1. راهکارهای پیشنهادی بهبود دقت در تعیین موقعیت
   1. مقدمه

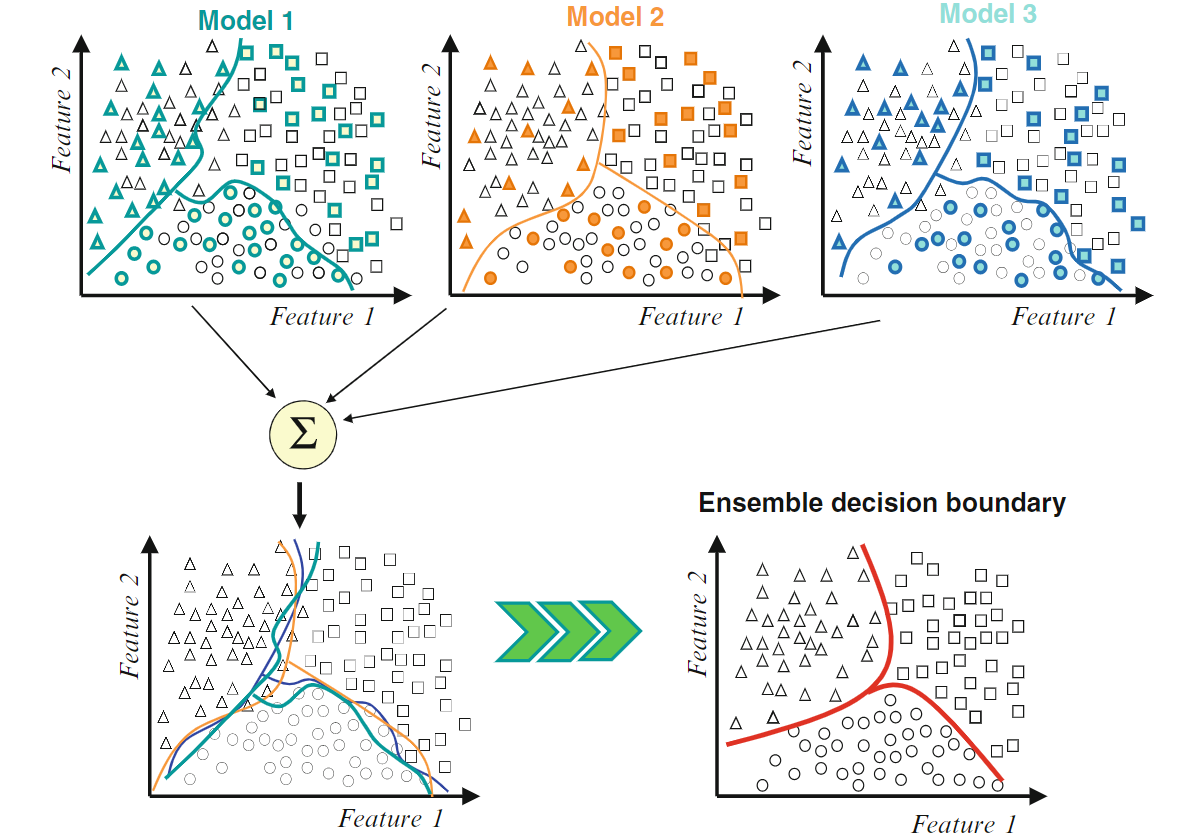
در فصل قبل انواع روش‌های موقعیت‌یابی با استفاده از الگوریتم‌های یادگیری ماشین بیان شد. روش‌های پایه با وجود سادگی در پیاده‌سازی، قادر به تخمین موقعیت با دقت بالا نیستند. به همین منظور، روش‌های یادگیری ماشین برای بهبود دقت موقعیت‌یابی مکان‌های سرپوشیده به کار گرفته شده‌اند. همچنین بر اساس فناوری‌های ارتباطی بی‌سیم، در عمل، عمدتاً از شبکه‌های بی‌سیم Wi-Fi و BLE استفاده می‌شود. در میان منابع پژوهشی، فناوری Wi-Fi پرکاربردترین فناوری مورداستفاده در موقعیت‌یابی مکان‌های سرپوشیده است. دو رویکرد برای موقعیت‌یابی با استفاده از Wi-Fi وجود دارد. رویکرد اول، استفاده از مدل انتشاری سیگنال در تخمین RSSI برای محاسبه فاصله و رویکرد دوم، ساخت یک نقشه اثر انگشت Wi-Fi و استفاده از سیگنال Wi-Fi برای تخمین موقعیت است. روش اثر انگشت به طور قابل‌توجهی در ارتقای موقعیت‌یابی مکان سرپوشیده در حال توسعه است.

بنا به تعریف، یک سیستم موقعیت‌یاب باید موقعیت هدف را با دقت مناسب و به‌صورت بلادرنگ تخمین بزند؛ بنابراین انتخاب یک روش یادگیری ماشین مناسب در تخمین موقعیت که بتواند الزامات سیستم موقعیت‌یاب را برآورده کند ضروری است. در این فصل راهکار‌های پیشنهادی الگوریتم‌های یادگیری ماشین در بهبود دقت موقعیت‌یابی با درنظرگیری امکان پیاده‌سازی و سرعت اجرا ارائه می‌شود. این راهکارها برای یک سیستم موقعیت‌یاب مبتنی بر نقشه اثر انگشت Wi-Fi و استفاده از شاخص RSSI در تشکیل نقشه اثر انگشت ارائه شده‌اند.

* 1. روش‌های یادگیری گروهی

بسیاری از پژوهش‌های انجام شده در موقعیت‌یابی سرپوشیده با استفاده از نقشه اثر انگشت Wi-Fi، بر اساس روش kNN شکل‌گرفته‌اند. ازآنجاکه برخی از نقاط دسترسی در محیط می‌تواند از یک نقطه مشخص دور باشد، آن نقاط دسترسی در یک موقعیت مشخص ممکن است مشاهده نشود و بردار RSSI در برخی از مکان‌ها شامل سیگنال‌های دریافت شده توسط همه نقاط دسترسی نباشد. همچنین ممکن است برخی نقاط مرجع در بردار شامل RSSI مشابه باشد. روش kNN همه نقاط مرجع روی نقشه اثر انگشت را بدون لحاظ‌کردن این نکته در نظر می‌گیرد. از سوی دیگر، همسایه‌هایی که از طریق الگوریتم kNN یافت می‌شود، ممکن است فراتر از اندازه‌گیری‌های ممکن در محیط باشد. زیرا تضعیف سیگنال هر نقطه دسترسی نه‌تنها به فاصله آن مرتبط است، بلکه تحت‌تأثیر بسیاری از عوامل محیطی نیز قرار می‌گیرد. این موضوع، باعث پدیدارشدن اثر حداقل فاصله سیگنالی بردار RSSI و نقاط مرجع می‌شود. باتوجه‌به محدودیت‌های بیان شده در روش kNN، استفاده از الگوریتم‌های یادگیری گروهی گزینه‌ای ایدئال برای جایگزینی با این روش‌ها است.

فرضیه استفاده از سیستم‌های تصمیم‌گیری بر اساس رأی‌گیری در زندگی روزمره ما اساساً با کاربرد آنها در هوش محاسباتی متفاوت نیست. ما اغلب پیش از تصمیم‌گیری با دیگران مشورت می‌کنیم؛ زیرا گذشته و دقت تصمیم‌گیری هر یک از تصمیم‌گیرندگان فردی متفاوت است. هر خطای طبقه‌بندی از دو مؤلفه تشکیل شده است که می‌بایست آنها را کنترل کرد: بایاس[[1]](#footnote-2)، دقت طبقه‌بندی‌کننده؛ و واریانس[[2]](#footnote-3)، حساسیت طبقه‌بندی‌کننده هنگام آموزش روی مجموعه‌دادگان مختلف. اغلب این دو مؤلفه رابطه‌ای متقابل دارند: طبقه‌بندی‌کننده‌هایی با بایاس پایین تمایل به واریانس بالا دارند و برعکس؛ بنابراین، هدف سیستم‌های طبقه‌بندی‌کننده ایجاد مدل با بایاس نسبتاً ثابت (یا مشابه) و سپس ترکیب خروجی‌های آنها، مثلاً با میانگین‌گیری، برای کاهش واریانس است.



کاهش واریانس با استفاده از الگوریتم‌های یادگیری گروهی [1]

کاهش واریانس به معنای کاهش نوسانات در یک مقدار است. این کار می‌تواند با میانگین‌گیری مقادیر مختلف انجام شود. در زمینه طبقه‌بندی، کاهش واریانس می‌تواند به بهبود دقت طبقه‌بندی کمک کند. این به این دلیل است که طبقه‌بندی‌کننده‌ها اغلب در طبقه‌بندی نمونه‌های جدید اشتباه می‌کنند. با میانگین‌گیری خروجی‌های چند طبقه‌بندی‌کننده مختلف، می‌توان خطاهای آنها را کاهش داد. روش‌های مختلفی برای ترکیب خروجی‌های طبقه‌بندی‌کننده‌ها وجود دارد. میانگین‌گیری تنها یکی از این روش‌ها است. ‏شکل (3˗1) نشان می‌دهد که چگونه میانگین‌گیری خروجی های دو طبقه بندی کننده با واریانس بالا می تواند واریانس خروجی را کاهش دهد.

الگوریتم‌های یادگیری گروهی در موقعیت‌یابی می‌تواند چالش‌های موجود در فرایند موقعیت‌یابی، از جمله چندمسیره شدن سیگنال و شرایط NLOS، را نیز بهبود بخشد. این الگوریتم‌ها با بهره‌گیری از تحلیل گروهی چندین مدل از داده‌ها، می‌تواند الگوهای پیچیده چندمسیره شدن را تشخیص داده و با ترکیب اطلاعات دقت موقعیت‌یابی را افزایش دهد. علاوه بر این، الگوریتم‌های یادگیری گروهی به کاهش اثرات تضعیف RSSI نیز می‌پردازد. در شرایطی که سیگنال‌ها به دلیل موانع موجود، تداخلات سیگنال در مسیر انتقال دچار تضعیف شوند، این الگوریتم‌ها با تجمیع تجربه‌های مختلف مدل‌های یادگیری ماشین، می‌توانند اطلاعات صحیح‌تری از قدرت سیگنال به دست آورده و اثر تضعیف را به حداقل برساند.

* + 1. توسعه الگوریتم‌های یادگیری گروهی

بسیاری از بررسی‌ها به کار Sheela و Dasarathy در سال 1979 میلادی به عنوان یکی از اولین نمونه‌های الگوریتم‌های یادگیری گروهی، با ایده‌های آن‌ها در مورد تقسیم فضای ویژگی‌ها و استفاده از چندین طبقه‌بندی‌کننده اشاره می‌کنند[2]. با این حال، نخستین بار در [3] روشی به نام Boosting معرفی شد و نشان داد که در یک مسئله طبقه‌بندی دو کلاسه، یک طبقه‌بندی‌کننده قوی با خطای کم می‌تواند از مجموعه ای از طبقه‌بندی کننده هایی که خطای هر یک از آن‌ها از خطای یک طبقه بندی کننده حدس تصادفی بیشتر باشد، ساخته شود. نظریه Boosting پایه و اساس الگوریتم یادگیری گروهی بعدی، AdaBoost را فراهم کرد که از محبوب ترین الگوریتم های یادگیری گروهی است و نظریه را به مسائل چند کلاسه و تقریب تابع گسترش می‌دهد[4].

به دلیل موفقیت در این کارهای پایه، از آن به بعد تحقیقات در الگوریتم‌های یادگیری گروهی گسترش‌یافته است و این الگوریتم‌ها تحت نام‌های مختلف ظهور پیدا کردند. Bagging [5]، جنگل تصادفی[[3]](#footnote-4) (گروهی از درختان تصمیم)، سیستم‌های طبقه‌بندی‌کننده مرکب، ترکیب ماهرها (MoE) [6]، روش تعمیم پشته‌سازی [7]، XGBoost [8] و بسیاری دیگر از الگوریتم‌ها معرفی شده‌اند.

* + 1. اجزای الگوریتم یادگیری گروهی

لازم است تا سه رویه برای تشکیل یک سیستم یادگیری گروهی در پیش گرفته شود. این سه روش به ترتیب عبارت است از نمونه‌برداری و انتخاب داده، آموزش یادگیرندگان ضعیف و درنهایت ترکیب یادگیرندگان که در ادامه به آن پرداخته می‌شود.

* نمونه‌برداری و انتخاب داده

مواجهه با خطاهای گوناگون به‌ازای هر نمونه در سیستم‌های یادگیری گروهی از اهمیت بالایی برخوردار است. چراکه، اگر همه مدل‌ها به‌ازای هر نمونه خروجی یکسانی داشته باشند، هیچ دانشی از ترکیب آن‌ها به دست نمی‌آید. ازاین‌رو، در تشکیل یک سیستم یادگیری گروهی نیاز به تنوع در تصمیم‌گیری اعضای گروه، به‌ویژه زمانی که اعضا در تصمیم‌گیری با خطا مواجه می‌شوند، است. این موضوع اهمیت **تنوع** داده را در این سیستم‌ها نشان می‌دهد.

با استفاده از رویکردهای متفاوت، می‌توان به تنوع داده در یادگیرندگان رسید. رایج‌ترین رویکرد استفاده از مجموعه‌دادگان آموزشی متفاوت است که در ‏شکل (3˗1) نیز نشان داده شده است. رویکردهای مختلف در انتخاب داده منجر به پدیدآمدن الگوریتم‌های گروهی متفاوت می‌شود. به‌عنوان‌مثال انتخاب با جای‌گذاری دادگان آموزشی، روش Bagging را نتیجه می‌دهد، درحالی‌که نمونه‌برداری با توزیعی که به نفع نمونه‌های اشتباه طبقه‌بندی شده‌اند، هسته اصلی الگوریتم‌های Boosting است. از سوی دیگر، می‌توان از زیرمجموعه‌های مختلفی از ویژگی‌های موجود برای آموزش هر یادگیرنده استفاده کرد که منجر به روش‌های زیر فضای تصادفی[[4]](#footnote-5) می‌شود. علی‌رغم اهمیت تنوع داده در عملکرد یادگیرندگان که به‌خوبی اثبات شده است، اما رابطه‌ای صریح بین تنوع و دقت الگوریتم گروهی شناسایی نشده است.

* آموزش یادگیرندگان

استراتژی مورداستفاده برای آموزش اعضای گروه، هسته هر سیستم یادگیری گروهی است. الگوریتم‌های رقابتی متعددی برای آموزش طبقه‌بندی‌کننده‌های گروه توسعه داده شده‌اند. بااین‌حال، Bagging (و الگوریتم‌های مرتبط مانند جنگل تصادفی)، Boosting (و انواع آن)، تعمیم پشته‌سازی به‌عنوان رایج‌ترین رویکردهای به کار گرفته شده حساب می‌شود. این رویکردها در بیان می‌شود.

* ترکیب یادگیرندگان

مرحله نهایی در سیستم‌های مبتنی بر مجموعه شامل ترکیب پیش‌بینی‌های یادگیرندگان است. استراتژی ترکیب به نوع یادگیرندگان مورداستفاده بستگی دارد. برای یادگیرندگان گسسته مانند SVM، رأی اکثریت رایج است. برای طبقه‌بندی‌کننده‌های پیوسته؛ مانند شبکه عصبی، ترکیب‌کننده‌های حسابی یا الگوهای تصمیم‌گیری پیچیده گزینه‌های مناسبی هستند. بسیاری از ترکیب‌کننده‌ها را می‌توان بدون آموزش اضافی استفاده کرد، درحالی‌که ترکیب‌های پیچیده‌تر ممکن است نیاز به یک مرحله اضافی داشته باشند.

ابتدا فرض می‌شود که فقط برچسب‌ها از خروجی یادگیرندگان در دسترس است. تصمیم یادگیرنده را به‌صورت که و و همچنین تعداد یادگیرندگان و تعداد برچسب ها است. اگر یادگیرنده (یا فرضیه) که با نمایش داده می شود، طبقه را انتخاب کند، آنگاه و در غیر این صورت برابر با صفر خواهد بود. طبق تعریف فوق قوانین ترکیبی که برای یادگیرندگان ارائه شده است در ادامه توضیح داده می‌شود.

1. **رأی اکثریت[[5]](#footnote-6):** رأی اکثریت دارای سه نوع است بسته به این که آیا تصمیم گروه (1) آن چیزی است که همه یادگیرندگان با آن موافق هستند (اتفاق نظر آرا) (2) پیش بینی شده توسط حداقل بیش از نیمی از تعداد یادگیرندگان (اکثریت ساده) یا (3) بیشترین مجموه تعداد آرا فارغ از آنکه بیش از 50 درصد موافق باشند (تجمیع رأی). در صورتی که نوع رأی اکثریت مشخص نشده باشد، این ترکیب به تجمیع رأی اشاره دارد و به زبان ریاضی به صورت رابطه ‏(3˗1) بیان می شود.

طبقه انتخاب می شود اگر

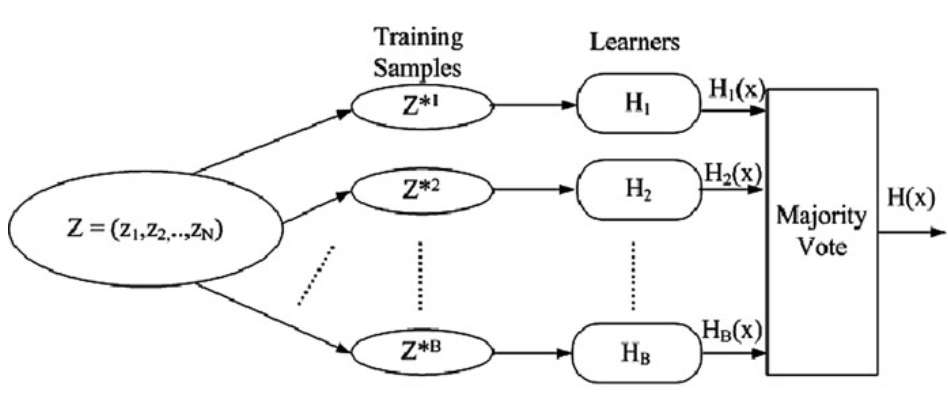
1. **رأی اکثریت وزن‌دار:** اگر دلیلی بر این باور داشته باشیم که برخی از یادگیرندگان بیشتر از بقیه صحیح هستند، وزن دادن به تصمیمات آن یادگیرندگان می‌تواند عملکرد کلی را در مقایسه با اکثریت آرا بهبود بخشد. می‌توانیم وزن را به یادگیرنده بر اساس عملکرد کلی تخمین آن اختصاص دهیم. گروه با توجه به رأی اکثریت آرا کلاس را انتخاب می کند اگر

یعنی کل آرای وزن‌دار دریافت شده توسط طبقه بیشتر از تمام آرای وزن دار دریافت شده توسط سایر طبقه ها باشد. به طور کلی، وزن های رأی به گونه ای نرمال می شوند که مجموع آن ها برابر با یک باشد.

مجموعه‌ای غنی از الگوریتم‌های گروهی در چند سال گذشته توسعه‌ یافته است. بااین‌وجود، برخی از این الگوریتم‌های منتخب و به‌خوبی تثبیت شده هستند که قابلیت‌های آن‌ها نیز به طور گسترده آزمایش و گزارش شده است. در ادامه برخی از برجسته‌ترین الگوریتم‌های یادگیری گروهی ارائه شده است و در فصل بعدی نتایج حاصل از شبیه‌سازی آن‌ها مقایسه شده است.

* + 1. روش Bagging

Bagging کوتاه شده عبارت (Bootstrap Aggregation) و به معنای «تجمیع خود راه‌انداز» است. این روش توسط Breiman در سال 1996 معرفی شده است و یکی از اولین، ساده ترین و در عین حال مؤثر ترین الگوریتم های یادگیری گروهی است [5]. با توجه مجموعه دادگان داده شده ، روش Bagging به طور ساده یادگیرنده مستقل که هر کدام با انتخاب و جایگذاری نمونه از مجموعه آموزش می‌بیند، ترکیب می‌کند. در این روش تنوع در داده با تغییرات انتخاب و جایگذاری داده و همچنین استفاده از یادگیرندگان ضعیف که مرز تصمیم گیری آن ها با تغییرات نسبتاً کوچک در دادگان تغییر می‌کند، تضمین می‌شود. طبقه کنندگان خطی مانند درخت تصمیم، SVM خطی و شبکه های عصبی تک لایه کاندیدهای خوبی به عنوان یادگیرندگان ضعیف هستند. یادگیرندگان در نهایت با روش اکثریت ساده ترکیب میشوند. ‏شکل (3˗2) نمای کلی از روش Bagging را برای مسائل طبقه بندی نمایش می دهد.



رویکرد Bagging در مسائل طبقه‌بندی [1]

* جنگل تصادفی

یکی از نسخه‌های خلاقانه روش Bagging، الگوریتم جنگل تصادفی است که اساس آن ترکیب مجموعه از درختان تصمیم آموزش‌دیده با مکانیزم Bagging است. جنگل تصادفی توسط Breiman معرفی شد [5]. جنگل تصادفی را می‌توان هم برای متغیرهای گسسته که در موقعیت‌یابی می‌تواند شرایط محیطی منظور شود، و هم برای متغیرهای پیوسته که در موقعیت‌یابی می‌تواند RSS سیگنال ارتباطی بی‌سیم باشد، استفاده کرد. به طور مشابه، متغیرهای پیش‌بینی‌شده نیز می‌تواند گسسته یا پیوسته باشد.

ازنقطه‌نظر محاسباتی نیز جنگل تصادفی عملکرد خوبی را دارد، چراکه هم در آموزش و هم در آزمایش نسبتاً سریع هستند، فقط به یک یا دو پارامتر بستگی دارد، می‌تواند برای مسائلی همچون موقعیت‌یابی که ابعاد بردار ویژگی بالا است استفاده شود و در نهایت این که می‌توان آن را به‌صورت موازی اجرا کرد.

عبارت «تصادفی» علاوه بر انتخاب تصادفی دادگان، به انتخاب تصادفی متغیرهای ویژگی در هر گره درخت تصمیم اشاره دارد. این به این معناست که هنگام انتخاب ویژگی در هر گره درخت، ویژگی از ویژگی در بردار انتخاب می‌شود. این انتخاب تصادفی بردار ویژگی ها باعث ساخت درختان متمایز از هم می شود و با در نظرگیری روش انتخاب و جایگذاری دادگان، بیشترین تنوع به وجود می‌آید. شبه‌کد الگوریتم جنگل تصادفی در الگوریتم 1 بیان شده است.

Breiman در [5] برای مسائل طبقه‌بندی، رشد درختان را تا زمانی که به یک گره خالص (برگ) برسد، پیشنهاد کرده است. اما در [9] که پژوهشی جدیدتر است، کنترل عمق درختان پیشنهاد شده است.

|  |
| --- |
| **الگوریتم 1: جنگل تصادفی** |
| فرض کنید به مجموعه دادگان آموزش اشاره دارد که در آن که تعداد ویژگی ها است. به ازای تا (تعداد یادگیرندگان ضعیف) انجام دهید:   1. *یک انتخاب با جای‌گذاری با اندازه از انجام دهید.* 2. *با استفاده از نمونه‌برداری با جای‌گذاری انتخاب شده به‌عنوان دادگان آموزشی، درخت تصمیم را تشکیل دهید:*    1. *با تمام ویژگی‌ها در یک گره شروع کن.*    2. *گام‌های زیر را برای گره‌های تقسیم نشده تکرار کن تا زمانی که شرط توقف درخت (گره با یک ویژگی/عمق درخت) ارضا شود:*       1. *ویژگی را به‌صورت تصادفی از ویژگی در دسترس انتخاب کنید.*       2. *بهترین ویژگی تقسیم در میان ویژگی انتخاب شده در گام (*i*) را پیدا کنید.*       3. *گره را با استفاده از ویژگی انتخابی در گام (*ii*) تقسیم کنید.*   *برای پیش‌بینی در نقطه جدید :*    *که در آن تخمین درخت برای ورودی و تابع به صورت زیر تعریف می‌شود:* |

جنگل تصادفی به این شهرت دارد که با تنظیم اولیه کاملاً خوب کار می‌کند. اما برای کاربرد موقعیت‌یابی که نیازمند سرعت بالا در محاسبات و سادگی پیاده‌سازی است. نیاز به تنظیم سه پارامتر وجود دارد:

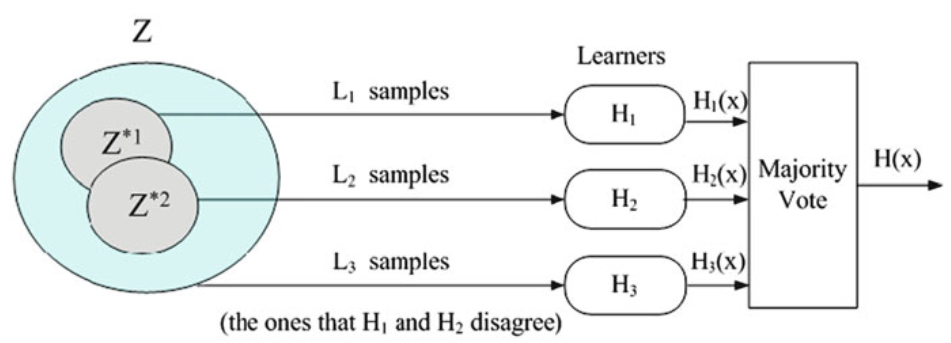
* ، تعداد ویژگی‌هایی که به‌صورت تصادفی در هر گره انتخاب می‌شود.
* ، تعداد درختان موجود در جنگل
* اندازه درخت که یا با کمترین اندازه برگ مشخص می‌شود یا بیشینه تعداد گره‌های نهایی درختان،

در مسائل طبقه‌بندی مقدار پیشنهاد شده است [10]. برای تنظیم پارامترهای و اندازه درخت، در این پژوهش روش الگوریتم ژنتیک را پیشنهاد شده است که در بخش ‏0 معرفی می‌شود.

* + 1. روش Boosting

Boosting کار ابتدایی Schapire در [3] معرفی شد. این روش، یک روش تکرارشونده برای تولید یک‌ طبقه‌بندی‌کننده قوی است و قابلیت رسیدن به خطای آموزش پایین، با گروهی از یادگیرندگان ضعیفی که کمی بهتر از حدس تصادفی عمل می‌کنند را دارد. این روش با اینکه مجموعه یادگیرندگان ضعیف را با استفاده از رأی اکثریت ترکیب می‌کند، اما از یک منظر مهم با روش Bagging تفاوت دارد. در روش Bagging انتخاب نمونه‌ها برای آموزش هر یک از یادگیرندگان ضعیف از طریق انتخاب بدون جای‌گذاری صورت می‌گیرد، به این معنی که هر نمونه شانس برابری برای قرارگرفتن در هر مجموعه‌داده آموزشی دارد. اما در روش Boosting مجموعه‌دادگان آموزش برای هر یادگیرنده بعدی بر نمونه‌هایی تمرکز می‌کند که توسط یادگیرنده‌های قبلی تولید شده، اشتباه طبقه‌بندی شده‌اند و احتمال انتخاب این نمونه‌ها افزایش می‌یابد.

ساده‌ترین روش Boosting برای مسائل طبقه‌بندی دوتایی طراحی شده است که مجموعه‌ای از سه یادگیرنده ضعیف را در یک‌زمان ایجاد می‌کند. اولین یادگیرنده (یا فرضیه) بر روی زیر*مجموعه ای تصادفی از دادگان آموزشی، آموزش داده می شود (مشابه روش* Bagging*). یادگیرنده دوم، ، بر روی زیر مجموعه ای متفاوت از مجموعه دادگان اصلی، آموزش داده می شود که دقیقاً نیمی از آن به درستی توسط تخمین زده شده و نیمی دیگر به اشتباه طبقه بندی شده اند. به این زیر مجموعه آموزشی که با توجه به تصمیم ساخته شده است «با اطلاعات ترین» مجموعه داده گفته می‌شود. یادگیرنده سوم، ، با نمونه هایی آموزش داده می شود که و در مورد آن ها اختلاف نظر دارند. در نهایت این سه یادگیرنده از طریق رأی اکثریت سه تایی ترکیب می‌شوند. ‏شکل (3˗3) نمایش گرافیکی از ایده اولیه الگوریتم* Boosting *را ترسیم کرده است.*

**

نمایش گرافیکی ایده اولین روش Boosting ارائه شده در [3]

*همچنین در [6] اثبات شده است که خطای آموزش این گروه با سه یادگیرنده به زیر محدود می شود که خطای هر یک از یادگیرندگان است، مشروط بر این که هر یادگیرنده دارای خطای ، کمترین مقداری که می توان از یک یادگیرنده دوتایی انتظار داشت، باشد.*

* الگوریتم AdaBoost

AdaBoost مخفف کلمه Adaptive Boosting و به معنای «تقویت تطبیقی»، یکی از پراستفاده‌ترین الگوریتم‌های یادگیری گروهی است. این الگوریتم پس از معرفی با چندین تغییر به الگوریتم AdaBoost.M1 و AdaBoost.M2 گسترش یافته شده است. در این بخش به الگوریتم AdaBoost.M1 که محبوب‌ترین الگوریتم در میان الگوریتم‌های AdaBoost است پرداخته می‌شود.

|  |
| --- |
| **الگوریتم 2: AdaBoost.M1** |
| **ورودی:** دادگان آموزش که و  ***یادگیرنده پایه؛***  *تعداد یادگیرندگان.*  ***مقداردهی اولیه:*** *.*  ***به‌ازای***  *انجام دهید:*   1. *زیرمجموعه را با استفاده از توزیع استخراج کنید.* 2. *یادگیرنده پایه را بر روی آموزش دهید و فرضیه را دریافت کنید* 3. *خطای فرضیه را محاسبه کنید:*     *اگر ادامه را رها کنید.*   1. *مقدار را تنظیم کنید:*      1. *توزیع نمونه‌برداری را به‌روز کنید:*     که در آن یک ثابت نرمال سازی برای اطمینان از صحت درست بودن تابع توزیع است.  ***پایان حلقه***  ***اکثریت آرای وزن‌دار:*** *برای نمونه ، تمام آرا را برای هر طبقه به دست آورید:*    ***خروجی:*** *طبقه با بیشترین مقدار .* |

AdaBoost دو تفاوت اساسی با روش‌های Boosting دارد: (1) نمونه‌هایی از توزیع نمونه به روز رسانی شده در هر مرحله به مجموعه دادگان بعدی نیز کشیده می‌شوند و (2) یادگیرندگان از طریق اکثریت آرای وزن‌دار ترکیب می‌شوند که در آن وزن‌های رأی‌گیری بر اساس اشتباهات آموزش یادگیرندگانی است که هر یک بر اساس توزیع نمونه برداری وزن‌دار شده اند.

شبه‌کد AdaBoost.M1 در الگوریتم 2 آورده شده است. توزیع نمونه برداری ، وزنی را برای هر نمونه آزمایش که اختصاص می‌دهد که از آن زیرمجموعه دادگان آموزشی برای هر یادگیرنده متوالی استخراج می شود. مقداردهی اولیه یک توزیع یکنواخت[[6]](#footnote-7) است و از این رو، تمام نمونه ها احتمال برابری برای استخراج شدن در مجموعه دادگان آموزشی ابتدایی دارند. در گام بعد خطای آموزش یادگیرنده ، یعنی ، با جمع کردن وزن های نمونه های اشتباه طبقه بندی شده مطابق با رابطه ‏(3˗5) محاسبه می شود. در این رابطه برابر با 1 است اگر عبارت شرطی داخل آن صادق باشد و در غیر این صورت برابر با صفر خواهد شد. این روش نیازمند آن است که خطای مطرح شده در آن کمتر از باشد، لذا مطابق با رابطه ‏(3˗6)، نیاز است تا نرمال سازی شود.

قلب الگوریتم AdaBoost.M1، قاعده به‌روزرسانی توزیع در رابطه ‏(3˗7) است. وزن های توزیع نمونه هایی که به درستی توسط یادگیرنده طبقه بندی شده اند، با ضریب کاهش می یابد. در حالی که وزن نمونه هایی که به اشتباه طبقه بندی شده اند بدون تغییر باقی می ماند. پس از آن وزن های به روز شده با ضریب نرمال سازی نرمال سازی می شوند. تا اطمینان حاصل شود که یک توزیع احتمال درست است. از این رو با هر یادگیرنده جدیدی که به گروه اضافه می‌شود، AdaBoost بر روی نمونه های دشوار تر تمرکز می‌کند. در هر تکرار t، رابطه ‏(3˗7) وزن نمونه های اشتباه طبقه بندی شده را به گونه ای افزایش می دهد که مجموع آن به برسد و وزن نمونه های درست طبقه بندی شده را به گونه ای کاهش می دهد که مجموع آن ها نیز به برسد. از آنجا که یادگیرندگان ضعیف می بایست خطایی کمتر از را داشته باشند، بنابراین تضمین می شود که حداقل یک نمونه آموزشی که قبلاً به اشتباه طبقه بندی شده است، به درستی طبقه بندی شود. هنگامی که آن یادگیرنده نتواند این کار را انجام دهد، روش AdaBoost آن را رها می‌کند. این روش تا جایی تکرار می‌شود که یادگیرنده ساخته شود. درنهایت این یادگیرندگان با رأی اکثریت وزن دار با هم ترکیب می شوند.

* الگوریتم XGBoost

XGBoost کوتاه شده عبارت eXtreme Gradient Boosting و به معنای «تقویت گرادیانی تشدید شده» است. این الگوریتم در تحقیق [8] در سال 2016 میلادی توسعه داده شده است و در بسیاری از مسائل تقریب تابع و طبقه بندی استفاده شده است و بسیاری از برندگان رقابت های هوش مصنوعی از این روش به عنوان بخشی از راه حل های خود استفاده کرده اند. اگرچه پیشرفت قابل توجهی در شبکه های عصبی عمیق حاصل شده است اما در بسیاری از کارها به واسطه نیازمندی تنظیم پارامترهای کمتر نسبت به مدل های عمیق، از XGBoost استفاده می‌شود.

XGBoost مانند سایر روش‌های تقویت گرادیانی، گروهی از درختان بازگشتی را استفاده می‌کند که شامل تابع افزایشی است:

که در آن تعداد درختان و مجموعه تمام درختان ممکن است. به این نکته توجه داشته باشید که در ‏(3˗9) به جای استفاده از از استفاده شده است، چراکه حرف در این الگوریتم بیانگر هسین[[7]](#footnote-8) تابع ضرر[[8]](#footnote-9) است. با توجه به تابع ضرر که معیاری برای تفاوت برچسب و پیش‌بینی یادگیرندگان است، XGBoost در صدد آن است که گروهی را بیابد که تابع ضرر را به حداقل برساند. بنابراین، تابع هزینه به صورت زیر تعریف می‌گردد:

می‌توان رابطه ‏(3˗10) را در تکرار ، به صورت زیر بازنویسی کرد:

که ، درخت جدیدی است که به گروه اضافه شده است. یافتن بهترین درخت برای تابع ضرر غیرممکن است، چراکه برشماردن تمام درختان را الزامی می کند. لذا با استفاده از رویکرد بهینه سازی تکراری سعی می‌شود درختی انتخاب شود که در هر گام تابع هزینه را به حداقل نزدیک تر کند. در روش XGBoost از روش نیوتون-رافسون برای همگرایی سریع تر به حداقل مقدار، استفاده شده است. این به این دلیل است که معمولاً تابع ضرر مربعی یا لگاریتمی است و متشق دوم آن به سادگی قابل محاسبه است. با استفاده از روش نیوتون-رافسون در حل مسئله یافتن حداقل مقدار تابع ضرر، بهترین کاهش تابع هزینه از درخت به صورت زیر بدست آمده است (به پیوست الف مراجعه شود):

که در آن تعداد برگ‌های درخت آخر، ضریب تنظیم وزن برگ است. و به ترتیب برابر با مجموع گرادیان و هسین تابع ضرر متناظر با نمونه های موجود در برگ است و به صورت زیر تعریف می‌شود:

همچنین وزن بهینه برای برگ‌های درخت (خروجی های درخت تصمیم اُم) نیز از طریق رابطه زیر محاسبه می شود:

باتوجه‌به این که در عمل نمی‌توان همه درخت‌های ممکن را برای رسیدن به چنین تابع هزینه حداقلی برشمارد، لذا در هنگام ساخت درخت تصمیم، بررسی شرط تقسیم در گره یعنی انتخاب ویژگی با بیشترین بهره اطلاعاتی[[9]](#footnote-10) (بیشترین کاهش تابع هزینه) به‌صورت زیر تعریف می‌شود:

که در آن و به ترتیب گرادیان و هسین گره والد و گرادیان و هسین فرزند به صورت زیر تعریف می‌شود:

مطابق روابط ریاضی گفته شده شبه کد روش XGBoost در الگوریتم 3 آورده شده است.

|  |
| --- |
| **الگوریتم 3: XGBoost** |
| **ورودی:**  : دادگان آموزش  : تابع ضرر مشتق پذیر  : تعداد تکرارهای Bossting  : نرخ یادگیری  : ضریب تنظیم برگ  : ضریب کاهش اولیه  ***مقداردهی اولیه:***    ***به‌ازای***  *انجام دهید:*  *(/\*محاسبه مجموع گرادیان و هسین تمام نمونه‌ها\*/)*    2. *ساخت درخت با استفاده از بهره ‏(3˗16) و پارامترهای ورودی* 3. *تمام برگ‌های درخت را درنظر بگیرید که* 4. ***به‌ازای***  *محاسبه کنید:*      1. *به‌روزرسانی مدل:*   ***پایان حلقه***  ***خروجی:*** *طبقه با بیشترین مقدار .* |

در مسائل طبقه‌بندی از تابع ضرر لگاریتمی استفاده می‌شود:

که در آن احتمال پیش‌بینی نمونه توسط یادگیرنده گروهی قبلی، ، است. با تعریف چنین تابع ضرری، در مسائل طبقه بندی محاسبات گرادیان و هسین بدون نیاز به محاسبه لگاریتم است و هزینه محاسباتی کاهش می یابد:

با جای‌گذاری و در سایر روابط، فرم ساده آن‌ها برای مسائل طبقه بندی به دست خواهد آمد.❧

گرچه در حالت کلی و در بسیاری از مسائل طبقه‌بندی، استفاده از الگوریتم‌های یادگیری گروهی باعث افزایش دقت طبقه‌بندی شده است، اما به‌کارگیری این الگوریتم‌ها با چالش‌هایی نیز همراه است. مهم‌ترین چالش آن که مانع از استفاده از این الگوریتم‌ها در موقعیت‌یابی مکان سرپوشیده است، پیچیدگی در محاسبات و نیازمندی به منابع محاسباتی و ذخیره‌ای بالا است. در نتیجه بلادرنگ بودن سیستم موقعیت‌یابی با مشکل مواجه می‌شود و همچنین هزینه تهیه چنین سیستمی زیاد خواهد بود. الگوریتم‌های ذکر شده از درخت تصمیم به‌عنوان یادگیرنده پایه استفاده می‌کنند. حجم محاسباتی و ذخیره‌ای این الگوریتم با تعداد گره‌های درخت تصمیم یا به طور کلی‌تر با عمق درخت ارتباط مستقیم دارد. در این پژوهش دو ایده برای کاهش عمق درخت پیشنهاد شده است. روش اول، به طور مستقیم بر روی مجموعه‌دادگان اعمال می‌شود و با کاهش ابعاد داده ورودی، به طور غیرمستقیم عمق درخت را کاهش می‌دهد. روش پیشنهادی بعدی استفاده از الگوریتم ژنتیک است. در این روش نه‌تنها سعی می‌شود که عمق درختان کاهش یابد، بلکه با درنظرگیری پارامترهای تنظیم الگوریتم‌های گروهی، زمان پردازش در عین حفظ دقت، کاهش یابد. در ادامه این دو پیشنهاد ارائه می‌گردد.

* 1. پیش‌پردازش داده

این مرحله از اهمیت بسیاری برخوردار است؛ زیرا کیفیت و کمیت داده‌های ورودی مستقیماً بر توانایی و سرعت مدل در یادگیری موقعیت و تعمیم آن تأثیر می‌گذارد. در این مرحله، اطلاعات جمع‌آوری شده از نقاط دسترسی و دستگاه‌های کاربران از جمله مکان واقعی، زمان، قدرت سیگنال، و سایر ویژگی‌های مرتبط با موقعیت، پس از پردازش وارد مرحله آموزش مدل می‌شوند. پیش‌پردازش شامل پاک‌سازی و نرمال‌سازی داده‌ها و استخراج ویژگی‌های اصلی است. این اقدامات در راستای افزایش یکنواختی و قابلیت تعمیم مدل در مواجهه با شرایط متفاوت در محیط‌های مختلف اجرا می‌شوند. داده‌های جمع‌آوری شده ممکن است شامل نویز، مقادیر گمشده باشند. پاک‌سازی داده‌ها شامل شناسایی و حذف داده‌های نامعتبر، تصحیح مقادیر گمشده و رفع ناسازگاری‌ها است.

یکی از پیشنهادهای اصلی در پردازش داده‌های موقعیت‌یابی استفاده از روش‌های کاهش ابعاد دادگان ورودی مدل‌های یادگیری گروهی است. روش‌های کاهش ابعاد اطلاعات با ابعاد بالا که در واقع، تعداد ویژگی‌های داده‌های موقعیت است و می‌تواند شامل مشخصات سیگنال و شرایط محیطی باشد را به فضای با ابعاد کمتر تبدیل می‌کنند. با اعمال تکنیک‌های کاهش ابعاد مانند «تحلیل مؤلفه‌های اساسی[[10]](#footnote-11)» (PCA) یا روش «t-جاسازی همسایگی تصادفی توزیع شده[[11]](#footnote-12)» (t-SNE)، داده‌های موقعیت ‌یا به‌صورت مؤثرتری برای مدل‌های یادگیری گروهی آماده می‌شوند. این‌گونه روش‌ها با درنظرگیری میزان تأثیر ویژگی‌ها، یعنی سطح سیگنال نقاط دسترسی و شرایط محیطی، به افزایش سرعت آموزش مدل‌ها کمک کرده و از مشکل نفرین ابعاد[[12]](#footnote-13) در داده‌ها جلوگیری می‌کنند. همچنین استفاده از روش‌های کاهش مرتبه می‌تواند باعث افزایش مقاوم‌بودن[[13]](#footnote-14) یک سیستم موقعیت‌یاب شود. به‌عنوان‌مثال، استفاده از روش PCA در [11] برای استخراج وابستگی اجزای سیگنال به موقعیت هدف در یک سیستم موقعیت‌یابی مبتنی بر CSI با فناوری Wi-Fi استفاده شده است و نرخ تشخیص اشتباه موقعیت هدف در یک دفتر کار را 3 درصد کاهش داده است. به‌عنوان کار مشابه در [12] نیز روش PCA را برای افزایش دقت و سرعت در موقعیت‌یابی با استفاده از الگوریتم‌های SVM، kNN و ANN پیشنهاد داده است.

کاهش ابعاد بردار ویژگی‌ها، به طور غیرمستقیم در روش‌های یادگیری گروهی بیان شده در ‏3˗2 اثرگذار است. از آنجا که عمق درختان یادگیرنده در یک الگوریتم گروهی، حداکثر می‌تواند برابر با تعداد ویژگی‌های بردار ورودی باشد، لذا با کاهش ابعاد، حداکثر عمق درختان کاهش یافته و درنتیجه مدل‌های یادگیرنده سریع تر آموزش می‌بینند. در ادامه دو روش محبوب و پر استفاده از روش های کاهش ابعاد معرفی می شود.

* + 1. تجزیه مؤلفه‌های اساسی (PCA)

تجزیه‌وتحلیل مؤلفه‌های اساسی (PCA) یک تکنیک کاهش ابعاد است که معمولاً در یادگیری ماشین و آمار استفاده می‌شود. هدف اصلی این تکنیک، تبدیل مجموعه‌داده با ابعاد بالا به یک فضای با ابعاد کمتر، در عین حفظ تنوع داده اصلی است. به‌عبارت‌دیگر، PCA به ساده‌تر کردن مجموعه‌داده‌های پیچیده، با پیداکردن یک مجموعه‌دادگان با ستون‌های جدید (مؤلفه‌های اصلی) که در آن داده‌ها بیشترین تغییر را دارند می‌پردازد.

PCA با انتقال داده‌های اصلی با ابعاد بالا به یک سیستم مختصات جدید که توسط مؤلفه‌های اصلی آن تعریف شده است عمل می‌کند. مؤلفه‌های اصلی، ترکیب‌های خطی از ویژگی‌های اصلی هستند که حداکثر واریانس در داده را حفظ می‌کنند. روش PCA با گام‌های زیر توصیف می‌شود:

* **گام اول، استانداردسازی**: اگر ویژگی‌های اصلی مقیاس‌های متفاوتی داشته باشند، اولین قدم استاندارد کردن داده‌ها با تفریق میانگین و تقسیم بر انحراف استاندارد برای هر ویژگی است. این کار تضمین می‌کند که همه ویژگی‌ها دارای مقیاس قابل‌قیاس هستند.
* **گام دوم، محاسبه کوواریانس:** *ماتریس کوواریانس نشان می‌دهد که چگونه ویژگی‌های مختلف نسبت به یکدیگر متفاوت هستند.*

که در آن، ماتریس داده در فضای *اصلی است، بردار میانگین از ستون‌های و تعداد دادگان ورودی است.*

* ***گام سوم، محاسبه بردارها و مقادیر ویژه:***

*که در آن، ماتریس ویژه (بردارهای ویژه به‌صورت ستونی)، ماتریس قطری مقادیر ویژه است*

* ***گام چهارم، محاسبه بردارها و مقادیر ویژه:*** *بردارهای ویژه بر اساس مقادیر ویژه مربوطه به ترتیب نزولی مرتب می‌شوند. بردارهای ویژه (مولفه‌های اصلی) یک مبنای متعامد جدید برای داده‌ها تشکیل می‌دهند. داده‌های اصلی بر روی این سیستم مختصات جدید تصویر می‌شود.*
* ***گام آخر، کاهش ابعاد:*** زیرمجموعه‌ای از مؤلفه‌های اصلی متناسب با مرتبه کاهش (ابعاد فضای ثانویه) انتخاب کنیم (). این مؤلفه‌ها به مهم‌ترین ویژگی‌ها وزن بیشتر و برای داده های ویژگی ها با تغییرات کم، وزن کمتری را اختصاص می‌دهد. با استفاده از مؤلفه‌های اساسی انتخاب شده، داده‌ها به فضای جدید منتقل می‌شود:

که در آن مجموعه‌دادگان جدید در فضای *ثانویه است.*

PCA ویژگی‌های مرتبط را حذف می‌کند و بیش برازش مدل را کاهش می‌دهد. همچنین می‌تواند در تصویرسازی دادگان در فضا با ابعاد کم کارآمد باشد.

* + 1. t-جاسازی همسایگی تصادفی توزیع شده (t-SNE)

روش PCA در حفظ نزدیکی‌ داده‌ها با مشکل مواجه است. به‌عبارت‌دیگر، اگر نقاطی در مجموعه‌داده به هم نزدیک باشند یا یک نزدیکی محلی داشته باشند، این نزدیکی در نتیجه حاصل از PCA ممکن است به طور کامل حفظ نشوند. استفاده پایه از t-SNE تصویرسازی داده‌های ابعاد بالاتر است. این روش برای اولین‌بار در سال 2008 میلادی و در [13] معرفی شد. ایده اصلی t-SNE مدل‌سازی احتمالاتی شباهت‌های بین نقاط داده در فضا با ابعاد بالا و انتقال آنها به احتمالات در فضای کم بعد و بیان معیاری برای توصیف واگرایی بین این دو توزیع احتمال و در نهایت ارائه الگوریتمی که این واگرایی را به حداقل برساند؛ بنابراین الگوریتم t-SNE را می‌توان در دو مرحله اصلی توصیف کرد: محاسبه احتمالات و به‌حداقل‌رساندن واگرایی.

* **گام اول، محاسبه احتمالات:** شباهت بین نقاط داده در فضای با ابعاد بالا، با استفاده از توزیع احتمال شرطی مدل‌سازی می‌شود. باتوجه‌به یک جفت از نقاط داده و ، احتمال شرطی به عنوان احتمالی تعریف می‌شود که اگر همسایه‌ها بر اساس شباهت‌هایشان انتخاب شوند، نقطه i نقطه j را به عنوان همسایه خود انتخاب کند:

که در آن و نقاط داده در فضای با ابعاد بالا و واریانس توزیع گوسی t برای نقطه است. همچنین برای نرمال سازی تمام جفت داده های شامل نقطه عبارت جمع در مخرج رابطه ‏(3˗28) قرار داده شده است.

به طور مشابه، شباهت در فضای با ابعاد پایین با استفاده از توزیع شرطی تعریف می شود:

که در آن و نقاط داده در فضای با ابعاد پایین است.

* **گام دوم، به‌حداقل‌رساندن واگرایی:** هدف، به‌حداقل‌رساندن واگرایی بین دو توزیع و است. معیاری که در t-SNE برای واگرایی بین دو توزیع احتمالاتی استفاده می شود، واگرایی Kullback-Leibler (به اختصار KL) به عنوان تابع هزینه است و به صورت زیر تعریف می‌شود:

**برای حداقل رساندن** این تابع هزینه، از روش نزول گرادیان استفاده می‌شود. گرادیان این تابع هزینه بر نقاط  **در فضای با ابعاد پایین محاسبه می‌شود و این نقاط در راستای کمینه کردن واگرایی به روز رسانی می شوند.**

**همانطور که در روابط ‏(3˗28) تا ‏(3˗30) مشاهده می شود،** محاسبه لگاریتمی و نمایی برای داده ورودی در مرحله آزمایش نیازمند صرف هزینه محاسباتی بیشتری نسبت به روش PCA است و می تواند در برخی از کاربردهای موقعیت یابی بلادرنگ یا کم هزینه مناسب نباشد. اما همچنان این روش تصویرسازی خوبی از مجموعه دادگان جمع آوری شده را ارائه می دهد.

**توجه به این نکته حائز اهمیت است که روش‌های کاهش مرتبه، علی‌رغم مزایایی که دارند، تفسیرپذیری مجموعه‌دادگان را بسیار کم می‌کند، علاوه‌برآن، ممکن است با انتخاب نادرست فضای ابعاد ثانویه، برخی از اطلاعات پس از اعمال این الگوریتم‌های از دست برود.**

* 1. الگوریتم ژنتیک

به‌کارگیری الگوریتم ژنتیک برای تنظیم ابرپارامترهای یک مدل یادگیری ماشین بسیار متداول است. این الگوریتم یکی از شاخه‌های پردازش تکاملی[[14]](#footnote-15) است که بر اساس ایده تکامل طبیعی و فرضیه داروین توسعه یافته است. بر اساس این ایده، در هر جامعه‌ای معمولاً افراد قوی‌تر از منابع بیشتری استفاده می‌کنند و با احتمال بیشتری زنده می‌مانند. در مقابل، افراد ضعیف‌تر با احتمال کم‌تری باقی می‌مانند. افراد باقیمانده از هر نسلی، تولیدمثل کرده و فرزندان نسل بعد را تشکیل می‌دهند. انتظار می‌رود که افراد هر نسل از جامعه، قوی‌تر از افراد نسل قبل خود باشد. در این انتقال نسل نیز ممکن است جهش ژنتیکی رخ دهد که باعث تنوع و پراکندگی در افراد یک نسل می‌شود.

مرحله ابتدایی در الگوریتم ژنتیک تعیین روش بازنمایی و تابع برازش[[15]](#footnote-16) است. «روش بازنمایی» مشخص می‌کند که چگونه هر وضعیت مسئله به‌صورت مناسب مدل و ذخیره شود. در این روش فنوتیپ[[16]](#footnote-17)ها وضعیت‌های واقعی مسئله هستند و ژنوتیپ[[17]](#footnote-18)ها یا کروموزوم[[18]](#footnote-19)ها به‌عنوان آرایه‌های مدل شده از هر وضعیت است و در فضای راه‌حل تعریف می‌شود ژن نیز به هر یک از آرایه‌های یک کروموزوم گفته می‌شود. نکته حائز اهمیت آن است که برای امکان یافتن جواب بهینه روش بازنمائی باید تمام راه‌حل‌های ممکن را پوشش دهد.

در مسئله تنظیم ابرپارامترهای یادگیری گروهی، فنوتیپ‌ها، ابرپارامترهای مدل یادگیری گروهی، یعنی «عمق درخت» و «تعداد یادگیرندگان» در نظر گرفته می‌شود که با کدگذاری به ژنوتیپ‌ها تبدیل می‌شود.

هر وضعیت مسئله توسط «تابع برازش» یا «ارزیاب» رتبه‌بندی می‌شود. تابع برازش باید به‌گونه‌ای تعریف شود که برای حالت‌های بهتر مقادیر بزرگ‌تری را برگرداند. در تنظیم ابرپارامترهای یادگیرنده گروهی موقعیت‌یاب، تابع برازش ترکیبی از متغیرهایی است که برای مدل یادگیری گروهی که بهترین دقت، سرعت و کمترین هزینه را دارد، بیشترین مقدار را بازگرداند؛ بنابراین، ترکیب خطی از معیار دقت، زمان تخمین موقعیت و عمق درخت (برای کاهش هزینه پیاده‌سازی) به‌عنوان تابع برازش استفاده می‌شود. ضرایب ترکیب خطی متناسب با میزان اهمیت هر یک از متغیرها انتخاب می‌گردد.

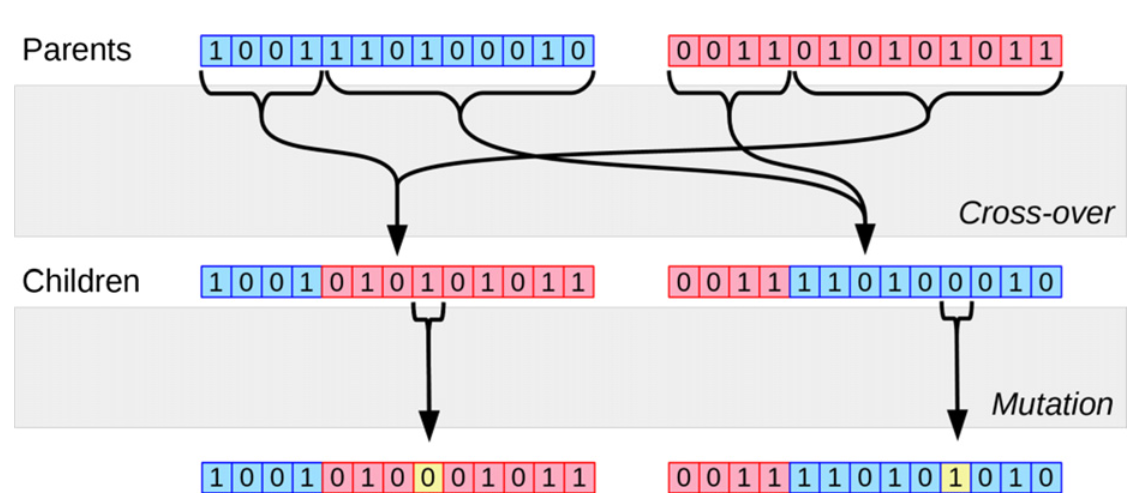
الگوریتم ژنتیک یک الگوریتم تکرارشونده برای یافتن کمینه مقدار تابع برازش است. در هر تکرار از الگوریتم ژنتیک دو عمل اصلی انجام می‌شود. عمل ترکیب متقاطع[[19]](#footnote-20) و عمل جهش[[20]](#footnote-21) در ساخت جمعیت جدید به کار می‌روند. در عمل ترکیب متقاطع دو کروموزوم والد به شیوه‌های مختلفی می‌توانند با هم ترکیب شوند و دو فرزند تولید کنند. متداول‌ترین شیوه ترکیب، ترکیب یک نقطه‌ای است که در آن یک نقطه از موقعیت‌های داخل کروموزوم به طور تصادفی انتخاب می‌شود. پس از آن فرزندان از برخورد کروموزوم‌های والد در نقطه پیوند متولد می‌شوند. در هر مرحله با یک احتمال مشخص، ممکن است هر کروموزوم مورد جهش واقع شود. در عمل جهش، هم ژن و هم مقدار جدید آن به طور تصادفی انتخاب می‌شود. در ‏شکل (3˗4) کروموزم های آبی و قرمز با یکدیگر ترکیب متقاطع شده‌اند و ژن‌های زرد رنگ با مقدار تصادفی جهش یافته‌اند.

در عمل ترکیب متقاطع، کروموزوم انتخابی با احتمال انتخاب می‌شود:

که در آن تابع برازش کروموزوم و جمعیت نسل است.

|  |
| --- |
| **الگوریتم 4: الگوریتم ژنتیک** |
| **ورودی:**  : جمعیت  : تابع برازش  : تکرار الگوریتم  ***به‌ازای***  *انجام دهید:*   1. *مجموعه تهی* 2. ***به‌ازای***  *تا اندازه انجام دهید:*    1. *انتخاب تصادفی از با احتمال در رابطه ‏(3˗31)*    2. *انتخاب تصادفی از با احتمال در رابطه ‏(3˗31)*    3. *ترکیب متقاطع و y.*    4. ***اگر*** *احتمال تصادفی جهش برآورده شود* ***آنگاه***  *را جهش دهید.*    5. *را به اضافه کنید.*   ***پایان حلقه***  ***خروجی:*** *بهترین در مطابق با تابع* |

شبه کد الگوریتم ژنتیک در الگوریتم ۴ آورده شده است. در الگوریتم ژنتیک معمولاً نسل‌های ابتدایی، افراد جامعه پراکندگی خوبی دارند، همچنین عملگر ترکیب بر روی وضعیت‌های مختلف والد، می‌تواند حالت‌های متفاوت نسبت به هر دو والد تولید کند. به تدریج، افراد با شباهت بیشتر در جمعیت ظهور می‌یابند و درنهایت بهترین فرد در جمعیت نهایی به عنوان دارنده بهترین ابرپارامتر الگوریتم یادگیری گروهی معرفی می‌گردد.



عملیات ترکیب متقاطع و جهش در تولید جمعیت جدید الگوریتم ژنتیک [14]

* 1. معیارهای ارزیابی

پارامترهای ارزیابی به دو دسته تقسیم‌بندی می‌شود. دسته اول معیارهای قابل‌اندازه‌گیری عملکرد و دسته دوم موضوعات کیفی عملکرد است:

* + 1. معیارهای قابل‌اندازه‌گیری عملکرد

عملکرد یک سیستم موقعیت‌یاب بر اساس معیارهای قابل‌اندازه‌گیری زیر ارزیابی می‌شود:

* Accuracy

Acccuracy یا دقت یکی از مهم‌ترین معیارهای عملکردی در هر سیستم موقعیت‌یاب است. در ادبیات یادگیری ماشین دقت به‌صورت است که در آن تعداد نمونه ها با پیش بینی درست و تعداد کل نمونه ها است. طبق پیش‌بینی مثبت و منفی نمونه‌ها، دقت به‌صورت زیر تعریف می‌شود:

که در آن TP و TN به ترتیب تعداد نمونه‌های مثبت و منفی هستند که به‌درستی تخمین زده‌اند و به همین ترتیب FP و FN نیز نمونه‌های مثبت و منفی هستند که به‌اشتباه تخمین زده شده‌اند.

* Precision

Precision یا صحت برای اندازه‌گیری این که چقدر می‌توان به طور مداوم به Accuracy دست‌یافت به کار برده می‌شود. در بیان یادگیری ماشین، نسبت موارد مثبت است که به‌درستی طبقه‌بندی‌شده است؛ بنابراین به‌صورت زیر تعریف می‌شود.

در موقعیت‌یابی با اثر انگشت، Preceision نشان‌دهنده نسبت تشخیص درست موقعیت هر بلوک هدف به مجموع تشخیص درست و نادرست همان بلوک است.

* Recall

از Recall به‌عنوان حساسیت نیز نام‌برده می‌شود و به نرخ مثبتِ درست (TP) اشاره دارد:

و در موقعیت‌یابی با اثر انگشت، به معنای نسبت تشخیص درست هر بلوک هدف به مجموع تشخیص درست همان بلوک و تشخیص نادرست بلوک‌های دیگر به‌عنوان بلوک هدف است.

* F1-Score

F1-Score به‌عنوان معیاری برای توازن میان Preceision و Recall معرفی می‌شود:

استفاده از F1-Score به‌جای Accuracy در ارزیابی یک مدل یا سیستم دارای مزایای متعدد است که می‌تواند اطلاعات مفیدتری ارائه دهد. وقتی که تشخیص خطاها ( و) مهم است، F1-Score بهترین معیار اندازه‌گیری است. اگر تعداد خطاها در یک کلاس به اندازه قابل توجهی باشد، Accuracy ممکن است زیاد باشد اما تشخیص خطاها را نادیده بگیرد.

* میانگین خطای موقعیت‌یابی

نرخ خطا، ، در یک سیستم موقعیت‌یابی به صورت که در آن تعداد نمونه‌های طبقه‌بندی نشده یا به اشتباه طبقه‌بندی شده است. این خطای موقعیت‌یابی عموماً با فاصله اقلیدسی بین مکان واقعی و مکان پیش‌بینی شده تعریف می‌شود و برخلاف معیارهای پیشین که ارزیابی را در فضای گسسته انجام میدادند، در فضای پیوسته صورت می پذیرد. میانگین خطای موقعیت‌یابی به‌صورت زیر تعریف می‌شود.

دقت (به درصد) تعداد دفعاتی را که یک سیستم به‌درستی یک مکان را مشخص می‌کند نشان می‌دهد؛ لذا، درصد پیش‌بینی‌های نادرست نیز به دست می‌آید. در صورت پیش‌بینی نادرست، فاصله مکان پیش‌بینی از مکان واقعی را با این معیار نمی‌توان به دست آورد.

* میانگین زمان تخمین

میانگین زمان تخمین، ، به عنوان معیار زمانی عملکرد سیستم موقعیت یاب معرفی می‌شود. چناچه مجموع میانگین زمان تخمین موقعیت و مدت زمان ارتباطی کاربر با واحد پردازش، بسیار کمتر از دوره نمونه برداری مقادیر حسگر باشد، سیستم موقعیت یاب بلادرنگ خوانده می شود. این معیار به صورت زیر محاسبه می شود:

که در آن مدت زمان تخمین نمونه اُم است.

* + 1. معیارهای کیفی عملکرد

سیستم‌های موقعیت‌یاب برای ارائه خدمات یکپارچه، باید معیارهای کیفی زیر را رعایت کنند:

* مقیاس‌پذیری

یک سیستم زمانی مقیاس‌پذیر است که به‌راحتی در یک منطقه آزمایشی بزرگ قابلیت استقرار داشته باشد و به تعداد دستگاه‌های متنوع با عملکرد مشابه خدمات ارائه کند.

* مقاوم‌بودن

مقاوم‌بودن این اطمینان را ایجاد می‌کند که یک سیستم موقعیت‌یاب می‌تواند خدمات خود را در یک محیط پیش‌بینی‌نشده مانند تغییر اشیای اطراف، وجود دستگاه‌های مختل‌کننده، عملکرد نادرست برخی از نقاط دسترسی و غیره ارائه دهد.

* مصرف انرژی

یکی از مسائل مهم در ارزیابی کیفی سیستم‌های موقعیت‌یاب مصرف بهینه توان است. نیاز به تعویض باتری و یا استفاده از منابع تغذیه علاوه بر افزایش هزینه، کار میدانی را نیز افزایش می‌دهد. محاسبات و پیچیدگی الگوریتم‌ها می‌تواند در مصرف انرژی سیستم‌های موقعیت‌یاب تأثیرگذار باشد.

* دردسترس‌بودن

یک سیستم موقعیت‌یاب در محیط سرپوشیده، نیازمند دردسترس‌بودن تجهیزات و فناوری را دارد. فناوری بلوتوث و Wi-Fi بر خلاف تکنولوژی UWB، ZigBee و غیره که نیازمند ادوات خاص خود هستند، معمولاً در اکثر تلفن‌های هوشمند وجود دارد؛ لذا علی‌رغم دقت خوب در سیستم‌های موقعیت‌یاب مبتنی بر UWB، انتخاب سیستم‌های مبتنی بر Wi-Fi و بلوتوث ارجحیت دارند.

* هزینه

پایین‌بودن هزینه سیستم‌های موقعیت‌یابی، تعداد کاربران آن را افزایش می‌دهد. هزینه زیرساخت‌های اولیه، تأمین نیروی انسانی برای تعمیر و نگهداری و کار میدانی از مهم‌ترین آن‌ها در سیستم‌های موقعیت‌یابی است؛ بنابراین ایجاد توازن بین سایر معیارهای ارزیابی و هزینه‌های پیاده‌سازی یک سیستم موقعیت‌یاب نیز نیازمند مهندسی و محاسبات است.

* 1. جمع‌بندی

در فصل سوم، راهکارهای پیشنهادی جهت بهبود دقت در تعیین موقعیت مورد بررسی قرار گرفته است. ابتدا به بررسی روش‌های یادگیری گروهی پرداخته شده است. مروری بر توسعه الگوریتم‌های یادگیری گروهی انجام شد و اجزای مهم آن‌ که شامل نمونه‌برداری و انتخاب داده، آموزش یادگیرندگان و ترکیب یادگیرندگان ضعیف مورد بررسی قرار گرفته است. سپس روش‌های Bagging و Boosting به‌تفصیل بررسی شده‌اند که دراین‌بین الگوریتم‌های جنگل تصادفی، AdaBoost و XGBoost که محبوب‌ترین الگوریتم‌های یادگیری گروهی هستند، بیان شده‌اند. الگوریتم‌های یادگیری گروهی، علی‌رغم دقت تخمین خوب، نیازمند منابع محاسباتی و ذخیره‌ای زیاد هستند و به سبب آن می‌توانند تخمین بلادرنگ موقعیت را با چالش همراه کنند. ازاین‌رو، دو راهکار کاهش ابعاد و به‌کارگیری الگوریتم ژنتیک ارائه شد. فرایند پیش‌پردازش داده با استفاده از تجزیه مؤلفه‌های اساسی (PCA) و t-SNE به‌عنوان راهکار غیرمستقیم مورد ارزیابی قرار گرفته است. الگوریتم ژنتیک نیز به‌عنوان راهکار مستقیم در تعیین ابرپارامترهای یادگیری گروهی برای موقعیت‌یابی معرفی شده است. در پایان این فصل، معیارهای ارزیابی با تأکید بر معیارهای قابل‌اندازه‌گیری عملکرد مانند Accuracy، Precision، Recall، F1-Score، میانگین خطای موقعیت‌یابی و میانگین زمان تخمین بررسی و معیارهای کیفی عملکرد معرفی شده‌اند.

[1] C. Zhang and Y. Ma, *Ensemble machine learning: methods and applications*. Springer, 2012.

[2] B. V. Dasarathy and B. V. Sheela, "A composite classifier system design: Concepts and methodology," *Proceedings of the IEEE,* vol. 67, no. 5, pp. 708-713, 1979.

[3] R. E. Schapire, "The strength of weak learnability," *Machine learning,* vol. 5, pp. 197-227, 1990.

[4] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," *Journal of computer and system sciences,* vol. 55, no. 1, pp. 119-139, 1997.

[5] L. Breiman, "Bagging predictors," *Machine learning,* vol. 24, pp. 123-140, 1996.

[6] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural computation,* vol. 3, no. 1, pp. 79-87, 1991.

[7] D. H. Wolpert, "Stacked generalization," *Neural networks,* vol. 5, no. 2, pp. 241-259, 1992.

[8] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," presented at the Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, California, USA, 2016.

[9] M. Segal and Y. Xiao, "Multivariate random forests," *Wiley interdisciplinary reviews: Data mining and knowledge discovery,* vol. 1, no. 1, pp. 80-87, 2011.

[10] R. Díaz-Uriarte and S. Alvarez de Andrés, "Gene selection and classification of microarray data using random forest," *BMC bioinformatics,* vol. 7, pp. 1-13, 2006.

[11] G. Gottardi, M. A. Hannan, B. Li, A. Polo, M. Salucci, and F. Viani, "PCA-Based inversion of WiFi signal for robust device-free indoor target detetion," in *Journal of Physics: Conference Series*, 2020, vol. 1476, no. 1, p. 012015: IOP Publishing.

[12] Y. Basiouny, M. Arafa, and A. M. Sarhan, "Enhancing Wi-Fi fingerprinting for indoor positioning system using single multiplicative neuron and PCA algorithm," in *2017 12th International Conference on Computer Engineering and Systems (ICCES)*, 2017, pp. 295-305: IEEE.

[13] L. Van der Maaten and G. Hinton, "Visualizing data using t-SNE," *Journal of machine learning research,* vol. 9, no. 11, 2008.

[14] A. Urso, A. Fiannaca, M. La Rosa, V. Ravì, and R. Rizzo, "Data mining: Prediction methods," *Encycl. Bioinforma. Comput. Biol. ABC Bioinforma,* vol. 1, p. 3, 2018.

1. Bias [↑](#footnote-ref-2)
2. Variance [↑](#footnote-ref-3)
3. Random Forest [↑](#footnote-ref-4)
4. Random subspace methods [↑](#footnote-ref-5)
5. Majority voting [↑](#footnote-ref-6)
6. uniform [↑](#footnote-ref-7)
7. Hessian [↑](#footnote-ref-8)
8. Loss Function [↑](#footnote-ref-9)
9. Information Gain [↑](#footnote-ref-10)
10. Principal Component Analysis [↑](#footnote-ref-11)
11. t-distributed Stochastic Neighbor Embedding [↑](#footnote-ref-12)
12. Curse of dimensionality [↑](#footnote-ref-13)
13. Robustness [↑](#footnote-ref-14)
14. Evolutionary Computing [↑](#footnote-ref-15)
15. Fitness function [↑](#footnote-ref-16)
16. Phenotype [↑](#footnote-ref-17)
17. Genotype [↑](#footnote-ref-18)
18. Chromosome [↑](#footnote-ref-19)
19. Crossover [↑](#footnote-ref-20)
20. Mutation [↑](#footnote-ref-21)