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

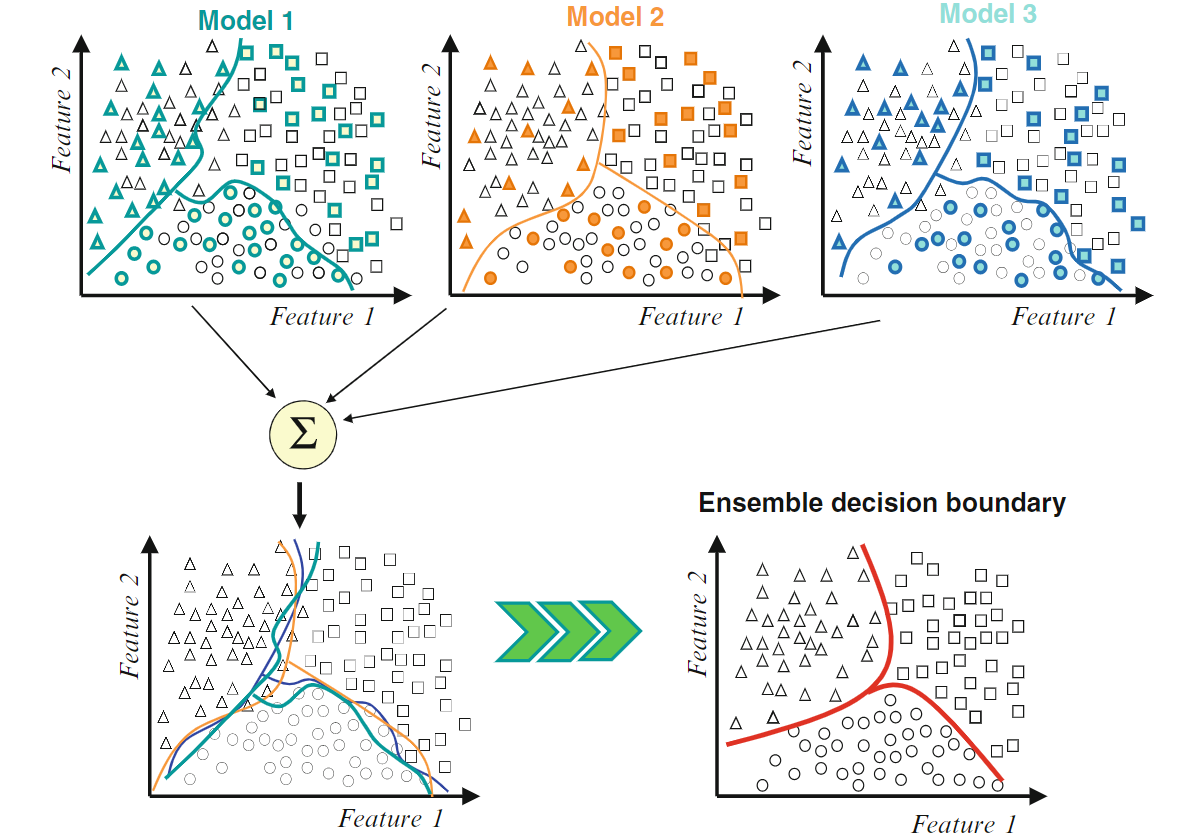
در فصل قبل انواع روش‌های موقعیت‌یابی با استفاده از الگوریتم‌های یادگیری ماشین بیان شد. روش‌های پایه با وجود سادگی در پیاده‌سازی، قادر به تخمین موقعیت با دقت بالا نیستند. به همین منظور، روش‌های یادگیری ماشین برای بهبود دقت موقعیت‌یابی مکان‌های سرپوشیده به کار گرفته شده‌اند. همچنین بر اساس فناوری‌های ارتباطی 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 مورد استفاده برای آموزش اعضای گروه، هسته هر سیستم یادگیری گروهی است. الگوریتم‌های رقابتی متعددی برای آموزش طبقه‌بندی‌کننده‌های گروه توسعه داده شده‌اند. با این حال، Bagging (و الگوریتم های مرتبط مانند جنگل تصادفی)، Boosting (و انواع آن)، تعمیم پشته‌سازی به عنوان رایج‌ترین رویکردهای به کار گرفته شده حساب می‌شود. این رویکردها در بیان می‌شود.

* ترکیب یادگیرندگان

مرحله نهایی در سیستم های مبتنی بر مجموعه شامل ترکیب پیش بینی های یادگیرندگان است. استراتژی ترکیب به نوع یادگیرندگان مورد استفاده 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، در حالی که ترکیب‌های پیچیده‌تر ممکن است نیاز به یک مرحله اضافی داشته باشند.

ابتدا فرض می شود که فقط برچسب ها از خروجی یادگیرندگان در دسترس است. تصمیم یادگیرنده را به صورت 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

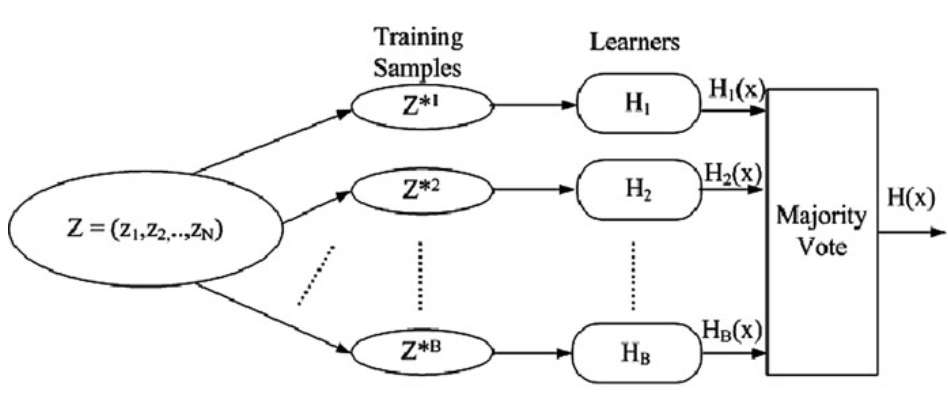
1. **رأی اکثریت وزن‌دار:** اگر دلیلی بر این باور داشته باشیم که برخی از یادگیرندگان بیشتر از بقیه صحیح هستند، وزن دادن به تصمیمات آن یادگیرندگان می تواند عملکرد کلی را در مقایسه با اکثریت آرا بهبود بخشد. می توانیم وزن را به یادگیرنده بر اساس عملکرد کلی تخمین آن اختصاص دهیم. گروه با توجه به رأی اکثریت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

یعنی کل آرای وزن دار 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.

مجموعه ای غنی از الگوریتم‌های گروهی در چندسال گذشته توسعه یافته است. با این وجود، برخی از این الگوریتم های منتخب و به خوبی تثبیت شده هستند که قابلیت های آن ها نیز به طور گسترده آزمایش و گزارش شده است. در ادامه برخی از برجسته ترین الگوریتم‌های یادگیری گروهی ارائه شده است و در فصل بعدی نتایج حاصل از شبیه سازی آن ها 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 طبقه بندی [1]

* جنگل تصادفی

یکی از نسخه های خلاقانه روش Bagging، الگوریتم جنگل تصادفی است که اساس آن ترکیب مجموعه از درختان تصمیم آموزش دیده با مکانیزم Bagging است. جنگل تصادفی توسط Breiman معرفی شد [5]. جنگل تصادفی را می توان هم برای متغیرهای گسسته، که در موقعیت یابی می تواند شرایط محیطی منظور شود، و هم برای متغیرهای پیوسته، که در موقعیت یابی می تواند RSS سیگنال ارتباطی بی سیم باشد، استفاده کرد. به طور مشابه، متغیرهای پیش بینی شده نیز می تواند گسسته یا پیوسته باشد.

از نقطه نظر 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 هم چون موقعیت یابی که ابعاد بردار ویژگی بالا است استفاده شود و در نهایت این که می توان آن را به صورت 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 یگ 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 نمونه برداری با جایگذاری انتخاب شده به عنوان دادگان آموزشی، درخت تصمیم را تشکیل دهید:*    1. *با تمام ویژگی‌ها در یک گره شروع کن.*    2. *گام های زیر را برای گره های تقسیم نشده تکرار کن تا زمانی که شرط توقف درخت (گره با یک ویژگی/عمق درخت) ارضا شود:*       1. *ویژگی را به صورت 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 موقعیت یابی که نیازمند سرعت بالا در محاسبات و سادگی پیاده سازی است. نیاز به تنظیم سه پارامتر وجود دارد:

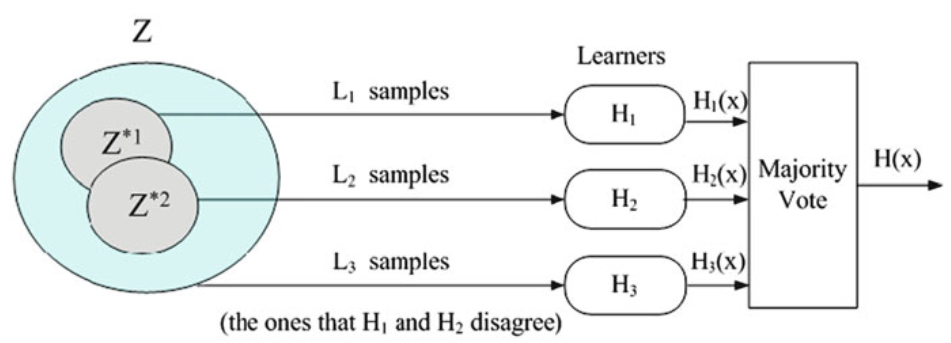
* ، تعداد ویژگی‌هایی که به صورت تصادفی در هر گره انتخاب می‌شود.
* ، تعداد درختان موجود در جنگل
* اندازه درخت، که یا با کمترین اندازه برگ مشخص می شود یا بیشینه تعداد گره های نهایی درختان

در مسائل طبقه بندی مقدار پیشنهاد شده است [10]. برای تنظیم پارامترهای و اندازه درخت، در این پژوهش روش الگوریتم ژنتیک را پیشنهاد شده است که در بخش ‏3˗4 معرفی می‌شود.

* + 1. روش Boosting

Boosting کار ابتدایی Schapire در [3] معرفی شد. این روش، یک روش تکرار شونده برای تولید یک طبقه بندی کننده قوی است و قابلیت رسیدن به خطای آموزش پایین، با گروهی از یادگیرندگان ضعیفی که کمی بهتر از حدس تصادفی عمل می کنند را دارد. این روش با اینکه مجموعه یادگیرندگان ضعیف را با استفاده از رأی اکثریت ترکیب می کند، اما از یک منظر مهم با روش Bagging تفاوت دارد. در روش Bagging انتخاب نمونه ها برای آموزش هر یک از یادگیرندگان ضعیف از طریق انتخاب بدون جایگذاری صورت می گیرد، به این معنی که هر نمونه شانس برابری برای قرار گرفتن در هر مجموعه داده آموزشی دارد. اما در روش Boosting مجموعه دادگان آموزش برای هر یادگیرنده بعدی بر نمونه هایی تمرکز می کند که توسط یادگیرنده های قبلی تولید شده، اشتباه طبقه بندی شده اند و احتمال انتخاب این نمونه ها افزایش می یابد.

ساده ترین روش Boosting برای مسائل طبقه بندی دوتایی طراحی شده است که مجموعه ای از سه یادگیرنده ضعیف را در یک زمان 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 پر استفاده‌ترین 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 الگوریتم های AdaBoost است پرداخته می‌شود.

|  |
| --- |
| **الگوریتم 2: AdaBoost.M1** |
| **ورودی:** دادگان آموزش که و  ***یادگیرنده پایه؛***  *تعداد یادگیرندگان.*  ***مقداردهی اولیه:*** *.*  ***به ازای***  *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 نمونه برداری را به روز 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 به روز رسانی 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 روش های تقویت گرادیانی، گروهی از درختان بازگشتی را استفاده می کند 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:

می توان 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:

با توجه به این که در عمل نمی توان همه درخت های 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) به صورت زیر تعریف می شود:

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:***    ***به ازای***  *انجام دهید:*  *(/\*محاسبه مجموع گرادیان و هسین تمام نمونه ها\*/)*    2. *ساخت درخت با استفاده از بهره ‏(3˗16) و پارامترهای ورودی* 3. *تمام برگ های درخت را درنظر بگیرید که* 4. ***به ازای***  *محاسبه کنید:*      1. *به روز رسانی مدل:*   ***پایان حلقه***  ***خروجی:*** *طبقه با بیشترین مقدار .* |

در مسائل طبقه بندی از تابع ضرر لگاریتمی استفاده می شود:

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 محاسبات کاهش می یابد:

* 1. پیش‌پردازش داده

متن

* 1. الگوریتم ژنتیک

متن

* 1. معیارهای ارزیابی

متن

* 1. جمع‌بندی

متن

[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.

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)