مقاله پژوهشی

ارائه یک الگوریتم مناسب برای یادگیری جریانی بر اساس الگوریتم ماشین‌های بردار پشتیبان دوقلوی مربعات حداقلی فازی

جواد سلیمی سرتختی و سلمان گلی بیدگلی

چکیده: الگوریتم ماشین بردار پشتیبان یکی از الگوریتم‌های مشهور و با کارایی بالا در یادگیری ماشین و کاربردهای مختلف است. از این الگوریتم تا کنون نسخه‌های متعددی ارائه شده که آخرین نسخه آن ماشین‌های بردار پشتیبان دوقلوی مربعات حداقلی فازی می‌باشد. اغلب کاربردها در دنیای امروز دارای حجم انبوهی از اطلاعات هستند. از سویی دیگر یکی از جنبه‌های مهم داده‌های حجیم، جریانی‌بودن آنها می‌باشد که باعث شده است بسیاری از الگوریتم‌های سنتی، کارایی لازم را در مواجهه با آن نداشته باشند. در این مقاله برای نخستین بار نسخه افزایشی الگوریتم ماشین‌های بردار پشتیبان دوقلوی مربعات حداقلی فازی، در دو حالت برخط و شبه برخط ارائه شده است. برای بررسی صحت و دقت الگوریتم ارائه‌شده دو کاربرد آن مورد ارزیابی قرار گرفته است. در یک کاربرد، این الگوریتم بر روی ٦ دیتاست مخزن UCI اجرا شده که در مقایسه با سایر الگوریتم‌ها از کارایی بالاتری برخوردار است. حتی این کارایی در مقایسه با نسخه‌های غیر افزایشی نیز کاملاً قابل تشخیص است که در آزمایش‌ها به آن پرداخته شده است. در کاربرد دوم، این الگوریتم در مبحث اینترنت اشیا و به طور خاص در داده‌های مربوط به فعالیت روزانه به کار گرفته شده است. طبق نتایج آزمایشگاهی، الگوریتم ارائه‌شده بهترین کارایی را در مقایسه با سایر الگوریتم‌های افزایشی دارد.

کلیدواژه: **یادگیری جریانی، ماشین‌های بردار پشتیبان، دسته‌بندی فازی، FLSTSVM.**

۱- مقدمه

امروزه حجم عظیم داده‌های تولیدی و نوع آنها باعث شده که الگوریتم‌های یادگیری ماشین سنتی کارایی چندانی نداشته باشند. بسیاری از این الگوریتم‌ها در مواجهه با کلان داده‌ها[1] کارایی خوبی ندارند. علاوه بر حجم داده‌ها، نوع داده‌ها هم می‌تواند باعث عدم کارایی الگوریتم‌های سنتی شود. داده‌های جریانی[2] یکی از انواع داده‌هایی است که با ظهور و بروزش نیاز به تغییرات اساسی در الگوریتم‌های یادگیری ماشین به وجود آورده است. داده‌های جریانی داده‌هایی هستند که در طول زمان تولید می‌شوند و در هر لحظه به همه داده‌ها دسترسی نداریم [۱]. علاوه بر آن در داده‌های جریانی عموماً تغییر توزیع داده‌ها وجود دارد. یعنی داده‌ها تا یک زمان خاص از یک توزیع مشخص تبعیت می‌کنند اما به تدریج ممکن است توزیعشان تغییر کند. این ویژگی باعث می‌شود که الگوریتم‌های سنتی به راحتی قادر به مواجهه با این نوع داده‌ها نباشند.

داده‌های جریانی عموماً در حجم و سرعتی بالا تولید می‌شوند. به عنوان مثالی از داده‌های جریانی می‌توان به داده‌هایی که در کاربردهای اینترنت اشیا، لاگ‌های امنیتی و لاگ‌های سرورها، تبلیغات بلادرنگ، کلیک‌های اپلیکیشن‌ها و وب‌سایت‌ها تولید می‌شوند اشاره کرد. در همه این مثال‌ها ابزارهایی وجود دارند که به طور پیوسته داده‌هایی با ساختار و یا بدون ساختار را تولید می‌کنند. بنابراین به دلیل عدم وجود داده‌ها به صورت یک‌جا و نیز تغییر توزیع آنها در طول زمان، الگوریتم‌های سنتی کارایی بالایی در این نوع داده‌ها ندارند. بیشتر راهکارهای ارائه‌شده برای مواجهه با این نوع داده‌ها بر پایه همان الگوریتم‌های سنتی استوار هستند و محققین با تغییراتی در نسخه‌های اصلی، در کاربردهای داده‌های جریانی استفاده نموده‌اند. این راهکارها بدین گونه عمل می‌کنند که ابتدا الگوریتم یادگیری بر اساس یک مجموعه داده در دسترس آموزش می‌بینند و سپس داده‌های جدیدی را که وارد می‌شوند بر اساس فاز یادگیری دسته‌بندی می‌کند. اما تغییر توزیع ممکن است این راهکارها را به شکست بکشاند. از سویی دیگر ممکن است برای مقابله با این نقطه ضعف به ازای هر نمونه جدید تصمیم بگیریم مدل آموزش دیده شده را بر روی تمامی داده‌هایی که تا کنون دریافت کرده‌ایم مجدداً آموزش دهیم. این روش اگرچه تغییرات توزیع داده‌ها را یاد می‌گیرد اما به شدت کند است و عملاً در کاربردهای برخط استفاده از آن غیر ممکن می‌باشد. در راهکارهای دیگر ابتدا اهمیت هر نمونه جدید ورودی بر اساس یک سری از معیارها سنجیده می‌شود و سپس بر اساس آن تصمیم گرفته می‌شود که آیا نیاز به آموزش دوباره مدل آموزش‌دیده وجود دارد یا خیر. در صورت نیاز، فاز آموزش الگوریتم بر روی تمامی داده‌هایی که تا کنون دریافت شده است مجدداً اجرا می‌شود.

یکی از الگوریتم‌هایی که در کاربردهای سنتی کارایی قابل قبولی دارد الگوریتم ماشین‌های بردار پشتیبان (SVM)[3] است [۲]. الگوریتم ماشین بردار پشتیبان یکی از الگوریتم‌های معروف در زمینه یادگیری با ناظر است که برای دسته‌بندی و رگرسیون استفاده می‌شود. یکی از خصوصیات مهم ماشین بردار پشتیبان این است که به طور هم‌زمان خطای تجربی دسته‌بندی را کمینه نموده و حاشیه‌های هندسی را از نمونه‌های دو کلاس بیشینه می‌کند. تا کنون چندین نسخه از ماشین‌های بردار پشتیبان ارائه شده که از آن جمله می‌توان به ماشین‌های بردار پشتیبان دوقلو، مقدار

این مقاله در تاریخ ۱۰ اردیبهشت ماه ۱۳۹۹ دریافت و در تاریخ ۷ تیر ماه ۱۴۰۰ بازنگری شد.

جواد سلیمی سرتختی (نویسنده مسئول)، گروه مهندسی کامپیوتر، دانشکده برق و کامپیوتر، دانشگاه کاشان، ایران، (email: salimi@kashanu.ac.ir).
سلمان گلی بیدگلی، گروه مهندسی کامپیوتر، دانشکده برق و کامپیوتر، دانشگاه کاشان، ایران، (email: salmangoli@kashanu.ac.ir).

1. Big Data
2. Stream Data
3. Support Vector Machine

ویژه و مربعات حداقل و نیز ماشین بردار پشتیبان دوقلوی مربعات حداقلی فازی (FLSTSVM) اشاره کرد. در سال‌های اخیر کاربرد مفهوم فازی در این دسته از الگوریتم‌ها به شدت توسعه یافته و بنابراین ماشین بردار پشتیبان دوقلوی مربعات حداقلی فازی کاربردهای فروانی پیدا کرده است.

از سوی دیگر با توجه به آنچه که به آن اشاره شد، این الگوریتم در مواجهه با داده‌های حجیم و داده‌های جریانی کارایی بالایی ندارد. در این مقاله برای نخستین بار الگوریتمی افزایشی بر اساس الگوریتم FLSTSVM ارائه می‌شود که دارای کارایی به مراتب بهتری از سایر نسخه‌های SVM در مواجهه با داده‌های جریانی می‌باشد. برای اثبات کارایی الگوریتم ارائه‌شده که آن را Incremental FLSTSVM می‌نامیم از ۶ مجموعه داده متعلق به مخزن UCI استفاده شده که در همه آنها الگوریتم ارائه‌شده دقت و سرعت بیشتری دارد.

ساختار ادامه مقاله بدین شرح است: در بخش دوم به مفاهیم پایه در مورد نسخه‌های مختلف SVM و تفاوت آنها با یکدیگر می‌پردازیم و کارهای پیشین انجام‌شده در زمینه الگوریتم افزایشی ماشین‌های بردار پشتیبان مورد بررسی قرار می‌گیرد. در بخش سوم الگوریتم پیشنهادی به طور مفصل ارائه شده و در مورد پیچیدگی متوسط الگوریتم ارائه‌گردیده بحث خواهد شد. در بخش چهارم نتایج پیاده‌سازی حاصل از اجرای الگوریتم پیشنهادی بر روی مجموعه داده‌های گوناگون ارائه می‌شود و در بخش نهایی جمع‌بندی و کارهای آینده بیان خواهد گردید.

۲- یادگیری ماشین و الگوریتم‌های ژنتیک موازی

در این بخش مروری بر انواع نسخه‌های SVM خواهیم داشت و توانمندی هر یک از آنها را بیان می‌کنیم. در بخش بعدی نیز به انواع رویکردهایی که برای تبدیل این الگوریتم‌ها به الگوریتم‌های جریانی و یا افزایشی (الگوریتم‌هایی که فرایند یادگیری آنها به تدریج و با توجه به دریافت داده‌های جریانی تکمیل می‌شود) ارائه شده است می‌پردازیم.

۲-۱ الگوریتم استاندارد ماشین بردار پشتیبان

ماشین بردار پشتیبان یک دسته‌بند متعلق به روش‌های مبتنی بر کرنل است که برای اولین بار در سال ۱۹۹۲ توسط وپنیک در موسسه AT&T ارائه شد [۲]. شهرت ماشین بردار پشتیبان به دلیل موفقیت آن در تشخیص حروف دست‌نویس است که با شبکه‌های عصبی که به دقت برای این کار تنظیم شده بودند برابری می‌کرد. ایده اصلی در ماشین بردار پشتیبان، بیشینه‌سازی حاشیه بین دو کلاس می‌باشد. اگر داده‌های آموزشی به صورت $x_i \in R^d$، $y_i \in \{-1,+1\}$ و $\{x_i, y_i\}, i = 1,\ldots,l$ برچسب‌گذاری شده باشند ماشین بردار پشتیبان به دنبال ابرصفحه‌ای با معادله $W.X + b = 0$ است که توانایی ارضای محدودیت‌های بیان‌شده در (۱) را داشته باشد

$$y_i(w.x_i + b) \geq 1 \ , \ \forall i \qquad (1)$$

چنین ابرصفحه‌ای با حل (۲) به دست خواهد آمد

$$\text{Minimize } f(x) = \frac{\|w\|^2}{2} \qquad (2)$$
$$\text{subject to : } y_i(w.x_i + b) - 1 \geq 0$$

برای حل (۲) باید از روش مضرب‌های لاگرانژ استفاده کرد [۲] که برای این کار دو دلیل وجود دارد. اول این که محدودیت‌های نشان داده شده در (۲) با خود ضرایب لاگرانژ جایگزین خواهند شد که این مسأله کار ما را بسیار راحت‌تر خواهد نمود. دوم این که در این گونه فرموله‌نمودن مسأله، داده‌های آموزشی فقط به صورت ضرب نقطه‌ای بردارها پدیدار می‌شوند [۳]. این یک ویژگی حیاتی است که به ما این اجازه را می‌دهد که فرایند حل مسأله را به حالت غیر خطی تعمیم دهیم.

۲-۲ الگوریتم ماشین بردار پشتیبان مربعات حداقلی

یکی از معایب الگوریتم ماشین بردار پشتیبان، زمان یادگیری نسبتاً طولانی آن می‌باشد. دلیل این زمان زیاد هم به دست آوردن مضرب‌های لاگرانژ با استفاده از برنامه‌نویسی غیر خطی است که راه حل‌های آن دارای چندین تکرار برای به دست آوردن این ضرایب می‌باشند. از طرفی اگر بتوان محدودیت‌های نابرابری را به محدودیت‌های برابری تبدیل کرد، آن گاه تنها با یک بار اجرا می‌توان مضرب‌های لاگرانژ را به دست آورد.

ماشین‌های بردار پشتیبان مربعات حداقل[1] [۴] به دنبال تبدیل محدودیت‌های نابرابری موجود در ماشین بردار پشتیبان به محدودیت‌های برابری هستند. برای این کار باید محدودیت‌های نمونه‌های مثبت و منفی تغییر کنند. به همین منظور در ماشین بردار پشتیبان مربعات حداقل به دنبال معادله ابرصفحه‌ای هستیم که تمام نمونه‌های مثبت و منفی بر روی لبه‌های حاشیه این ابرصفحه قرار گیرند. از طرفی پیداکردن چنین ابرصفحه‌ای عملاً امکان‌پذیر نمی‌باشد و بنابراین باید ابرصفحه‌ای را یافت که با در نظر گرفتن فاصله هر نمونه از لبه حاشیه آن، علاوه بر این که پهنای حاشیه‌اش بیشینه باشد، فاصله هر نمونه نیز از لبه حاشیه این ابرصفحه حداقل باشد (یعنی سعی شود تا آنجایی که امکان دارد نمونه‌های مختلف بر روی لبه حاشیه قرار گیرند) که به این ترتیب محدودیت‌های نابرابری به محدودیت‌های برابری تبدیل خواهند شد. با تبدیل محدودیت‌ها به تساوی، (۲) به (۳) تبدیل خواهد شد

$$\text{Minimize } f(x) = \frac{\|w\|^2}{2} \qquad (3)$$
$$\text{subject to : } y_i(w.x_i + b) - \xi_i = 0$$

همان گونه که در (۳) آمده است برای محاسبه فاصله هر نمونه تا لبه حاشیه از فرمول مربعات حداقل استفاده شده است. بیان دوباره این نکته ضروری است که هنگامی که میزان خطا (فاصله هر نمونه از لبه حاشیه) درون فرمول کمینه‌سازی وجود دارد دیگر نیازی به محدودیت نابرابری نیست، چون خطای آن محاسبه می‌شود و بنابراین محدودیت نابرابری به محدودیت برابری تبدیل خواهد شد.

حال برای کمینه‌سازی (۳) باز هم می‌توان از مضارب لاگرانژ استفاده کرد، با این تفاوت که این بار احتیاجی به حل یک مسئله برنامه‌نویسی غیر خطی نیست و مستقیماً می‌توان مقادیر مضارب لاگرانژ، b و w را به دست آورد. تابع لاگرانژ به دست آمده برای (۳) در (۴) نمایش داده شده که در صورت حل آن به یک دستگاه $m+1$ معادله و $m+1$ مجهول خواهیم رسید که مجهولات را می‌توان با حل (۵) به دست آورد

$$L(w,b,\Delta) = \frac{\mu}{2}\|w\|^2 + \frac{\zeta}{2}\sum_{i=1}^{n}\xi_i^2 - \sum_{i=1}^{n}\lambda_i w.x - \sum_{i=1}^{n}\lambda_i b + \sum_{i=1}^{n}\lambda_i y_i - \sum_{i=1}^{n}\lambda_i \xi_i \qquad (4)$$

$$\begin{bmatrix} 0 & e \\ e & \Omega + \zeta^{-1} I_m \end{bmatrix} \begin{bmatrix} b \\ \Delta \end{bmatrix} = \begin{bmatrix} 0 \\ Y \end{bmatrix} \qquad (5)$$

1. Least Squares Support Vector Machine

پارامتر e، I_m و Ω به ترتیب برداری با طول مورد نیاز و درایه‌های با مقدار یک، ماتریس همانی به اندازه m (تعداد نمونه‌ها) و ماتریسی که درایه i و j آن بیانگر ضرب داخلی x_i و x_j می‌باشد را بیان می‌کنند.

الگوریتم LSSVM در بیشتر کاربردها دارای سرعت و دقت بیشتری از الگوریتم SVM است [۵].

۲-۳ الگوریتم ماشین بردار پشتیبان دوقلو

ماشین‌های بردار پشتیبان دوقلو [۶] در ذات، شبیه ماشین‌های بردار پشتیبان مقدار ویژه می‌باشند، یعنی در اینجا هم به دنبال به دست آوردن دو ابرصفحه ناموازی هستیم. اما تفاوت آن با رویکرد ماشین‌های بردار پشتیبان مقدار ویژه در این است که در اینجا به دنبال بیشینه‌کردن فاصله از کلاس مقابل نیستیم بلکه نمونه‌های کلاس مقابل تنها به صورت محدودیت در معادله ابرصفحه جاری ظاهر می‌شوند (یعنی نمونه‌های کلاس مقابل باید حداقل حاشیه‌ای را با ابرصفحه کلاس جاری رعایت کنند). محدودیت‌های یادشده در معادلات ارائه‌شده برای هر یک از صفحات به صورت نابرابری ظاهر خواهند شد (همانند ماشین بردار پشتیبان عادی)، اما این محدودیت‌های نابرابری برای هر یک از ابرصفحات به اندازه نمونه‌های موجود در کلاس مقابل می‌باشد. بنابراین اگر تعداد نمونه‌های مثبت و منفی برابر باشند، این محدودیت‌های نابرابری برای هر یک از معادلات به اندازه نیمی از محدودیت‌های نابرابری در ماشین بردار پشتیبان ارائه‌شده در بخش دو است. بنابراین به صورت کلی می‌توان گفت که به جای حل معادله برنامه‌نویسی غیر خطی با محدودیت‌های فراوان، ماشین بردار پشتیبان دوقلو، دو معادله با محدودیت‌های بسیار کمتر را حل می‌کند که علاوه بر افزایش دقت، باعث افزایش سرعت نیز (تا چهار برابر) می‌گردد.

همان گونه که گفته شد الگوریتم ماشین بردار پشتیبان دوقلو سعی دارد دو ابرصفحه (برای هر یک از کلاس‌های مثبت و منفی) را به گونه‌ای بیابد که علاوه بر این که به نمونه‌های کلاس خود نزدیک‌ترین هستند برای نمونه‌های کلاس مقابل نیز حداقل حاشیه‌ای را (با در نظر گرفتن خطای y) در نظر بگیرند. برای به دست آوردن چنین ابرصفحاتی الگوریتم مورد نظر باید قادر به حل روابط کمینه‌سازی (۶) و (۷) به ترتیب برای کلاس مثبت و کلاس منفی باشد

$$\min_{w^{(1)},b^{(1)},q} \frac{1}{2}(Aw^{(1)}+e_1b^{(1)})^T(Aw^{(1)}+e_1b^{(1)})+c_1e_2^Tq \quad (۶)$$
$$\text{subject to}: -(Bw^{(1)}+e_2b^{(1)})+q\geq e_2\ ,\ q\geq 0$$

$$\min_{w^{(1)},b^{(1)},q} \frac{1}{2}(Bw^{(2)}+e_2b^{(2)})^T(Bw^{(2)}+e_2b^{(2)})+c_2e_1^Tq \quad (۷)$$
$$\text{subject to}: -(Aw^{(2)}+e_1b^{(2)})+q\geq e_1\ ,\ q\geq 0$$

برای حل هر یک از این معادلات می‌توان همانند ماشین بردار پشتیبان از مضرب‌های لاگرانژ و نهایتاً از برنامه‌نویسی غیر خطی استفاده کرد.

۲-۴ ماشین بردار پشتیبان دوقلوی مربعات حداقل

ماشین بردار پشتیبان دوقلوی مربعات حداقل [۷] سعی در ترکیب ایده‌های موجود در ماشین بردار پشتیبان مربعات حداقل و ماشین بردار پشتیبان دوقلو دارد. به بیان دیگر از نقاط قوت هر دو رویکرد استفاده کرده و از نقاط ضعف آنها دوری می‌گزینند. بنابراین می‌توان گفت که ماشین بردار پشتیبان دوقلوی مربعات حداقل به جای حل معادله برنامه‌نویسی غیر خطی با محدودیت‌های نابرابری فراوان، دو معادله با محدودیت‌های برابری بسیار کمتر (تقریباً نصف) را حل می‌کند. به عبارت دیگر ماشین بردار پشتیبان دوقلوی مربعات حداقل، به دنبال معادلات دو ابرصفحه است که هر کدام از آنها علاوه بر این که سعی دارند به نمونه‌های کلاس خود نزدیک‌ترین باشند، تلاش می‌کنند که با در نظر گرفتن یک فاصله، تمام نمونه‌های کلاس مقابل را بر روی لبه حاشیه خود قرار دهند. به این ترتیب علاوه بر این که در ماشین بردار پشتیبان دوقلوی مربعات حداقل، دو ابرصفحه با مشخصات ذکرشده در ماشین بردار پشتیبان دوقلو وجود دارد، هر یک از آنها از ویژگی ماشین بردار پشتیبان مربعات حداقل (یعنی در نظر گرفتن خطا به صورت مربعات حداقل برای قرارگرفتن نمونه‌های کلاس مقابل در لبه حاشیه ابرصفحه) نیز سود می‌برند. این تعریف از دو ابرصفحه منجر به رابطه کمینه‌سازی (۸) و (۹) برای به دست آوردن معادله هر دو ابرصفحه می‌گردد. برای حل این دو معادله کمینه‌سازی نیز می‌توان از مضرب‌های لاگرانژ استفاده کرد که در این صورت مقادیر w و b برای هر یک از ابرصفحات (با در نظر گرفتن $E=[A\ e]$ و $F=[B\ e]$ که e یک بردار با طول مناسب و درایه‌های با مقدار یک می‌باشد) طبق (۱۰) و (۱۱) محاسبه می‌گردد. یادآوری این نکته نیز ضروری است که در این دو مورد احتیاجی به استفاده از راه حل‌های برنامه‌نویسی غیر خطی نمی‌باشد

$$\min_{w^{(1)},b^{(1)}} \frac{1}{2}(Aw^{(1)}+e_1b^{(1)})'(Aw^{(1)}+e_1b^{(1)})+\frac{c_1}{2}y'y \quad (۸)$$
$$\text{subject to}: -(Bw^{(1)}+e_2b^{(1)})+y=e$$

$$\min_{w^{(1)},b^{(1)}} \frac{1}{2}(Bw^{(2)}+e_2b^{(2)})'(Bw^{(2)}+e_2b^{(2)})+\frac{c_2}{2}y'y \quad (۹)$$
$$\text{subject to}: -(Aw^{(2)}+e_1b^{(2)})+y=e$$

$$\begin{bmatrix} w^{(1)} \\ b^{(1)} \end{bmatrix} = -(F'F+\frac{1}{c_1}E'E)^{-1}F'e \quad (۱۰)$$

$$\begin{bmatrix} w^{(2)} \\ b^{(2)} \end{bmatrix} = -(E'E+\frac{1}{c_2}F'F)^{-1}E'e \quad (۱۱)$$

۲-۵ الگوریتم ماشین بردار پشتیبان دوقلوی مربعات حداقلی فازی

در این بخش می‌خواهیم راهکار ارائه‌شده در مورد کاربرد بنیادی مفهوم فازی را در الگوریتم LSTSVM (یعنی بیان فازی دو ابرصفحه مورد استفاده در الگوریتم F-LSTSVM و روند بهینه‌سازی معادلات آن) بیان کنیم. بنابراین به دنبال بیان مفهوم ابرصفحه فازی و تمایز بین داده‌های کلاس هدف و دیگر کلاس به صورت فازی هستیم. تمام پارامترهای مورد استفاده در این مدل فازی هستند، حتی مؤلفه‌های موجود در بردار W به صورت اعداد فازی بیان می‌شوند. بیان این نکته نیز ضروری است که تمام پارامترهای مورد استفاده در این مدل با تابع عضویت مثلثی بیان شده‌اند.

در الگوریتم سلیمی و همکاران [۸] تمام مؤلفه‌های موجود در بردار وزنی W و ترم بایاس b که در معادله ابرصفحه مورد استفاده قرار می‌گیرند به صورت عدد فازی مثلثی متقارن بیان شده است (هرچند که می‌توان از سایر توابع عضویت نیز استفاده کرد). بردار وزنی فازی $W=(w,c)$ و ترم بایاس فازی $B=(b,d)$ را در نظر بگیرید. W یک بردار وزنی فازی می‌باشد که هر مؤلفه از آن یک عدد فازی است که به

$$\begin{bmatrix} w^{(1)} \\ b^{(1)} \\ c^{(1)} \\ d^{(1)} \end{bmatrix} = \begin{bmatrix} \mu N'N + \frac{1}{c_1} P'P & \mu N'e + \frac{1}{c_1} P'e & \frac{1}{c_1} P'P & \frac{1}{c_1} P'e \\ \mu e'N + \frac{1}{c_1} e'P & \mu m_2 + \frac{1}{c_1} m_1 & \frac{1}{c_1} e'P & \frac{1}{c_1} m_1 \\ P'P & P'e & M + P'P & P'e \\ e'P & m_1 & e'P & m_1 \end{bmatrix} \begin{bmatrix} \mu N'e \\ \mu \\ 0e \\ Me \end{bmatrix} \quad (۱۴)$$

شکل $W_i = (w_i, c_i)$ بیان می‌شوند. به بیان دیگر برای بردار وزنی W ما دو بردار $[w_1, w_2, \ldots, w_n]^t$ و $[c_1, c_2, \ldots, c_n]^t$ خواهیم داشت که یکی مراکز و دیگری پهنای اعداد فازی مثلثی را بیان می‌کنند. در این صورت ابرصفحه فازی $Y = W_1 x_1 + \ldots + W_n x_n + B = \langle W.x \rangle + B$ توسط تابع عضویت زیر تعریف می‌گردد [۸]

$$\mu_Y(y) = \begin{cases} 1 - \dfrac{|y - (\langle w.x \rangle + b)|}{\langle c.|x| \rangle + d} & , \ x \neq 0 \\ 1 & , \ x = 0 \ , \ y = 0 \\ 0 & , \ x = 0 \ , \ y \neq 0 \end{cases} \quad (۱۲)$$

که $\mu_Y(y) = 0$ است هنگامی که $\langle c.|x| \rangle + d \leq |y - (\langle w.x \rangle + b)|$ باشد. حال اگر با توجه به تابع عضویت یک ابرصفحه فازی بخواهیم (۱۵) را بازنویسی کنیم خواهیم داشت

$$\begin{aligned} &\underset{w^{(1)}, b^{(1)}, c^{(1)}, d^{(1)}}{\text{Minimize}} \ J = \frac{1}{2}(P.W^{(1)} + eB^{(1)})(P.W^{(1)} + eB^{(1)}) \\ &+ \frac{C_1}{2} \mu y'y + M(\frac{1}{2} c^{(1)2} + d^{(1)}) \quad (۱۳) \\ &\text{subject to} : (\langle N.W^{(1)} \rangle + eB^{(1)}) + \Theta + y = 0e \end{aligned}$$

در این رابطه Θ به عنوان حاشیه فازی[1] در نظر گرفته شده است (همانند حاشیه‌ای که در شکل یک قرار دارد تنها با این تفاوت که به صورت فازی تعریف می‌گردد) و در اینجا یک عدد فازی به مرکز صفر و پهنای $O_w.w^2$ می‌باشد. در این رابطه ترم $(1/2)c^2 + d$ میزان ابهام[2] مدل را بیان می‌کند و این بدان معناست که ابهام بیشتر inexactness پاسخ را افزایش می‌دهد. در معادله (۲۷)، M وظیفه تنظیم ابهام را بر عهده دارد. همچنین $(C_1/2)\mu y'y$ بیانگر میزان خطای مربعات حداقل می‌باشد که در آن بردار μ درجه عضویت هر نمونه مثبت و بردار y ضرایب slack (که میزان انحراف از قیدهای معادله را بیان می‌کنند) را بیان می‌کنند. میزان تأثیر خطای مربعات حداقل در معادله ابرصفحه مربوط به کلاس مثبت توسط ضریب C1 تنظیم می‌گردد.

با حل معادله ابرصفحه (۱۳)، (۱۴) را خواهیم داشت. حال که تمام پارامترها مقادیرشان مشخص شد، تابع عضویت ابرصفحه فازی بهینه برای نمونه‌های مثبت همانند زیر به دست می‌آید

$$\mu_{Y_i^*}(y) = 1 - \frac{|y - (\langle w^{(1)}.x \rangle + b^{(1)})|}{\langle c^{(1)}.|x| \rangle + d^{(1)}} \quad (۱۵)$$

همین روند را می‌توان برای ابرصفحه دوم نیز به کار برد.

۳- کارهای پیشین انجام‌شده در زمینه الگوریتم افزایشی ماشین‌های بردار پشتیبان

تا کنون الگوریتم‌های زیادی در زمینه یادگیری افزایشی ارائه شده است. لوزینگ و همکاران [۹] در سال ۲۰۱۸ اغلب الگوریتم‌های معروف افزایشی را بر روی طیف وسیع و متنوعی از دیتاست‌ها اجرا کردند و نشان دادند که نسخه‌های افزایشی الگوریتم ماشین بردار پشتیبان اغلب بیشترین دقت را دارند. نسخه‌های متعدد یادگیری افزایشی با استفاده از الگوریتم ماشین بردار پشتیبان در سال‌های اخیر ارائه شده است. شاید اولین ایده در زمینه تبدیل الگوریتم ماشین بردار پشتیبان به نسخه افزایشی آن، در نظر گرفتن نقاط و داده‌های بردار پشتیبان از مراحل قبل در کنار داده‌های ورودی جدید باشد که کار یادگیری بر روی آنها انجام می‌شود [۱۰] و [۱۱]. نسخه افزایشی ماشین بردار پشتیبان ارائه‌شده در [۱۱] یک راه حل دقیق برای ماشین بردار پشتیبان برخط می‌باشد که با اضافه‌کردن یا حذف یک نمونه در هر مرحله، کار را انجام می‌دهد. گلوگاه الگوریتم ارائه‌شده پیچیدگی محاسباتی است. تا کنون نسخه‌های متعددی از الگوریتم ماشین بردار پشتیبان ارائه شده که یکی از این نسخه‌ها ماشین بردار پشتیبان دوقلوست. کمچندانی و همکاران نسخه افزایشی ماشین بردار پشتیبان دوقلو را ارائه کردند که از مفهوم بردار حاشیه و بردار خطا برای انتخاب نمونه بعدی بهره می‌بردند [۱۰]. در سال ۲۰۱۴ یونه و همکاران یک نسخه سریع از الگوریتم افزایشی ماشین بردار پشتیبان دوقلو ارائه کردند که از استراتژی مبتنی بر فاصله برای انتخاب نمونه بعدی استفاده می‌کرد [۱۲]. در پژوهشی دیگر در سال ۲۰۱۶ علمدار و همکاران، الگوریتم افزایشی ماشین بردار پشتیبان مستقل دوقلو را ارائه کردند که از متد نیوتن اصلاح‌یافته برای ساخت یک تابع تصمیم بر اساس داده‌های رؤیت‌شده تا کنون استفاده می‌کرد [۱۳]. در سال‌های بعد، فونگ و همکاران استراتژی به روز رسانی مدل در این الگوریتم را بر اساس یک پنجره زمانی تغییر دادند [۱۴]. همچنین تویت و همکاران در پژوهشی دیگر این استراتژی را بر اساس کاهش ضرایب قرار داده بودند [۱۵]. در سال ۲۰۲۰ الکساندر و همکاران یک نسخه افزایشی فازی از الگوریتم ماشین بردار پشتیبان دوقلو ارائه کردند [۱۶]. آقای ژو و همکاران در سال ۲۰۱۹ نسخه جدیدی از الگوریتم افزایشی مبتنی بر ماشین بردار پشتیبان ارائه کردند که انتخاب نمونه‌ها در آن بر اساس مدل مارکوف انجام می‌شد [۱۷]. آنها نشان دادند که الگوریتمشان در کاربردهای دسته‌بندی، کمتر دچار خطا می‌شود.

بررسی‌های انجام‌شده در ادامه تا حدود زیادی منطبق بر مقاله مروری لاوال [۱۸] است. متدهای ارائه‌شده برای رویکرد افزایشی در ماشین‌های بردار پشتیبان به دو دسته برخط و شبه برخط تقسیم می‌شوند [۱۹] و [۲۰]. متدهای برخط داده‌های جریانی را به صورت تکی در هر لحظه پردازش می‌کند و مطمئن می‌شود که شرایط لازم برای بردارهای پشتیبان در مدل تغییری نکرده باشد. در مدل شبه برخط نمونه داده‌های جریانی به

1. Fuzzy Margin
2. Vagueness

- Input: (x_1^{t+1}, y_1^{t+1}) : new sample at time $t+1$, f^t : decision function at t
- Output: f^{t+1} : updated decision function at $t+1$
- Definitions: α_i^t : coefficient of the i th sample at t, S_m^t : margin support vector set at t

1. Begin
2. Compute $z = y_1^{t+1} f^t(x_1^{t+1})$,
3. If $z > 1$ then $f^{t+1} \leftarrow f^t$ and go to step 10
4. Else
5. Initialize $\alpha_1^{t+1} \leftarrow 0$ for x_1^{t+1}
6. Compute Q_{1i} for all $x_i^t \in S_m^t$
7. Increment α_1^{t+1} to its largest value while maintaining the KKT optimality conditions on all previously seen samples
8. Check if one of the following conditions occurs:
9. $f^{t+1} \leftarrow f^t$, $S_m^{t+1} \leftarrow S_m^t$ and $S_b^{t+1} \leftarrow S_b^t$

I. If $y_1^{t+1} f^t(x_1^{t+1}) = 1$ then $S_m^t \leftarrow x_1^{t+1}$

II. Else if $\alpha_1^{t+1} = C$ then $S_b^t \leftarrow x_1^{t+1}$

III. Else if any $x_i^t \in S_m^t$ become part of S_b^t, due to the change in their corresponding α_i^t, then update S_m^t and S_b^t accordingly

10. End

شکل ۱: متد کاونبرگ و همکارش [۲۱].

صورت دسته‌ای[1] نمونه‌های جدید را پردازش می‌کند. در این متد زمانی که تصمیم به اجرای دوباره الگوریتم SVM شد، این الگوریتم بر روی داده‌های جدید به همراه بردارهای پشتیبانی که تا کنون به دست آمده‌اند آموزش خواهد دید.

۳-۱ متدهای برخط SVM افزایشی

اولین الگوریتم افزایشی SVM توسط کاونبرگ و همکارش پیشنهاد شد [۲۱] و در ادامه نسخه‌های متفاوتی از این کار توسط محققان ارائه گردید که به عنوان نمونه می‌توان به [۲۲] و [۲۳] اشاره کرد. در تمامی این کارها زمانی که نمونه‌ای جدید همانند (x^{t+1}, y^{t+1}) دریافت می‌شود، ابتدا $y^{t+1} f^t(x^{t+1})$ را محاسبه می‌کنند تا بر اساس آن تشخیص دهند آیا نمونه جدید می‌تواند منجر به بهبود دسته‌بند جاری شود یا خیر. در این رابطه تابع f^t همان دسته‌بند SVM است. اگر $y^{t+1} f^t(x^{t+1}) \le 1$ باشد نمونه جدید (x^{t+1}, y^{t+1}) یک نمونه مفید برای دسته‌بند خواهد بود. بنابراین ضریب لاگرانژ این نمونه را برابر صفر قرار داده و با افزایش تدریجی آن مقدار بهینه SVM را به دست می‌آورد. الگوریتم دقیق این روش در شکل ۱ آورده شده که در آن S مجموعه بردارهای پشتیبان و α ضریب لاگرانژ است.

در کاری دیگر، نسخه تک‌کلاسی SVM افزایشی به نسخه چندکلاسی آن تبدیل شد که البته بار محاسباتی آن به مراتب بیشتر شده است [۲۴]. بوخاروبا و همکارانش همین نسخه از SVM افزایشی را برای حالتی که داده‌های بدون برچسب فراوانی وجود دارد توسعه دادند [۲۵]. همچنین چن و همکارانش کاری مشابه و با استفاده از اصل انتقال[2] ارائه کردند [۲۶]. در کاری دیگر در زمینه متدهای برخط، رای و همکارانش الگوریتمی را با استفاده از اصل حداقل توپ محصور[3] به منظور یادگیری و به روز رسانی SVM پیشنهاد دادند [۲۷].

- Input $F^t = \{(x_1^{t+1}, y_1^{t+1}), \ldots, (x_N^{t+1}, y_N^{t+1})\}$: new samples at time $t+1$, S_m^t : margin support vector set at t, S_b^t : bounded support vector set at t, f^t : decision function at t
- Output: f^{t+1} : updated decision function, S_m^{t+1} : margin support vector set at $t+1$, S_b^{t+1} : bounded support vector set at $t+1$
- Definitions: α_i^{t+1} : coefficient of the i th sample at $t+1$,

1. Begin
2. Compute $\bar{F}^{t+1} = \{F^t \cup S_m^t \cup, S_b^t\}$,
3. Using $\bar{F}^{t+1}$ as input, solve for the problem of (11) in [] to obtain α_i^{t+1} for all i
4. Compute f^{t+1} by solving the problem of (12) in []
5. Compute $S_m^{t+1} = \{S_m^t \cup x_i^{t+1} : y_i^{t+1} f^{t+1}(x_i^{t+1}) = 1 \;\&\; \alpha_i^{t+1} > 0\}$
6. Compute $S_b^{t+1} = \{S_b^t \cup x_i^{t+1} : y_i^{t+1} f^{t+1}(x_i^{t+1}) < 1 \;\&\; \alpha_i^{t+1} = C\}$
7. Discard all x_i^{t+1} for which $\alpha_i^{t+1} = 0$
8. Return $f^{t+1}, S_m^{t+1}, S_b^{t+1}$
9. End

شکل ۲: متد ساید و همکارانش [۲۸].

۳-۲ متدهای شبه برخط SVM افزایشی

اولین کار در این زمینه توسط ساید و همکارانش ارائه شد [۲۸]. در این الگوریتم، داده‌ها در دسته‌هایی با اندازه‌های ثابت پردازش می‌شوند. اولین دسته از داده‌ها در زمان t که شامل همه نمونه‌ها تا آن لحظه می‌گردد، به عنوان ورودی الگوریتم SVM استفاده می‌شود. سپس الگوریتم، همه داده‌های مرحله قبل را به جز بردارهای پشتیبان از مجموعه حذف می‌کند. زمانی که می‌خواهد الگوریتم را بر روی داده‌های موجود در دسته بعدی اعمال کند، این داده‌ها را با مجموعه بردارهای پشتیبان موجود در دسته قبل ترکیب می‌کند و بر اساس مجموع آنها کار یادگیری انجام می‌شود. این کار در هر زمان که دسته جدیدی از داده‌ها آماده شود تکرار می‌گردد. شکل ۲ این الگوریتم را نمایش می‌دهد. اما این متد نمی‌تواند زمانی که توزیع داده‌ها تغییر می‌کند به درستی عمل کند. دلیل این امر نیز حضور بردارهای پشتیبان متعلق به مراحل قبلی در مراحل بعدی می‌باشد. معضل دیگر این الگوریتم، کنارگذاشتن همه داده‌ها به جز بردارهای ماشین است که ممکن است اقدام درستی نباشد و داده‌ای که در این مرحله بردار پشتیبان نبوده است با حضور داده‌های مراحل بعد یکی از بردارهای پشتیبان باشد. معضل دیگر این الگوریتم، ذخیره‌سازی همه بردارهای پشتیبان مراحل قبلی است که ممکن است در مراحل بعد دیگر بردار پشتیبان نباشند. برای حل معضل آخر در برخی روش‌ها، بردارهای پشتیبان به دست آمده در هر مرحله از نظر وابستگی خطی بین خودشان با توجه به مجموعه ویژگی‌ها بررسی می‌شوند [۲۹] و [۳۰]. اگر این وابستگی وجود داشت بهتر است تنها بردارهایی نگهداری شوند که به عنوان تابعی از سایرین نباشند.

برای حل مشکل دوم از ترکیب SVM با تکنیک‌های بهینه‌سازی همانند PSO استفاده می‌کنند تا بهترین مقادیر پارامترها در هر مرحله محاسبه شود [۳۱]. همچنین برای حل معضل نخست، دومینکنی و همکارانش الگوریتم ساید [۲۸] را توسعه دادند [۳۲]. آنها بدین گونه عمل می‌کنند که در هر لحظه که دسته داده جدیدی دریافت می‌شود، الگوریتم SVM در قالب k دسته‌بند متفاوت با توجه به k دسته اخیر به روز رسانی می‌شود. به عبارت دیگر در لحظه t ما k مدل ایجادشده از SVM داریم که به ترتیب شامل همه k دسته اخیر، $k-1$ دسته اخیر و

1. Batch
2. Principle of Transduction
3. Minimum Enclosing Ball

- Step 1. Calculate initial $u=[w_1,b_1]^T$ and $v=[w_2,b_2]^T$
  $u=\text{TWSVM(A)}$, $v=\text{TWSVM(B)}$
- Step 2. Calculate distance threshold T_u and T_v
  $d_u=\frac{\left|w_1^T A+b_1\right|}{w_1}$, $d_v=\frac{\left|w_2^T B+b_2\right|}{w_2}$
  $T_u=\min(d_u)+\theta*(\max(d_u)-\min(d_u))$
  $T_v=\min(d_v)+\theta*(\max(d_v)-\min(d_v))$
- Step 3. Add the new training sample set A_{new} and B_{new}, and establish the dynamic training sample set D_A and D_B
  $D_A=(A<T_u)\cup(A_{new}>T_u)$
  $D_B=(B<T_v)\cup(B_{new}>T_v)$
  $A=D_A, B=D_B$
- Step 4. Incremental Learning
  $u=\text{TWSVM}(D_A), v=\text{TWSVM}(D_B)$
- Step 5. If there will be a new training sample set in the future, we repeat Step 2, Step 3 and Step 4.

شکل ۳: متد هو و ژانگ [۱۲].

یک دسته اخیر می‌باشند. هر ورودی جدید که دریافت می‌شود یک دسته‌بند ایجادشده از بین می‌رود و یک دسته‌بند جدید ایجاد می‌گردد.

برای الگوریتم TSVM نیز چندین نسخه افزایشی ارائه شده که در ادامه به مرور برخی از آنها می‌پردازیم. الساندرو و همکاران یک نسخه افزایشی از TSVM با ترکیب آن با مفاهیم فازی ارائه داده‌اند [۳۳]. آنها بدین گونه عمل می‌کنند که به هر نمونه ورودی یک درجه اهمیت منتسب می‌کنند و سپس با استفاده از ابرصفحه‌های فازی‌ای که ارائه داده‌اند، این نمونه‌ها را دسته‌بندی می‌کنند. میزان اهمیت نمونه‌های جدید بیشتر و میزان اهمیت نمونه‌های قدیمی رفته‌رفته کمتر می‌شود. الگوریتم آنها به صورت شبه برخط عمل می‌کند و آنها الگوریتمشان را برای حالت چندکلاسی نیز گسترش دادند.

در مطالعه‌ای دیگر خمچندانی و همکاران یک مدل ساده از TSVM افزایشی ارائه دادند که بسیار شبیه به الگوریتم ساید [۲۸] در بخش ۲-۱-۲ عمل می‌کند [۱۰]. آنها در هر مرحله علاوه بر داده‌های دسته جدید، داده‌های متعلق به بردارهای پشتیبان مراحل قبل و نیز داده‌هایی که در باند هر ابرصفحه قرار می‌گیرند را به الگوریتم TSVM در آن مرحله می‌دهند و سپس بر اساس آن کار دسته‌بندی را انجام می‌دهند. همچنین هو و ژانگ در سال ۲۰۱۴ میلادی یک الگوریتم سریع افزایشی مبتنی بر TSVM ارائه دادند [۱۲]. آنها با استفاده از فاصله نمونه‌های موجود در هر دسته با هر یک از ابرصفحه‌ها، وزن‌دهی به دسته‌های داده‌ها را انجام می‌دهند و سپس بر اساس آن کار یادگیری اتفاق می‌افتد. شکل ۳ متد آنها را نمایش می‌دهد. با توجه به کارهای ارائه‌شده در این بخش و با توجه به دانش نویسندگان، هیچ کاری تا کنون برای ارائه نسخه افزایشی الگوریتم FLSTSVM ارائه نشده است. این پژوهش به آن پرداخته و ایده جدیدی حتی نسبت به سایر رویکردهای معرفی‌شده ارائه می‌دهد.

٤- الگوریتم پیشنهادی FLSTSVM افزایشی

همان گونه که در بخش ۲ به طور کامل توضیح داده شد، یک الگوریتم افزایشی باید در خصوص نحوه به روز رسانی دسته‌بند ایجادشده با توجه به جریان داده‌ها و نمونه‌های ورودی جدید تصمیم بگیرد. ممکن است این تصمیم بر اساس هر نمونه باشد (یعنی برخط) و یا بر اساس تعداد ثابتی از نمونه‌های دریافتی جدید باشد (یعنی شبه برخط). در این پژوهش دو الگوریتم افزایشی برای هر دو نوع برخط و شبه برخط بر اساس الگوریتم جدید FLSTSVM ارائه شده است. در ابتدا الگوریتم مورد نظر برای حالت برخط ارائه می‌شود.

الگوریتم برخط ارائه‌شده بر مبنای الگوریتم مربعات حداقلی ماشین‌های بردار پشتیبان فازی است که اخیراً ارائه شده است [۸]. دلایل انتخاب و بهبود این الگوریتم عبارت است از: ۱) الگوریتم SVM از نظر دقت، عملکرد بسیار خوبی دارد. ۲) نسخه TSVM از نظر دقت و سرعت از الگوریتم SVM در بسیاری از موارد بهتر عمل می‌کند. ۳) نسخه LSTSVM از نظر سرعت تقریباً سرعتی تا ۸ برابر نسخه اصلی الگوریتم ماشین بردار پشتیبان دارد. ۴) نسخه FLSTSVM علاوه بر بهبود دقت نسبت به LSTSVM به دلیل استفاده از مفهوم فازی در آن بسیار مناسب یادگیری دنباله‌ای است که در آن داده‌ها به صورت جریانی تولید می‌شوند. به عنوان مثال، در برخی از کاربردها می‌خواهیم درجه اهمیت نمونه‌های اخیر بیشتر از نمونه‌های در گذشته دور باشد. در این موارد ما می‌توانیم از یک تابع عضویت فازی که بر اساس زمان طراحی شده است استفاده کنیم که در این صورت این تابع می‌تواند بر اساس زمان، به هر نمونه یک درجه عضویت اختصاص دهد. در ادامه این الگوریتم توضیح داده می‌شود.

٤-١ نسخه برخط الگوریتم افزایشی FLSTSVM

در الگوریتم برخط ارائه‌شده سعی بر آن است الگوریتم FLSTSVM به نحوی تغییر یابد که بتوان از داده‌های جریانی که یکی از چالش‌های موجود در بسیاری از کاربردها همانند اینترنت اشیا است، به طور بهینه برای یادگیری افزایشی استفاده نمود. فرض می‌کنیم در یادگیری افزایشی تا لحظه tام داده‌های $\{(x_1,y_1),(x_2,y_2),\ldots,(x_t,y_t)\}$ را دریافت کرده‌ایم. اگر تنها بخواهیم بر اساس همین داده‌ها عمل کنیم به راحتی می‌توانیم الگوریتم FLSTSVM را بر روی آنها اعمال کرده و مدل مورد نظر را ایجاد کنیم. در ادامه نیز با دریافت هر نمونه می‌توانیم از مدل ساخته‌شده به منظور پیش‌بینی استفاده نماییم. اما مشکل جایی است که می‌خواهیم نمونه‌های بعدی را نیز در ساخت مدل مشارکت دهیم. اما اگر بخواهیم با دریافت هر نمونه، کل الگوریتم را از ابتدا اجرا کنیم از نظر زمانی و توان محاسباتی با مشکل مواجه خواهیم شد. ضمناً همه نمونه‌هایی که در آینده دریافت می‌کنیم بااهمیت نیستند و شاید در مدل تأثیر چندانی نداشته باشند و بنابراین اهمیت نمونه‌ها متفاوت خواهد بود. حال ما باید به سه نکته توجه کنیم: ۱) چگونه می‌توانیم اهمیت نمونه‌های مختلف را به دست آوریم؟ ۲) چگونه می‌توانیم این اهمیت را در الگوریتم دخالت دهیم؟ ۳) چگونه می‌توانیم بدون نیاز به این که تمامی نمونه‌های قبلی را اجرا کنیم مدل جدید را بسازیم؟ در پاسخ به سؤال اول، ۳ معیار مهم ارائه کرده‌ایم که ترکیب آنها، میزان اهمیت نمونه جدید (x_{t+1},y_{t+1}) را بیان می‌کند. اگر مدل ساخته‌شده تا لحظه t را با M و داده‌های تا این لحظه را با D_L نمایش دهیم آن گاه $Inf((x_{t+1},y_{t+1})|M,D_l)$ را به عنوان میزان اهمیت نمونه (x_{t+1},y_{t+1}) تعریف می‌کنیم. این میزان اهمیت بر اساس سه عامل محاسبه می‌شود: فاصله تا ابرصفحه‌ها، تراکم و شباهت نمونه جدید. چگالی نشان‌دهنده نزدیکی نمونه‌ها است. چگالی بالا نشان‌دهنده اطلاعات اضافی نمونه‌ها و محتوای اطلاعاتی حاشیه‌ای است و برعکس. میزان اطلاعات مربوط به چگالی در راهکار ارائه‌شده به صورت (۱۶) محاسبه می‌شود

$$Inf^{Den}(x_{t+1}|D_L)=1-\frac{1}{\left|D_L^{s(t+1)}\right|}\sum_{x_j\in D_L^{s(t+1)}}Cosdise(x_{t+1},x_j) \qquad (16)$$

- Incremental FLSTSVM
- Input: Model $M(t)$, Sample $x(t+1)$, $D_{l(t)}$
- Output: Model $M(t+1)$, $y(t+1)$
- $S(t+1) \leftarrow M(x(t+1))$
- Compute $Inf(x_{t+1}|D_L, M) = \alpha Inf_{Den}(x_i|D_L) + \alpha Inf_{Simi}(x_i|D_L) + \gamma dist_s + \frac{\delta}{dist_{\bar{s}}}$ using:
  - $Inf_{Den}(x_i|D_L) = 1 - \frac{1}{|D_L^{S(t+1)}|} \sum_{x_j \in D_L^{S(t+1)}} cosdise(x_{t+1}, x_j)$
  - $Inf_{Simi}(x_i|D_L) = 1 - \max cosdise(x_{t+1}, x_j)$
  - compute $dist_{\bar{s}(t+1)}$ and $dist_s$
- $\mu(x(t+1)) \leftarrow Inf(x_{t+1}|D_L, M)$
- If $\mu(x(t+1)) < Threshold$
  - $y(t+1) \leftarrow S(t+1)$
- Else
  - $D_{L(t+1)} \leftarrow$ All support vectors in $D_{L(t)} \cup (x(t+1), y(t+1))$
  - $y(t+1) \leftarrow S(t+1)$
  - Apply FLSTSVM on $D_{L(t+1)}$ and obtain $M(t+1)$
- End

شکل ۴: نسخه پیشنهادی برخط الگوریتم افزایشی FLSTSVM.

جدول ۱: توصف ویژگی‌های شش بانک داده استفاده‌شده در الگوریتم پیشنهادی.

Dataset	Missing Value	#Features	#Classes	#Samples
Heart-Statlog	No	۱۳	۲	۲۷۰
Heart-C	No	۱۴	۲	۳۰۳
Ionosphere	No	۳۴	۲	۳۵۴
Votes	Yes	۱۶	۲	۴۴۵
Australian	Yes	۱۴	۲	۶۹۰
Japanese Credit	No	۱۴	۲	۶۵۳

که در آن s برچسب پیش‌بینی شده توسط مدل M، D_L^s مجموعه بردار پشتیبان با برچسب s در D_L و $cosdise$ معیار فاصله $cosine$ است. از شباهت برای اطمینان از تنوع نمونه‌ها استفاده شده است. این نشان می‌دهد که x_{t+1} باید با دیگر نمونه‌ها در D_L متفاوت باشد تا دارای اهمیت باشد. بنابراین اطلاعات مفید را می‌توان بر اساس تشابه به صورت (۱۷) محاسبه کرد

$$Inf^{simi}(x_{t+1}|D_L) = 1 - \max_{x_j} Cosdise(x_{t+1}, x_j) \qquad (17)$$

معیار سوم میزان فاصله نمونه جدید با برچسب پیش‌بینی شده s از دو ابرصفحه موجود در مدل M است که فاصله آن با ابرصفحه متعلق به کلاس s، $dist_s(x_{t+1}, (w_s, b_s, c_s, d_s))$ و با ابرصفحه متعلق به کلاس مقابل s یعنی $(\bar{s})$ برابر $dist_{\bar{s}}(x_{t+1}, (w_s, b_s, c_s, d_s))$ خواهد بود. نهایتاً این سه قسمت با وزن‌های گوناگون برای محاسبه $Inf(x_i|D_L)$ در (۱۸) ترکیب می‌شوند

$$Inf((x_{-t+1}, y_{t+1})|D_L, M) = Inf_{Den}(x_i|D_L) + Inf_{Simi}(x_i|D_L) + dist_s + \frac{\delta}{dist_{\bar{s}}} \qquad (18)$$

در پاسخ به سؤال دوم که چگونه میزان اهمیت نمونه جدید را در الگوریتم تأثیر دهیم نیز این گونه عمل می‌کنیم که با انتخاب مناسب α، β، γ و δ یعنی $\alpha + \beta + \gamma + \delta = 1$ از $Inf((x_{t+1}, y_{t+1})|M, D_l)$ به

- Train $FLSTSVM_i$ with D_i and Save the set of:
  - Support vectors (i.e., SV_i)
  - Erroneous samples (Ei) obtained w.r.t the $FLSTSVM_i$ classifier (i.e., E_i)
  - Samples such as (x_k, y_k) that $Inf((x_k, y_k)|FLSTSVM_i, D_i) \geq$ Threshold where $Inf((x_k, y_k)|FLSTSVM_i, D_i)$ is given by Formula 18 (i.e., In_i)
- Set i ← i + 1.
- $D_i \leftarrow D_i \cup SV_i \cup E_i \cup In_i$
- Use the updated $FLSTSVM_i$ classifier to assess the prediction accuracv of the Test cases.

شکل ۵: نسخه پیشنهادی شبه برخط الگوریتم افزایشی FLSTSVM.

عنوان μ یا همان درجه عضویت نمونه جدید در الگوریتم FLSTSVM استفاده می‌کنیم.

در پاسخ به سؤال سوم نیز بدین گونه عمل می‌کنیم: برای جلوگیری از تکرار عملیات یادگیری بر روی داده‌های قدیمی در صورتی که میزان اهمیت نمونه جدید $Inf((x_{t+1}, y_{t+1})|D_L, M)$ از یک مقدار کمتر باشد، آن را حذف می‌کنیم و در غیر این صورت این نمونه در کنار بردارهای پشتیبان به دست آمده از مدل و داده‌های قدیمی قرار گرفته و با استفاده از این مجموعه کار یادگیری مجدداً انجام می‌شود. الگوریتم بهبودیافته به صورت شبه‌کد شکل ۴ ارائه شده است.

۴-۲ نسخه شبه برخط الگوریتم افزایشی FLSTSVM

در نسخه شبه برخط بر اساس تعداد ثابتی از نمونه‌های دریافتی جدید ایجاد می‌شود. یعنی به جای یک نمونه برای یک دسته از نمونه‌ها باید تصمیم‌گیری کنیم و بر اساس آن، الگوریتم خود را به روز رسانی نماییم. بدین منظور ما در هر بار به روز رسانی از نمونه‌های موجود در سری‌های قبل استفاده می‌کنیم: ۱) تمامی بردارهای پشتیبان مراحل قبل، ۲) نمونه‌هایی که در باند هر یک از ابرصفحه‌ها (نمونه‌های دارای خطا) قرار می‌گیرند و ۳) نمونه‌هایی که بر اساس (۱۸) دارای اطلاعاتی بیش از یک حد آستانه باشند. این حد آستانه با آزمون و خطا تعیین می‌شود.

الگوریتم شبه برخط ارائه‌شده برای نسخه افزایشی FLSTSVM در شکل ۵ آورده شده است.

۵- نتایج آزمایشگاهی

در این بخش ابتدا به معرفی بانک‌های داده مورد استفاده در این مطالعه می‌پردازیم و در ادامه معیارهای ارزیابی دسته‌بند معرفی و نهایتاً نتایج حاصل از اعمال روش پیشنهادی بر روی بانک‌های داده معرفی‌شده به تفصیل بیان می‌گردد.

۵-۱ بانک داده

برای بررسی قدرت و صحت الگوریتم پیشنهادی از ۶ بانک داده مربوط به دو نوع متفاوت از بیماری‌های قلب، دیابت، نارسایی کبد و سرطان سینه استفاده شده است. جدول ۱ مشخصات کلی شش بانک داده را بیان می‌کند. هر شش بانک داده متعلق به مخزن UCI می‌باشند.

۵-۲ نتایج اعمال الگوریتم پیشنهادی بر روی چهار بانک داده

در این بخش نسخه افزایشی FLSTSVM[1]، نسخه افزایشی TWSVM و نسخه افزایشی SVM، با الگوریتم‌های FLSTSVM،

1. Inc-FLSTSVM

جدول ۲: مقایسه صحت الگوریتم پیشنهادی INC-FLSTSVM با سایر روش‌ها.

	Algorithms					
Datasets	Inc-TWSVM	TWSVM	Inc-SVM	SVM	FLSTSVM	Inc-FLSTSVM
Heart-Statlog	۸۴٫۰۷ ± ۴٫۷۰	۸۴٫۸۱ ± ۴٫۵۲	۸۳٫۷۰ ± ۶٫۰۲	۸۳٫۷۰ ± ۴٫۴۴	۸۸٫۷۰ ± ۳٫۱۸	۹۰٫۰۳ ± ۳٫۴۲
Heart-C	۸۳٫۸۰ ± ۵٫۵۳	۸۴٫۱۴ ± ۵٫۷۱	۸۲٫۴۸ ± ۵٫۶۲	۸۴٫۴۴ ± ۵٫۲۷	۸۷٫۶۲ ± ۲٫۸۷	۸۹٫۶۰ ± ۴٫۱۵
Ionosphere	۸۶٫۰۵ ± ۷٫۷۰	۸۷٫۴۶ ± ۶٫۴۲	۸۶٫۸۸ ± ۵٫۶۲	۸۷٫۷۴ ± ۶٫۰۲	۸۹٫۸۳ ± ۳٫۴۱	۹۰٫۸۷ ± ۳٫۶۱
Votes	۹۶٫۳۱ ± ۳٫۴۷	۹۶٫۰۹ ± ۲٫۹۳	۹۵٫۴۰ ± ۲٫۷۱	۹۵٫۸۷ ± ۲٫۶۶	۹۸٫۶۹ ± ۴٫۶۲	۹۸٫۹۵ ± ۲٫۷۰
Australian	۸۶٫۹۶ ± ۲٫۶۷	۸۵٫۹۴ ± ۲٫۸۳	۸۵٫۵۱ ± ۲٫۶۷	۸۵٫۵۱ ± ۲٫۶۷	۹۲٫۹۱ ± ۴٫۵۰	۹۳٫۰۳ ± ۲٫۸۴
Japanese Credit	۸۶٫۶۸ ± ۴٫۰۶	۸۶٫۴۲ ± ۴٫۰۴	۸۶٫۵۲ ± ۴٫۰۴	۸۶٫۳۷ ± ۲٫۹۶	۸۸٫۸۵ ± ۳٫۶۲	۹۰٫۵۵ ± ۳٫۳۳

جدول ۳: مقایسه زمان اجرای الگوریتم INC-FLSTSVM با سایر روش‌ها بر حسب ثانیه.

	Algorithms					
Datasets	Inc-TWSVM	TWSVM	Inc-SVM	SVM	FLSTSVM	Inc-FLSTSVM
Heart-Statlog	۲۳٫۵۰	۲٫۸۴	۵۳٫۷۰	۴٫۹۶	۰٫۰۴۵	۰٫۹۸۰
Heart-C	۳۲٫۱۶	۲٫۹۳	۶۷٫۴۷	۵٫۰۵	۰٫۰۷۰	۱٫۷۵
Ionosphere	۴۱٫۱۷	۳٫۷۴	۸۰٫۸۴	۶٫۴۷	۰٫۱۰۰	۲٫۳۲
Votes	۴۸٫۹۲	۳٫۷۶	۸۱٫۶۵	۶٫۵۲	۰٫۱۱۰	۲٫۵۱
Australian	۷۳٫۷۵	۵٫۹۷	۱۰۵٫۱۰	۹٫۰۱	۰٫۱۹۰	۴٫۲۸
Japanese Credit	۶۸٫۱۲	۳٫۷۷	۱۳۱٫۹۵	۶٫۴۹	۰٫۱۰۴	۲٫۹۰

TWSVM و SVM مقایسه می‌شوند و نشان داده می‌شود که الگوریتم پیشنهادی دارای دقت بهتری نسبت به سایر الگوریتم‌ها است. نتایج مربوط به الگوریتم‌های TWSVM، SVM و نسخه‌های افزایشی آنها از [۱۰] اخذ گردیده است. همچنین دو الگوریتم FLSTSVM و Inc-FLSTSVM در این پژوهش با شرایطی مشابه با [۱۰] پیاده‌سازی شده تا شرایطی عادلانه برای مقایسه وجود داشته باشد. جدول ۲ نتایج حاصل از الگوریتم‌های مذکور را نشان می‌دهد. همان گونه که مشاهده می‌شود الگوریتم پیشنهادی Inc-FLSTSVM در همه مجموعه داده‌ها نتایج بهتری از نظر صحت[1] $((TN+TP)/(TP+TN+FP+FN))$ نسبت به سایر الگوریتم‌ها دارد.

با مقایسه الگوریتم‌های افزایشی موجود از نظر زمان اجرا با یکدیگر، به وضوح الگوریتم افزایشی Inc-FLSTSVM زمان اجرای کمتری را به خود اختصاص می‌دهد. اما باید این نکته را اضافه کنیم که تمامی نسخه‌های افزایشی الگوریتم‌ها از نسخه غیر افزایشی آنها زمان اجرای به مراتب بیشتری دارند. این امر هم به آن دلیل است که عموماً در یک الگوریتم افزایشی یک مدل ممکن است ده‌ها و بلکه صدها و هزاران بار اجرا شود که قاعدتاً منجر به افزایش زمان خواهد شد. این مقایسه‌ها در جدول ۳ آورده شده است. اما نکته قابل توجه این است که نسخه‌های افزایشی در مقابل حالت بدتری از زمان اجرا مطرح می‌شوند و آن هم حالتی است که ما بخواهیم به ازای هر نمونه دریافتی جدید مدل را مجدد آموزش دهیم. همچنین شکل ۶ روش پیشنهادی را با سایر روش‌ها بر اساس امتیاز F۱ مقایسه می‌کند که طبق آن روش پیشنهادی تقریباً بر روی همه دیتاست‌ها وضعیت بهتری دارد.

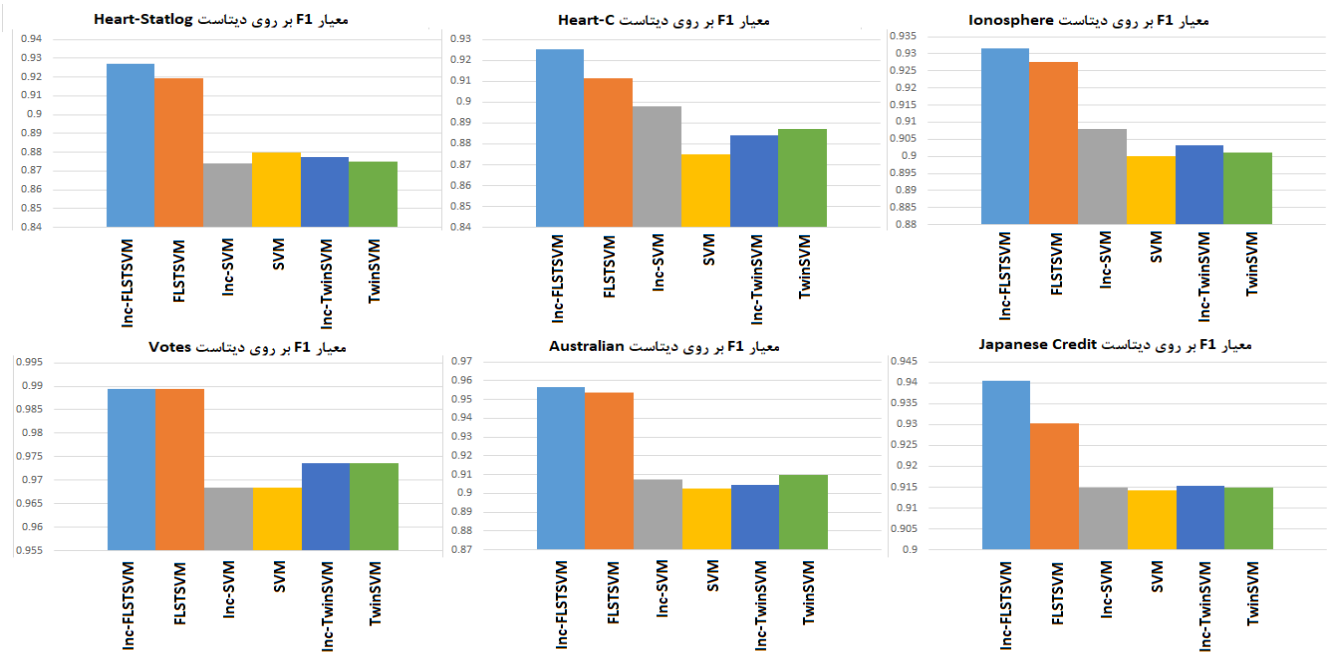
در کاربردی دیگر و به منظور بررسی صحت و دقت الگوریتم ارائه‌شده از مجموعه داده‌های مربوط به فعالیت‌های روزمره زندگی (ADL) استفاده شده است [۳۴]. مجموعه داده‌های موجود شامل ۵ دیتاست است که هر کدام از آنها در شرایط متفاوتی تولید شده‌اند. این تفاوت مربوط به مکان زندگی، تعداد سنسورها و زمان مانیتورینگ افراد است. طبق [۳۵] اطلاعات سنسورها در یک دوره زمانی مشخص اخذ شده است. برای یکسان‌سازی زمان‌ها در این پژوهش، ما یک مقطع زمانی تعریف می‌کنیم که مجموعه داده شامل اطلاعات همه سنسورها در همه مقاطع زمانی هستند. اطلاعات سنسورها در مقطع زمانی t را می‌توان با بردار $x_t = (x_t^1, x_t^2, x_t^3, \ldots, x_t^N)$ نمایش داد. خروجی این سیستم باید تشخیص فعالیت متناسب با اطلاعات سنسورها در لحظه t باشد که با y_t نمایش داده می‌شود. در این سیستم حداکثر ۱۲ فعالیت خواهیم داشت و بنابراین وظیفه الگوریتم دسته‌بند تشخیص این فعالیت است. جدول ۴ تعداد فعالیت‌ها را در هر یک از ۶ دیتاست نمایش می‌دهد.

داده‌های جریانی خام تولیدشده با استفاده از سنسورها می‌تواند به صورت مستقیم و یا با تغییرات و انجام پردازش‌هایی مورد استفاده قرار گیرد. به منظور تقویت فضای ویژگی‌ها در این پژوهش، سه نوع نمایش برای داده‌های سنسورها در نظر گرفته شده که نتایج نیز بر اساس همین سه مورد می‌باشند: ۱) نمایش خام سنسور: در این نوع نمایش وضعیت سنسور زمانی که فعال است برابر یک و در غیر این صورت برابر صفر است. ۲) نمایش نقطه تغییر: در این نوع نمایش، زمانی که سنسور تغییر وضعیت می‌دهد ویژگی برابر یک و در غیر این صورت برابر صفر خواهد بود. ۳) نمایش آخرین سنسور: در این نوع نمایش همواره مقدار ویژگی آخرین سنسوری که فعال شده است برابر یک و برای سایرین برابر صفر خواهد بود. با توجه به این نوع نمایش‌ها روش پیشنهادی به همراه چندین روش شناخته‌شده دیگر بر روی مجموعه داده‌ها اجرا شده‌اند که نتایج آن در جدول ۵ آورده شده است. لازم به ذکر است که به دلیل نامتوازن‌بودن مجموعه داده‌ها، برای مقایسه نتایج از معیار $F1-Measure$ استفاده شده است.

۶- نتیجه‌گیری

در این مقاله یک الگوریتم افزایشی برای الگوریتم جدید ماشین بردار پشتیبان دوقلوی مربعات حداقلی فازی ارائه شد. الگوریتم پیشنهادی شامل دو نسخه برخط و شبه برخط است. در ارائه الگوریتم جدید به دنبال تعیین میزان اطلاعات هر نمونه جدید بودیم که این امر در این مقاله با استفاده از چندین معیار از جمله میزان چگالی، میزان مشابهت و میزان

1. Accuracy



شکل ۶: مقایسه الگوریتم پیشنهادی با سایر روش‌ها بر اساس امتیاز F۱.

جدول ۴: درصد تعداد فعالیت‌ها در دیتاست‌های مربوط.

Activity	KasterenA	KasterenB	KasterenC	OrdonezA	OrdonezB
Leaving	۴۹٫۷۴	۵۴٫۳۶	۴۶٫۲۷	۸٫۳۲	۱۷٫۴۱
Toileting	۰٫۶۵	۰٫۲۷	۰٫۶۲	۰٫۷۶	۰٫۵۵
Showering	۰٫۷۰	۰٫۶	۰٫۶	۰٫۵۴	۰٫۲۴
Sleeping	۳۳٫۴۲	۳۳٫۵۳	۲۸٫۴۶	۳۹٫۱	۳۵٫۵۸
Breakfast	۰٫۲۳	۰٫۵۲	۰٫۶۲	۰٫۶۳	۰٫۵
Dinner	۱٫۰	۰٫۴۲	۱٫۲۶	۰	۰٫۳۸
Drink	۱۴٫۱۲	۰٫۷	۰٫۱۱	۰	۰
Idle/Unlabeled	۰	۱۰٫۱۲	۲۱٫۹۷	۵٫۶۱	۱۱٫۷۳
Lunch	۰	۰	۰	۱٫۵۹	۱٫۳
Snack	۰	۰	۰	۰٫۵	۱٫۳۳
Spare time/TV	۰	۰	۰	۴۲٫۷	۲۸٫۹۸
Grooming	۰	۰	۰	۰٫۷۳	۱٫۴۲

جدول ۵: مقایسه الگوریتم‌های گوناگون بر روی مجموعه داده‌های معرفی‌شده. نتایج بر اساس F-MEASURE و به صورت درصد بیان شده‌اند. نتایج سایر الگوریتم‌ها از [۳۵] آورده شده است.

مجموعه داده	نوع نمایش داده‌ها	SVM	k-NN	MLP	Inc-FLSTSVM
KasterenA	خام	۵۴ ± ۱۰	۵۵ ± ۱۱	۵۱ ± ۱۱	۵۸ ± ۷
	نقطه تغییر	۵۲ ± ۱۱	۵۴ ± ۱۰	۵۰ ± ۱۱	۵۹ ± ۸
	آخرین سنسور	۶۱ ± ۱۱	۶۱ ± ۱۱	۶۱ ± ۱۱	۶۲ ± ۹
KasterenB	خام	۵۷ ± ۱۰	۵۴ ± ۱۱	۵۰ ± ۱۰	۵۹ ± ۹
	نقطه تغییر	۵۶ ± ۶	۵۸ ± ۶	۵۳ ± ۵	۶۰ ± ۷
	آخرین سنسور	۶۵ ± ۱۲	۶۵ ± ۱۲	۶۵ ± ۱۲	۶۵ ± ۶
KasterenC	خام	۴۹ ± ۹	۴۸ ± ۱۰	۵۰ ± ۱۰	۵۳ ± ۸
	نقطه تغییر	۴۶ ± ۹	۴۶ ± ۷	۴۷ ± ۸	۴۹ ± ۸
	آخرین سنسور	۶۷ ± ۷	۶۷ ± ۷	۶۷ ± ۷	۷۱ ± ۱۰
OrdonezA	خام	۷۸ ± ۷	۷۷ ± ۵	۷۷ ± ۷	۸۲ ± ۸
	نقطه تغییر	۵۳ ± ۷	۵۳ ± ۷	۵۲ ± ۷	۵۵ ± ۶
	آخرین سنسور	۶۶ ± ۸	۶۵ ± ۸	۶۷ ± ۸	۷۲ ± ۵
OrdonezB	خام	۶۹ ± ۷	۶۹ ± ۷	۶۸ ± ۶	۷۳ ± ۸
	نقطه تغییر	۵۰ ± ۵	۵۲ ± ۷	۵۰ ± ۷	۵۷ ± ۶
	آخرین سنسور	۷۳ ± ۷	۷۳ ± ۷	۷۲ ± ۷	۷۷ ± ۹

نزدیکی به ابرصفحه‌ها اندازه‌گیری شد. الگوریتم ارائه‌شده بر روی انواع مجموعه داده‌ها اجرا شد که نتایج آزمایشگاهی از کارایی بالای این الگوریتم نسبت به نمونه‌های مشابه آن در نسخه‌های متفاوت الگوریتم SVM و همچنین سایر الگوریتم‌ها حکایت دارد. الگوریتم ارائه‌شده به دلیل در نظر گرفتن مفاهیم فازی این قدرت را دارد که به نمونه‌های جدیدتر اهمیت بیشتری بدهد. همچنین به دلیل وجود دو ابرصفحه و نیز معادلات خطی در این الگوریتم، سرعت آن نسبت به الگوریتم‌های SVM و TWSVM به مراتب بیشتر است.

مراجع

[1] D. Bhattacharya and M. Mitra, Analytics on Big Fast Data Using Real Time Stream Data Processing Prchitecture, EMC Corporation, 2013.

[2] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. of the 5th Annual Workshop on Computational Learning Theory*, pp. 144-152, Pittsburgh, PA, USA, 27-29 Jul. 1992.

[3] R. T. Rockafellar, "Lagrange multipliers and optimality," *SIAM Review*, vol. 35, no. 2, pp. 183-238, 1993.

[4] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural Processing Letters*, vol. 9, no. 3, pp. 293-300, Jun. 1999.

[5] J. A. K. Suykens, T. Van. Gestel, J. De Brabanter, B. De Moor, and J, Vandewalle, *Least Squares Support Vector Machines*, World Scientific, 2002.

[6] R. Khemchandani and S. Chandra, "Twin support vector machines for pattern classification," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, no. 5, pp. 905-910, May 2007.

[7] M. A. Kumar and M. Gopal, "Least squares twin support vector machines for pattern classification," *Expert Systems with Applications*, vol. 36, no. 4, pp. 7535-7543, May 2009.

[8] J. S. Sartakhti, H. Afrabandpey, and N. Ghadiri, "Fuzzy least squares twin support vector machines," *Engineering Applications of Artificial Intelligence*, vol. 85, pp. 402-409, Oct. 2019.

[9] V. Losing, B. Hammer, and H. Wersing, "Incremental on-line learning: a review and comparison of state of the art algorithms," *Neurocomputing*, vol. 275, pp. 1261-1274, Jan. 2018.

[10] R. KhemchandaniJayadeva, and S. Chandra, "Incremental twin support vector machines," in *Modeling, Computation and Optimization,* in S. K. Neogy, A. K. Das, and R. B. Bapat (eds.), pp. 263-272, World Scientific, 2009.

[11] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," in *Proc. of the 13th Int Conf. on Neural Information Processing Systems*, pp. 388-394, Denver, CO, USA, 1-1 Jan. 2000.

[12] Y. Hao and H. Zhang, "A fast incremental learning algorithm based on twin support vector machine," in *Proc. IEEE 7th Int. Symp. on Computational Intelligence and Design,* , vol. 2, pp. 92-95, Hangzhou, China, 13-14 Dec. 2014.

[13] F. Alamdar, S. Ghane, and A. Amiri, "On-line twin independent support vector machines," *Neurocomputing*, vol. 186, no. C, pp. 8-21, Apr. 2016.

[14] G. Fung and O. L. Mangasarian, "Incremental support vector machine classification," in *Proc. of the SIAM Int. Conf. on Data Mining,* pp. 247-260, Arlimgton, VA, USA, 11-13 Apr. 2002.

[15] A. Tveit, M. L. Hetland, and H. Engum, "Incremental and decremental proximal support vector classification using decay coefficients," in *Proc. of the Int. Conf. on Data Warehousing and Knowledge Discovery,* pp. 422-429, Prague, Czech Republic, 3-5 Sept. 2003.

[16] A. R. Mello, M. R. Stemmer, and A. L. Koerich, "Incremental and decremental fuzzy bounded twin support vector machine," *Information Sciences*, vol. 526, pp. 20-38, Jul. 2020.

[17] J. Xu, C. Xu, B. Zou, Y. Y. Tang, J. Peng, and X. You, "New incremental learning algorithm with support vector machines," *IEEE Trans. on Systems, Man, and Cybernetics, Systems*, vol. 49, no. 11, pp. 2230-2241, Nov. 2018.

[18] I. A. Lawal, "Incremental SVM learning," Studies in Big Data Learning from Data Streams in Evolving Environments, pp. 279-296, 2019.

[19] W. Xie, S. Uhlmann, S. Kiranyaz, and M. Gabbouj, "Incremental learning with support vector data description," in *Proc. IEEE 22nd Int. Conf. on Pattern Recognition,* pp. 3904-3909, Stockholm, Sweden, 24-28 Aug. 2014.

[20] Z. Zhu, X. Zhu, Y. F. Guo, and X. Xue, "Transfer incremental learning for pattern classification," in *Proc. of the 19th ACM International Conf. on Information and Knowledge Management*, pp. 1709-1712, Toronto, Canada, 26-30 Oct. 2010.

[21] G. Cauwenberghs and T. Poggio, "Incremental and decremental support vector machine learning," *Advances in Neural Information Processing Systems*, vol. ???, no. ???, pp. 409-415, ???. 2001.

[22] H. Duan, X. Shao, W. Hou, G. He, and Q. Zeng, "An incremental learning algorithm for Lagrangian support vector machines," *Pattern Recognition Letters*, vol. 30, no. 15, pp. 1384-1391, 1 Nov. 2009.

[23] H. Galmeanu, L. M. Sasu, and R. Andonie, "Incremental and decremental SVM for regression," *International J. of Computers Communications & Control*, vol. 11, no. 6, pp. 755-775, Dec. 2016.

[24] H. Galmeanu and R. Andonie, "A multi-class incremental and decremental SVM approach using adaptive directed acyclic graphs," in *Proc. IEEE Int Conf. on Adaptive and Intelligent Systems,* pp. 114-119, Klagenfurt, Austria, 24-26 Sept. 2009.

[25] J. Wang, D. Yang, W. Jiang, and J. Zhou, "Semisupervised incremental support vector machine learning based on neighborhood kernel estimation," *IEEE Trans. on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 10, pp. 2677-2687, Oct. 2017.

[26] M. S. Chen, T. Y. Ho, and D. Y. Huang, "Online transductive support vector machines for classification," in *Proc. IEEE Int. Conf. on Information Security and Intelligent Control,* pp. 258-261, Yunlin, Taiwan, 14-16 Aug. 2012.

[27] P. Rai, H. Daume, and S. Venkatasubramanian, "Streamed learning: one-pass SVMs," in *Proc. 21st Int. Joint Conf. on Artificial Intelligence*, pp. 1211-1216, Pasadena, CA, USA, 11-17 Jul. 2009.

[28] N. A. Syed, S. Huan, L. Kah, and K. Sung, "Incremental learning with support vector machines," in *Proc. IEEE Int. Conf. on Data Mining*, San Jose, CA, USA, 29 Nov.-2 Dec. 2001.

[29] F. Orabona, C. Castellini, B. Caputo, L. Jie, and G. Sandini, "On-line independent support vector machines," *Pattern Recognition*, vol. 43, no. 4, pp. 1402-1412, Apr. 2010.

[30] L. Ralaivola and F. d'Alche-Buc, "Incremental support vector machine learning: a local approach," in *Proc. Int. Conf. on Artificial Neural Networks,* pp. 322-330, Vienna, Austria, 21-25 Aug. 2001.

[31] M. N. Kapp, R. Sabourin, and P. Maupin, "Adaptive incremental learning with an ensemble of support vector machines," in *Proc. IEEE 20th Int. Conf. on Pattern Recognition,* pp. 4048-4051, Istanbul, Turkey, 23-26 Aug. 2010.

[32] C. Domeniconi and D. Gunopulos, "Incremental support vector machine construction," in *Proc. IEEE Int. Conf. on Data Mining*, pp. 589-592, San Jose, CA, USA, 29 Nov.-2 Dec. 2001.

[33] A. R. de Mello, M. R. Stemmer, and A. L. Koerich, *Incremental and Decremental Fuzzy Bounded Twin Support Vector Machine*, arXiv preprint arXiv:1907.09613, 2019.

[34] "Activities of Daily Living (ADLs) Recognition Using Binary Sensors Data Set," ed, 2013.

[35] F. Ordonez, P. De Toledo, and A. Sanchis, "Activity recognition using hybrid generative/discriminative models on home environments using binary sensors," *Sensors*, vol. 13, no. 5, pp. 5460-5477, 2013.

جواد سلیمی سرتختی در سال ۱۳۸۸ مدرک کارشناسی مهندسی کامپیوتر خود را از دانشگاه کاشان و در سال ۱۳۹۰ مدرک کارشناسی ارشد مهندسی نرم‌افزار خود را از دانشگاه تربیت مدرس دریافت نمود. از سال ۱۳۹۰ الی ۱۳۹۱ نامبرده به عنوان طراح ارشد در شرکت رایان اقتصاد نوین به کار مشغول بود و پس از آن به دوره دکترای مهندسی کامپیوتر در دانشگاه صنعتی اصفهان وارد گردید و در سال ۱۳۹۶ موفق به اخذ درجه دکترا در مهندسی کامپیوتر از دانشگاه مذکور گردید. دکتر سلیمی از سال ۱۳۹۶ در دانشکده مهندسی کامپیوتر دانشگاه کاشان مشغول به فعالیت گردید و اینک نیز عضو هیأت علمی این دانشکده می‌باشد. زمینه‌های علمی مورد علاقه نامبرده متنوع بوده و شامل موضوعاتی مانند یادگیری ماشین، یادگیری عمیق، تئوری بازی‌ها و طراحی مکانیزم و بلاک‌چین می‌باشد.

سلمان گلی بیدگلی تحصیلات خود را در مقطع کارشناسی مهندسی کامپیوتر در سال ۱۳۸۵ در دانشگاه کاشان و کارشناسی ارشد و دکتری مهندسی کامپیوتر- معماری سیستم های کامپیوتری به‌ترتیب در سال‌های ۱۳۸۹ و ۱۳۹۶ از دانشگاه اصفهان به پایان رسانده است و هم‌اکنون استادیار دانشکده مهندسی برق و کامپیوتر دانشگاه کاشان می‌باشد. زمینه‌های تحقیقاتی مورد علاقه ایشان عبارتند از: شبکه-های کامپیوتری، قابلیت اطمینان، اینترنت اشیا و بلاک‌چین می‌باشد.