evolving data streams* (2010), Ios Press, pp. .212–1

[6] Bifet, A., Holmes, G., Kirkby, R., and Pfahringer, B. Moa: Massive online analysis. *Journal of Machine Learning Research 11*, May (2010), .1604–1601

[7] Bouguelia, M.-R., Belaïd, Y., and Belaïd, A. An adaptive streaming active learning strategy based on instance weighting. *Pattern Recognition Letters 70* (2016), .44–38

[8] Domingos, P., and Hulten, G. Mining high-speed data streams. in *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* (2000), ACM, pp. .80–71

[9] Gama, J., and Gaber, M. M. *Learning from data streams*. Springer, .2007

[10] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. The weka data mining software: an update. *ACM SIGKDD explorations newsletter 11*, 1 (2009), .18–10

[11] Hashemi, S., Yang, Y., Mirzamomen, Z., and Kangavari, M. Adapted one-versus-all decision trees for data stream classification. *IEEE Transactions on Knowledge and Data Engineering 21*, 5 (2009), .637–624

[12] Hofmann, M., and Klinkenberg, R. *RapidMiner: Data mining use cases and business analytics applications*. CRC Press, .2013

[13] Hulten, G., Spencer, L., and Domingos, P. Mining time-changing data streams. in *Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining* (2001), ACM, pp. .106–97

[14] Kelly, M. G., Hand, D. J., and Adams, N. M. The impact of changing populations on classifier performance. in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining* (1999), ACM, pp. .371–367

[15] Leite, D., Costa, P., and Gomide, F. Evolving granular neural network for semi-supervised data stream classification. in *The 2010 international joint conference on neural networks (IJCNN)* (2010), IEEE, pp. .8–1

[16] Leite, D. F., Costa, P., and Gomide, F. Evolving granular classification neural networks. in *2009 International Joint Conference on Neural Networks* (2009), IEEE, pp. .1743–1736

[17] Leskovec, J., Rajaraman, A., and Ullman, J. D. *Mining of massive datasets*. Cambridge University Press, .2014

[18] Liu, H., and Yu, L. Toward integrating feature selection algorithms for classification and clustering. *IEEE Transactions on knowledge and data engineering 17*, 4 (2005), .502–491

[19] Narasimhamurthy, A. M., and Kuncheva, L. I. A framework for generating data to simulate changing environments. in *Artificial Intelligence and Applications* (2007), pp. .420–415

[20] Nguyen, H.-L., Woon, Y.-K., and Ng, W.-K. A survey on data stream clustering and classification. *Knowl Inf Syst 45* (2015), .569–535

[21] Nguyen, H.-L., Woon, Y.-K., Ng, W.-K., and Wan, L. Heterogeneous ensemble for feature drifts in data streams. in *Pacific-Asia Conference on Knowledge Discovery and Data Mining* (2012), Springer, pp. .12–1

[22] Oza, N. C. Online bagging and boosting. in *2005 IEEE international conference on systems, man and cybernetics* (2005), volume 3, IEEE, pp. .2345–2340

[23] Rai, P., Daumé III, H., and Venkatasubramanian, S. Streamed learning: one-pass svms. *arXiv preprint arXiv:09080572*. (2009).

[24] Seidl, T., Assent, I., Kranen, P., Krieger, R., and Herrmann, J. Indexing density models for incremental learning and anytime classification on data streams. in *Proceedings of the 12th international conference on extending database technology: advances in database technology* (2009), ACM, pp. .322–311

[25] Tsang, I. W., Kocsor, A., and Kwok, J. T. Simpler core vector machines with enclosing balls. in *Proceedings of the 24th international conference on Machine learning* (2007), ACM, pp. .918–911

[26] Wang, H., Fan, W., Yu, P. S., and Han, J. Mining concept-drifting data streams using ensemble classifiers. in *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (2003), ACM, pp. .235–226

[27] Zhang, P., Gao, B. J., Zhu, X., and Guo, L. Enabling fast lazy learning for data streams. in *2011 IEEE 11th International Conference on Data Mining* (2011), IEEE, pp. .941–932

[28] Zhang, P., Zhu, X., Shi, Y., Guo, L., and Wu, X. Robust ensemble learning for mining noisy data streams. *Decision Support Systems 50*, 2 (2011), .479–469