- روشهای مبتنی بر انتخاب ویژگی

مساله انتخاب ویژگی، یکی از مسائلی است که در مبحث یادگیری ماشین و همچنین شناسائی آماری الگو مطرح است. این مساله در بسیاری از کاربردها (مانند طبقه بندی) اهمیت به سزائی دارد، زیرا در این کاربردها تعداد زیادی ویژگی وجود دارد، که بسیاری از آنها یا بلااستفاده هستند و یا اینکه بار اطلاعاتی چندانی ندارند. حذف نکردن این ویژگیها مشکلی از لحاظ اطلاعاتی ایجاد نمیکند ولی بار محاسباتی را برای کاربرد مورد نظر بالا میبرد. و علاوه بر این باعث میشود که اطلاعات غیر مفید زیادی را به همراه دادههای مفید ذخیره کنیم.

برای مساله انتخاب ویژگی، راه حلها و الگوریتمهای فراوانی ارائه شده است که بعضی از آنها قدمت سی یا چهل ساله دارند. مشکل بعضی از الگوریتمها در زمانی که ارائه شده بودند، بار محاسباتی زیاد آنها بود، اگر چه امروزه با ظهور کامپیوترهای سریع و منابع ذخیره سازی بزرگ این مشکل، به چشم نمیآید ولی از طرف دیگر، مجموعههای دادهای بسیار بزرگ برای مسائل جدید باعث شده است که همچنان پیدا کردن یک الگوریتم سریع برای این کار مهم باشد.

در این بخش ما در ابتدا تعاریفی که برای انتخاب ویژگی ارائه شدهاند و همچنین، تعاریف مورد نیاز برای درک این مساله را ارائه میدهیم. سپس روشهای مختلف برای این مساله را بر اساس نوع و ترتیب تولید زیرمجموعه ویژگیهای کاندید و همچنین نحوه ارزیابی این زیرمجموعهها دسته بندی میکنیم. سپس تعدادی از روشهای معرفی شده در هر دسته را معرفی و بر اساس اهمیت، تا جائی که مقدور باشد، آنها را تشریح و الگوریتم برخی از آنها را ذکر میکنیم. لازم به ذکر است که بدلیل اینکه مبحث انتخاب ویژگی به مبحث طبقه بندی بسیار نزدیک است، بعضی از مسائلی که در اینجا مطرح میشود مربوط به مبحث طبقه بندی میباشد. توضیحات ارائه شده برای الگوریتمهای مختلف در حد آشنائی است. شما میتوانید برای کسب اطلاعات بیشتر به منابع معرفی شده مراجعه کنید.

3-1- تعاریف

مساله انتخاب ویژگی بوسیله نویسندگان مختلف، از دیدگاههای متفاوتی مورد بررسی قرار گرفته است. هر نویسنده نیز با توجه به نوع کاربرد، تعریفی را از آن ارائه داده است. در ادامه چند مورد از این تعاریف را بیان میکنیم[6]:

1. **تعریف ایدهآل:** پیدا کردن یک زیرمجموعه با حداقل اندازه ممکن، برای ویژگیها است، که برای هدف مورد نظر اطلاعات لازم و کافی را در بر داشته باشد. بدیهی است که هدف تمام الگوریتمها و روشهای انتخاب ویژگی همین زیر مجموعه است.
2. **تعریف کلاسیک:** انتخاب یک زیرمجموعه M عنصری از میان N ویژگی، به طوریکه $M < N$ باشد و همچنین مقدار یک تابع معیار (Criterion Function) برای زیرمجموعه مورد نظر، نسبت به سایر زیرمجموعههای هماندازه دیگر بهینه باشد. این تعریفی است که Fukunaga و Narenda در سال 1977 ارائه دادهاند.

3. **افزایش دقت پیشگوئی:** هدف انتخاب ویژگی این است که یک زیرمجموعه از ویژگیها برای افزایش دقت پیشگوئی انتخاب شوند. به عبارت دیگر کاهش اندازه ساختار بدون کاهش قابل ملاحظه در دقت پیشگوئی طبقهبندی کنندهای که با استفاده از ویژگیهای داده شده بدست میآید.

4. **تخمین توزیع کلاس اصلی:** هدف از انتخاب ویژگی این است که یک زیرمجموعه کوچک از ویژگیها انتخاب شوند، توزیع ویژگیهایی که انتخاب میشوند، بایستی تا حد امکان به توزیع کلاس اصلی با توجه به تمام مقادیر ویژگیهای انتخاب شده نزدیک باشد.

روشهای مختلف انتخاب ویژگی، تلاش میکنند تا از میان 2^N زیر مجموعه کاندید، بهترین زیرمجموعه را پیدا کنند. در تمام این روشها بر اساس کاربرد و نوع تعریف، زیر مجموعهای به عنوان جواب انتخاب می-شود، که بتواند مقدار یک تابع ارزیابی را بهینه کند. با وجود اینکه هر روشی سعی میکند که بتواند، بهترین ویژگیها را انتخاب کند، اما با توجه به وسعت جوابهای ممکن، و اینکه این مجموعههای جواب بصورت توانی با N افزایش پیدا میکنند، پیدا کردن جواب بهینه مشکل و در N های متوسط و بزرگ بسیار پر هزینه است.

به طور کلی روشهای مختلف انتخاب ویژگی را بر اساس نوع جستجو به دسته های مختلفی تقسیم بندی میکنند. در بعضی روشها تمام فضای ممکن جستجو میگردد. در سایر روشها که میتواند مکاشفهای و یا جستجوی تصادفی باشد، در ازای از دست دادن مقداری از کارآئی، فضای جستجو کوچکتر میشود.

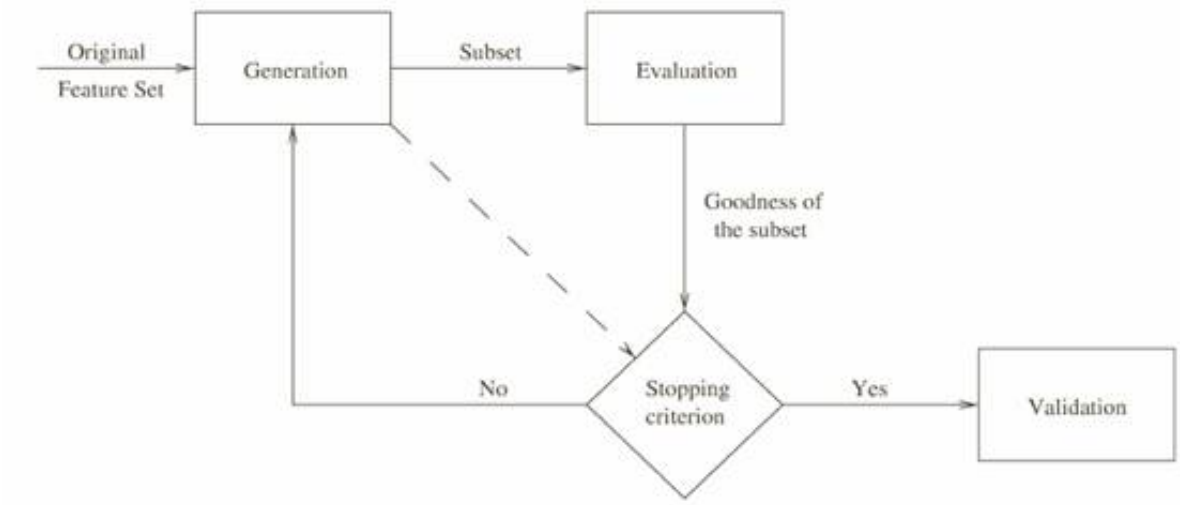
برای اینکه بتوانیم تقسیم بندی درستی از روشهای مختلف انتخاب ویژگی داشته باشیم، به این صورت عمل میکنیم که فرآیند انتخاب ویژگی در تمامی روشها را به این بخشها تقسیم میکنیم:

1. **تابع تولید کننده (Generation procedure) :** این تابع زیر مجموعههای کاندید را برای روش مورد نظر پیدا میکند.

2. **تابع ارزیابي (Evaluation function) :** زیرمجموعه مورد نظر را بر اساس روش داده شده، ارزیابی و یک عدد به عنوان میزان خوبی روش باز میگرداند. روشهای مختلف سعی در یافتن زیرمجموعهای دارند که این مقدار را بهینه کند.

3. **شرط خاتمه:** برای تصمیمگیری در مورد زمان توقف الگوریتم.

4. **تابع تعیین اعتبار (Validation procedure) :** تصمیم میگیرد که آیا زیر مجموعه انتخاب شده معتبر است یا خیر؟



شکل12 - فرآیند انتخاب ویژگی

تابع تولید کننده در واقع تابع جستجو است. این تابع زیرمجموعه‌های مختلف را به ترتیب تولید میکند، تا بوسیله تابع ارزیابی، مورد ارزیابی قرا بگیرد. تابع تولید کننده از یکی از حالتهای زیر شروع به کار می-کند:

1) بدون ویژگی
2) با مجموعه تمام ویژگیها
3) با یک زیرمجموعه تصادفی

در حالت اول ویژگیها به ترتیب به مجموعه اضافه میشوند و زیرمجموعه‌های جدید را تولید میکنند. این عمل آنقدر تکرار میشود تا به زیر مجموعه مورد نظر برسیم. به اینگونه روشها، روشهای پائین به بالا میگویند.

در حالت دوم از یک مجموعه شامل تمام ویژگیها، شروع میکنیم و به مرور و در طی اجرای الگوریتم، ویژگیها را حذف میکنیم، تا به زیرمجموعه دلخواه برسیم. روشهایی که به این صورت عمل میکنند، روشهای بالا به پائین نام دارند.

یک تابع ارزیابی، میزان خوب بودن یک زیرمجموعه تولید شده را بررسی کرده و یک مقدار به عنوان میزان خوب بودن زیرمجموعه مورد نظر بازمیگرداند. این مقدار با بهترین زیرمجموعه قبلی مقایسه می-شود. اگر زیر مجموعه جدید، بهتر از زیرمجموعه‌های قدیمی باشد، زیرمجموعه جدید به عنوان زیرمجموعه بهینه، جایگزین قبلی میشود.

باید توجه داشت که بدون داشتن یک شرط خاتمه مناسب، فرآیند انتخاب ویژگی ممکن است، برای همیشه درون فضای جستجو، برای یافتن جواب سرگردان بماند. شرط خاتمه میتواند بر پایه تابع تولید کننده باشد، مانند:

1) هر زمان که تعداد مشخصی ویژگی انتخاب شدند.
2) هر زمان که به تعداد مشخصی تکرار رسیدیم.

و یا اینکه بر اساس تابع ارزیابی انتخاب شود، مانند:

1) وقتیکه اضافه یا حذف کردن ویژگی، زیرمجموعه بهتری را تولید نکند
2) وقتیکه به یک زیرمجموعه بهینه بر اساس تابع ارزیابی برسیم.

تابع تعیین اعتبار جزئی از فرآیند انتخاب ویژگی نیست، اما در عمل بایستی یک زیرمجموعه ویژگی را در شرایط مختلف تست کنیم تا ببینیم که آیا شرایط مورد نیاز، برای حل مساله مورد نظر ما را دارد یا نه؟ برای اینکار میتوانیم از دادههای نمونهبرداری شده و یا مجموعه داده های شبیه سازی شده استفاده کنیم.

3-2- روشهای مختلف انتخاب ویژگی

در این بخش ابتدا روشهای مختلف انتخاب ویژگی را بر اساس دو معیار تابع تولید کننده و تابع ارزیابی طبقه بندی میکنیم. سپس آنها را بر اساس عملکرد دستهبندی و نحوه اجرای هر دسته را به اختصار شرح میدهیم.

3-2-1- توابع تولید کننده

اگر تعداد کل ویژگیها برابر N باشد، تعداد کل زیرمجموعههای ممکن برابر 2^N میشود. این تعداد برای N های متوسط هم خیلی زیاد است. بر اساس نحوه جستجو در میان این تعداد زیر مجموعه، روشهای مختلف انتخاب ویژگی را میتوان به سه دسته زیر تقسیمبندی نمود:

1) جستجوی کامل
2) جستجوی مکاشفهای
3) جستجوی تصادفی

در ادامه به معرفی هر کدام از این دستهها میپردازیم.

3-2-1-1- جستجوی کامل

در روشهایی که از این نوع جستجو استفاده میکنند، تابع تولید کننده بر اساس تابع ارزیابی استفاده شده، تمام فضای جواب (زیرمجموعههای ممکن) را برای یافتن جواب بهینه جستجو میکند. البته Schlimmer استدلال آورده است که[7]: "کامل بودن جستجو به این معنی نیست که جستجو باید جامع باشد".

توابع مکاشفهای مختلف زیادی طراحی شدهاند، تا جستجو را بدون از دست دادن شانس پیدا کردن جواب بهینه، کاهش دهند. اما با توجه به بزرگی فضای جستجو، ($O(2^N)$، این روشها باعث میشوند که فضای کمتری جستجو شود. روشها و تکنیکهای مختلفی برای اینکار استفاده شده اند، بعضی از آنها از تکنیک بازگشت به عقب (Backtracking) نیز در جریان کار استفاده کردهاند، مانند: branch and bound، best first search و beam search.

3-2-1-2- جستجوی مکاشفه ای

در روشهای با این نوع جستجو، در هر بار اجرای الگوریتم، یک ویژگی به مجموعه ویژگی انتخاب شده، اضافه و یا از آن حذف میشود. به همین دلیل پیچیدگی زمانی آنها محدود و کمتر از ($O(N^2)$ میباشد. در

اینگونه موارد، اجرای الگوریتم خیلی سریع میباشد و پیاده سازی آنها نیز بسیار ساده است.

3-2-1-3- جستجوی تصادفی

روشهایی که از این نوع جستجو استفاده میکنند، محدوده کمتری از فضای کل حالات را جستجو میکنند، که اندازه این محدوده به حداکثر تعداد تکرار الگوریتم بستگی دارد. در این روشها پیدا شدن جواب بهینه به اندازه منابع موجود و زمان اجرای الگوریتم بستگی دارد. در هر بار تکرار، تابع تولید کننده تعدادی از زیرمجموعههای ممکن از فضای جستجو را به صورت تصادفی انتخاب میکند و در اختیار تابع ارزیابی قرار میدهد. تابع تولید کننده تصادفی پارامترهایی دارد که بایستی تنظیم شوند، تنظیم مناسب این پارامترها در سرعت رسیدن به جواب و پیدا شدن جوابهای بهتر موثر است.

3-2-2- تابع ارزیابی

پیدا شدن یک زیرمجموعه بهینه از مجموعه ویژگیها، به صورت مستقیم با انتخاب تابع ارزیابی بستگی دارد. چرا که اگر تابع ارزیابی به زیرمجموعه ویژگی بهینه یک مقدار نامناسب نسبت دهد، این زیرمجموعه هیچگاه بعنوان زیرمجموعه بهینه انتخاب نمیشود. مقادیری که توابع ارزیابی مختلف به یک زیرمجموعه میدهند، با هم متفاوت است.

توابع ارزیابی را میتوان به طرق مختلفی دسته بندی کرد. در اینجا ما دسته بندیای که توسط Dash و Liu ارائه شده است [6] را بیان میکنیم. آنها این معیارها را به پنج دست تقسیم کردهاند:

1. **معیارهای مبتنی بر فاصله (Distance Measures)**: در این معیارها، مثلاً برای یک مساله دو کلاسه، یک ویژگی یا یک مجموعه ویژگی مثل X بر یک ویژگی یا یک مجموعه ویژگی دیگر مثل Y ارجحیت دارد، اگر که با آن مجموعه ویژگی مقادیر بزرگتری برای اختلاف بین احتمالات شرطی دو کلاس داشته باشیم. نمونهای از این معیارها همان معیار فاصله اقلیدسی میباشد.

2. **معیارهای مبتنی بر اطلاعات (Information Measures)** : این معیارها میزان اطلاعاتی را که بوسیله یک ویژگی بدست میآید را در نظر میگیرند. ویژگی X در این روشها بر ویژگی Y اولویت دارد، اگر اطلاعات بدست آمده از ویژگی X بیشتر از اطلاعاتی باشد، که از ویژگی Y بدست میآید. نمونهای از این معیارها همان معیار آنتروپی میباشد.

3. **معیارهای مبتنی بر وابستگی (Dependence Measures)**: این معیارها که با عنوان معیارهای همبستگی (Correlation) نیز شناخته میشوند، قابلیت پیشگوئی مقدار یک متغیر بوسیله یک متغیر دیگر را اندازهگیری میکنند. ضریب (Coefficient) یکی از معیارهای وابستگی کلاسیک است و میتوانیم آنرا برای یافتن همبستگی بین یک ویژگی و یک کلاس به کار ببریم. اگر همبستگی ویژگی X با کلاس C بیشتر از همبستگی ویژگی Y با کلاس C باشد، در اینصورت ویژگی X بر ویژگی Y برتری دارد. با یک تغییر کوچک، میتوانیم وابستگی یک ویژگی با ویژگی-های دیگر را اندازهگیری کنیم. این مقدار درجه افزونگی این ویژگی را نشان میدهد. همه توابع ارزیابی بر پایه معیار وابستگی را میتوانیم بین دو دسته معیارهای مبتنی بر فاصله و اطلاعات تقسیم کنیم. اما به خاطر اینکه این روشها از یک دید دیگر به مساله نگاه میکنند، این کار را

انجام نمیدهیم.

4. **معیارهای مبتنی بر سازگاری (Consistency Measures)**: این معیارها جدیدتر هستند و اخیراً توجه بیشتری به آنها شده است. این معیارها خصوصیات متفاوتی نسبت به سایر معیارها دارند، زیرا که به شدت به دادههای آموزشی تکیه دارند و در انتخاب یک زیرمجموعه از ویژگیها تمایل دارند، که مجموعه ویژگیهای کوچکتری را انتخاب کنند. این روشها زیرمجموعههای با کمترین اندازه را بر اساس از دست دادن یک مقدار قابل قبول سازگاری که توسط کاربر تعیین می-شود، پیدا میکنند.

5. **معیارهای مبتنی بر خطای طبقه بندی کننده (Classifier Error Rate Measures)**: روشهایی که این نوع از تابع ارزیابی را استفاده میکنند، با عنوان "wrapper methods" شناخته میشوند. دقت عملکرد در این روشها برای تعیین کلاسی که نمونه داده شده متعلق به آن است، برای نمونههای دیده نشده بسیار بالا است، اما هزینههای محاسباتی در آنها نیز نسبتاً زیاد است.

در جدول زیر مقایسهای بین انواع مختلف تابع ارزیابی، صرف نظر از نوع تابع تولید کننده مورد استفاده، انجام شده است. پارامترهایی که برای مقایسه استفاده شدهاند به صورت زیر میباشند:

1. **عمومیت(Generality)** : اینکه بتوان زیرمجموعه انتخاب شده را برای طبقهبندی کنندههای متفاوت به کار ببریم.
2. **پیچیدگی زمانی:** زمان لازم برای پیدا کردن زیرمجموعه ویژگی جواب.
3. **دقت:** دقت پیشگوئی با استفاده از زیرمجموعه انتخاب شده.

علامت "---" که در ستون آخر آمده است، به این معنی است که در مورد میزان دقت حاصل نمیتوانیم مطلبی بگوئیم. بجز خطای طبقهبندی کننده، دقت سایر توابع ارزیابی به مجموعه داده مورد استفاده و طبقه بندی کنندهای که بعد از انتخاب ویژگی برای طبقهبندی کلاسها استفاده میشود، بستگی دارد.

نوع تابع ارزیابی	عمومیت	پیچیدگی زمانی	دقت
معیار فاصله	دارد	پائین	---
معیار اطلاعات	دارد	پائین	---
معیار وابستگی	دارد	پائین	---
معیار سازگاری	دارد	متوسط	---
خطای طبقه بندی کننده	ندارد	بالا	خیلی زیاد

شکل 13- مقایسه توابع ارزیابی مختلف

3-2-3- دسته بندی و تشریح الگوریتم های مختلف انتخاب ویژگی

در این قسمت بر اساس تابع ارزیابی و تابع تولید کننده، روشهای مختلف انتخاب ویژگی را به چند دسته تقسیم بندی میکنیم و سپس تعدادی از روشها را شرح داده و الگوریتم کار را به صورت شبه کد، ذکر میکنیم.

قبل از اینکه بحث را ادامه دهیم، لازم است که متغیرهای به کار رفته در شبه کدها را معرفی کنیم. این متغیرها و شرح آنها به صورت زیر میباشد:

- **D:** مجموعه آموزشی
- **S:** مجموعه ویژگی اصلی (شامل تمام ویژگیها)
- **N:** تعداد ویژگیها
- **T:** زیرمجموعه ویژگی انتخاب شده
- **M:** تعداد ویژگیهای انتخاب شده یا تعداد ویژگیهایی که لازم است انتخاب شوند.

3-2-3-1- تابع ارزیابی مبتنی بر فاصله - تابع تولید کننده مکاشفه ای

مهمترین روش در این گروه Relief [8] است. در اینجا ما ابتدا این روش را به عنوان نماینده این گروه شرح میدهیم، سپس یک مرور مختصری بر سایر روشها خواهیم داشت.

روش Relief از یک راه حل آماری برای انتخاب ویژگی استفاده میکند، همچنین یک روش مبتنی بر وزن است که از الگوریتمهای مبتنی بر نمونه الهام گرفته است. روش کار به این صورت است که از میان مجموعه نمونههای آموزشی، یک زیرمجموعه نمونه انتخاب میکنیم. کاربر بایستی تعداد نمونه-ها(NoSample) در این زیرمجموعه را مشخص کرده باشد. و آنرا به عنوان ورودی به الگوریتم ارائه دهد. الگوریتم به صورت تصادفی یک نمونه از این زیرمجموعه را انتخاب میکند، سپس برای هر یک از ویژگیهای این نمونه، نزدیکترین برخورد (Nearest Hit) و نزدیکترین شکست (Nearest Miss) را بر اساس معیار اقلیدسی پیدا میکند.

نزدیکترین برخورد نمونهای است که کمترین فاصله اقلیدسی را در میان سایر نمونههای همکلاس با نمونه انتخاب شده دارد. نزدیکترین شکست نیز نمونهای است که کمترین فاصله اقلیدسی را در میان نمونههایی که همکلاس با نمونه انتخاب شده نیستند، دارد.

ایده اصلی در این الگوریتم این است که هر چه اختلاف بین اندازه یک ویژگی در نمونه انتخاب شده و نزدیکترین برخورد کمتر باشد، این ویژگی بهتر است و بعلاوه یک ویژگی خوب آن است که اختلاف بین اندازه آن ویژگی و نزدیکترین شکست وی بیشتر باشد. دلیل کار هم خیلی ساده است، ویژگیهایی که به خوبی دو کلاس (یا یک کلاس از سایر کلاسها) را از هم تمییز میدهند، برای نمونههای متعلق به دو کلاس متفاوت مقادیری نزدیک بههم نمیدهند و یک فاصله معنیداری بین مقادیری که به نمونههای یک کلاس میدهند و مقادیری که به سایر کلاس(ها) میدهند وجود دارد.

الگوریتم پس از تعیین نزدیکترین برخورد و نزدیکترین شکست، وزنهای ویژگیها را به روزرسانی می-کند، این بهروزرسانی به این صورت است که مربع اختلاف بین مقدار ویژگی مورد نظر در نمونه انتخاب شده و نمونه نزدیکترین برخورد از وزن ویژگی کم میشود و در عوض مربع اختلاف بین مقدار ویژگی در نمونه انتخاب شده و نزدیکترین شکست به وزن ویژگی اضافه میشود. هر چه مقدار این وزن بزرگتر باشد، ویژگی مورد نظر، بهتر میتواند نمونههای یک کلاس را از دیگران جدا کند.

بعد از تعیین فاصله برای تمام نمونههای موجود در مجموعه نمونهها، الگوریتم ویژگیهایی را که وزن آنها کمتر یا مساوی با یک حد آستانهای است، را حذف میکند، و سایر ویژگیها بعنوان زیرمجموعه ویژگی جواب باز میگردند. مقدار حد آستانهای توسط کاربر تعیین میگردد، البته ممکن است که بصورت

اتوماتیک بوسیکه یک تابعی از تعداد کل ویژگیها تعیین شود و یا اینکه با سعی و خطا تعیین گردد. همچنین میتوان ویژگیهایی که وزن آنها منفی است را حذف کرد.

Relief(D, S, *NoSample*, *Threshold*)
(1) $T = \phi$
(2) Initialize all weights, W_i, to zero.
(3) For $i = 1$ to *NoSample*/* Arbitrarily chosen */
Randomly choose an instance x in D
Finds its *nearHit* and *nearMiss*
For $j = 1$ to N
$W_j = W_j - diff(x_j, nearHit_j)^2 + diff(x_j, nearMiss_j)^2$
(4) For $j = 1$ to N
If $W_j \geqslant Threshold$
Append feature f_j to T
(5) Return T

شکل 14- الگوریتم Relief

الگوریتم Relief برای ویژگیهای دارای نویز یا ویژگیهای دارای همبستگی خوب کار میکند و پیچیدگی زمانی آن بصورت خطی و تابعی از تعداد ویژگیها و تعداد نمونههای مجموعه نمونه میباشد. و هم برای دادههای پیوسته و هم برای دادههای صوری خوب کار میکند.

یکی از محدودیتهای اساسی این الگوریتم این است که ویژگیهایی که دارای افزونگی باشند را پیدا نمیکند و بنابراین مجموعههای غیر بهینه را پیدا میکند که دارای افزونگی هستند. این مشکل را می-توان با یک جستجوی تعیین جامعیت برای زیرمجموعههای تولید شده توسط الگوریتم حل کرد. علاوه بر این مشکل دیگر این الگوریتم این است که با مسائل دو کلاسه خوب کار میکند. این محدودیت نیز با الگوریتم Relief-F [9] مرتفع شده است، با الگوریتم جدید مشکل دادههای غیر کامل (نمونههای آموزشی غیرکامل) نیز حل شده است.

روشی که Jakub Segen [10] برای انتخاب ویژگی مطرح کرده است، از یک تابع ارزیابی استفاده می-کند که مجموع یک معیار اختلاف آماری و یک معیار پیچیدگی ویژگی را محاسبه کرده و آنرا مینیمم می-کند. این الگوریتم، اولین ویژگی را که بهتر بتواند کلاسها را از هم تمییز دهد را پیدا میکند. سپس ویژگیهایی را پیدا میکند، که در ترکیب با ویژگیهای انتخاب شده، جدائیپذیری کلاسها را افزایش دهند. این فرآیند زمانی متوقف میشود که به حداقل معیار بازنمائی مورد انتظار برسیم.

3-2-3-2- تابع ارزیابی مبتنی بر فاصله - تابع تولید کننده کامل

استفاده از این ترکیب در روشهای قدیمی نظیر B&B (Branch and Bound) یافت میشود. سایر روشهای این گروه، نسخههای متفاوتی از B&B هستند. به این ترتیب که یا یک تابع تولید کننده دیگری را استفاده کردهاند (BFF [11]) و یا اینکه از یک تابع ارزیابی متفاوتی استفاده کردهاند (Bobrowski's

[12] method). در اینجا ابتدا به شرح B&B میپردازیم و سپس یک شرح مختصری در مورد دو روش دیگر ارائه میدهیم.

تعریف کلاسیک ارائه شده بوسیله Fukunaga و Narenda از انتخاب ویژگی، احتیاج دارد که تابع ارزیابی یکنوا باشد. یعنی اگر دو زیرمجموعه ویژگی A و B با اندازههای M و N موجود باشند، و $B \subset A$ در اینصورت مقدار تابع ارزیابی برای A نباید بیشتر از مقدار تابع برای B باشد. این تعریف باعث ایجاد مشکل در مسائل دنیای واقعی میشود، زیرا اندازه تخمینی زیرمجموعه ویژگی بهینه در حالت عمومی ناشناخته است.

البته به سادگی میتوان این تعریف را تغییر داد تا با مسائل عمومی سازگار شود، به اینصورت که می-گوئیم: الگوریتمهای مشابه B&B تلاش میکنند که دو شرط زیر را همزمان ارضاء کنند:

1. زیرمجموعه ویژگی جواب تا حد امکان کوچک باشد.
2. یک کران برای مقدار تابع ارزیابی را در نظر بگیرد. (یا یک اندازه مینیمم برای تعداد ویژگیهای انتخاب شده مثلاً بهترین زیرمجموعه ویژگی سه عنصری)

بوسیله کران تعیین شده، فضای جستجو تا حد امکان کوچک میشود. به این ترتیب الگوریتم B&B از یک زیرمجموعه شامل تمام ویژگیهای موجود شروع میکند و درخت جستجو را تشکیل میدهد. در این درخت در ریشه تمام ویژگیها قرار دارند و فرزندان وی، زیرمجموعههایی هستند که زیرمجموعه، گره پدر هستند و از حذف تنها یکی از عناصر پدرشان تشکیل شدهاند. این روند برای سایر گرههای درخت تکرار میشود تا به مجموعهها تک عنصری (یا تعداد ویژگیهای تعیین شده بعنوان کران) برسیم. یعنی برگ-های درخت مجموعههای تک عنصری هستند و ریشه درخت یک مجموعه شامل همه ویژگیهای موجود.

با توجه به این خاصیت که تمام زیرمجموعههای یک مجموعه مقدار کمتری برای تابع ارزیابی دارند، در حین جستجو اگر یک گره به واسطه کم بودن مقدار تابع ارزیابی انتخاب نشد، زیرشاخههای آنرا برای یافتن جواب جستجو نمیکنیم، چون قطعاً تابع ارزیابی مقدار کمتری را برای آنها باز میگرداند.
عموماً توابع ارزیابی زیر برای اینکار استفاده میشوند:

- (Mahalanobis Distance)فاصله ماهالانوبیس
- (Discriminant Function)تابع جداساز
- (Fisher Criterion)معیار فیشر
- (Bhattacharya)فاصله باتاچاریا
- Divergence

یک الگوریتم مشابه برای انتخاب ویژگی BFF است، در این الگوریتم، تابع جستجو به این صورت تغییر کرده است که مشابه حل مساله جستجوی یک مسیر بهینه در یک درخت وزندار با یک استراتژی تغییر یافته از Best first search است. این الگوریتم تضمین میکند که بهترین هدف(زیرمجموعه بهینه) بدون از دست دادن جامعیت مساله پیدا شود، البته با ارضای معیار یکنوا بودن تابع ارزیابی.

```
B&B(D, S, M)
  if (card(S) <> M) {
    /* subset generation */
    j = 0
    For all features f in S {
      S_j = S − f/* remove one feature at a time */
      if (S_j is legitimate) */
        if IsBetter(S_j, T)
          T = S_j
        /* recursion */
        B&B(S_j, M)}
      j++}
    return T }}
```

شکل 15- الگوریتم Branch and Bound

3-2-3-3- تابع ارزیابی مبتنی بر اطلاعات - تابع تولید کننده مکاشفه ای

در این دسته دو روش وجود دارد:

1) روش درخت تصمیم

در روش درخت تصمیم، نمونهها به یک الگوریتم C4.5[15]، که یکی از درختهای تصمیمگیری است اعمال میشوند، سپس درخت هرس شده حاصل از الگوریتم C4.5 را گرفته و کلیه ویژگیهایی که در آن وجود دارد را بعنوان جواب مساله باز میگردانیم.

الگوریتم C4.5، از یک تابع مکاشفه بر پایه اطلاعات استفاده میکند، یک فرم ساده این توابع برای مسائل دو کلاسه به صورت زیر است:

$$I(p,n) = \frac{-p}{p+n}\log_2\frac{p}{p+n} - \frac{n}{p+n}\log_2\frac{n}{p+n}$$

که در آن **p** تعداد نمونههای کلاس اول و **n** تعداد نمونههای کلاس دوم است. فرض کنید که صفت $\mathbf{F_1}$ بعنوان ریشه درخت در نظر گرفته شده است و مجموعه آموزشی را به دو زیرمجموعه T_1 و T_0 تقسیم کرده است. آنتروپی ویژگی $\mathbf{F_1}$ برابر است با:

$$E(F_1) = \frac{p_0+n_0}{p+n} I(p_0,n_0) + \frac{p_1+n_1}{p+n} I(p_1,n_1)$$

الگوریتم درخت تصمیم به صورت زیر است[13]:

DTM(D)
(1) $T = \phi$
(2) Apply C4.5 to the training set, D
(3) Append all features appearing in the pruned decision tree to T
(4) Return T

شکل 16- الگوریتم درخت تصمیم

2) روش استفاده شده توسط Koller و Sahami

روش استفاده شده توسط Koller و Sahami که اخیراً ارائه شده است، بر این پایه استوار است که ویژگیهایی که داده مفید چندانی را در بر ندارند و یا اصلاً داده مفیدی را در اختیار قرار نمیدهند و می-توان آنها را با سایر ویژگیها نمایش داد، یا ویژگیهایی که بیربط هستند و یا داده اضافی هستند، را شناسائی و حذف میکنیم. برای پیاده سازی این مطلب، تلاش میکنیم تا با پوشش مارکوف آنها را پیدا کنیم، به این صورت که یک زیرمجموعه مانند T، یک پوشش مارکوف برای ویژگی f_i است، اگر f_i برای زیرمجموعه T بصورت مشروط هم از کلاس و هم از سایر ویژگیهایی که در T نیستند، مستقل باشد[14].

3-2-3-4- تابع ارزیابی مبتنی بر اطلاعات - تابع تولید کننده کامل

مهمترین روشی که در این گروه میتوانیم پیدا کنیم، روش Minimum Description Length Method (MDLM) است[16]. نویسندگان این روش تلاش میکنند تا همه ویژگیهای بدون استفاده (بیربط یا اضافی) را حذف نمایند، با این دید که اگر ویژگیهای زیرمجموعه V را بتوانیم بصورت یک تابع ثابتی مانند F که وابسته به کلاس نیست، بر اساس یک زیرمجموعه ویژگی دیگر مانند U بیان کنیم. در این صورت وقتی که مقادیر ویژگیهای زیرمجموعه U شناخته شده باشند، ویژگیهای موجود در زیرمجموعه V بدون استفاده هستند.

از دیدگاه انتخاب ویژگی، اجتماع دو زیرمجموعه U و V، مجموعه کامل، شامل تمام ویژگیها را تشکیل میدهد. و کاری که ما باید در انتخاب ویژگی انجام دهیم این است که این دو زیرمجموعه را جدا کنیم. برای انجام این کار، نویسندگان MDLM، از معیار Minimum Description Length Criterion (MDLC) که بوسیله Rissanen ارائه شده است[17]، استفاده کردهاند. آنها فرمولی را بدست آوردهاند، که شامل تعداد بیتهای لازم برای انتقال کلاسها، پارامترهای بهینه سازی، ویژگیهای مفید و ویژگیهای غیرمفید است. الگوریتم تمام زیرمجموعههای ممکن (2N) جستجو میکند و بعنوان خروجی زیرمجموعهای را بازمیگرداند که معیار MDLC را ارضا کند. این روش میتواند تمام ویژگیهای مفیدی را پیدا کند که دارای توزیع نرمال باشند. برای حالتهای غیر نرمال این روش قادر نیست، ویژگیهای مفید را پیدا کند. الگوریتم زیر روش کار و فرمولهای استفاده شده را نشان میدهد.

MDLM(D)

(1) $MDL = \infty$

(2) For all feature subsets L

1.1 Compute $Length_L = \sum_{i=1}^{i=q} \frac{p_i}{2} \log \frac{|D_L(i)|}{|D_L|} + h_L$

where $h_L = \frac{1}{2}(N - M)(N + M + 3) \log P + \sum_{i=1}^{i=q} M(M + 3) \log P_i$,

N – total number of features,

M – number of features in the candidate subset,

P – total number of instances in D,

P_i – number of instances with class label i,

q – total number of class labels,

D_L – covariance matrix formed from all the useful feature vectors,

$D_L(i)$ – covariance matrix formed from the useful feature vectors, of class i,

$|.|$ – denotes determinant.

1.2 If $Length_L < MDL$ then

$T = L, MDL = Length_L$

(3) Return T

شکل 17- الگوریتم روش Minimum Description Length Method (MDLM)

3-2-3-5- تابع ارزیابی مبتنی بر وابستگی - تابع تولید کننده مکاشفه ای

دو روش عمده در این گروه میبینیم:

3) Probability of Error & Average Correlation Coefficient (POE1ACC)

که خود شامل هفت روش است[18]، ما در اینجا روش هفتم را که به گفته نویسنده کاملتر است را بررسی میکنیم.

در این روش اولین ویژگی به این صورت تعیین میشود که احتمال خطا را برای تمام ویژگیها محاسبه میکنیم، ویژگی با کمترین احتمال خطا (P_e)، به عنوان اولین ویژگی انتخاب میشود. ویژگی بعدی، آن ویژگی است که مجموع وزندار P_e و میانگین ضریب همبستگی(ACC) با ویژگی(های) انتخاب شده را مینیمم کند. سایر ویژگیها به همین ترتیب انتخاب میشوند. میانگین ضریب همبستگی به اینصورت است که میانگین ضریب همبستگی ویژگی کاندید با ویژگیهای انتخاب شده در آن نقطه محاسبه می-شوند.

POE+ACC(D, M, w_1, w_2)

(1) $T = \phi$

(2) Find the feature with minimum P_e and append it to T

(3) For $i = 1$ to $M - 1$

Find the next feature with minimum $w_1(P_e) + w_2(ACC)$

Append it to T

(4) Return T

شکل 18- الگوریتم Probability of Error & Average Correlation Coefficient (POE1ACC)

این روش میتواند تمام ویژگیها را بر اساس مجموع وزندار درجهبندی کند. شرط خاتمه نیز در این روش تعداد ویژگیهای مورد نیاز خواهد بود.

4) PreSet

این روش از تئوری مجموعههای ناهموار (Rough) استفاده میکند. در اینجا یک کاهش (Reduct) پیدا میکنیم. یک کاهش مانند R از یک مجموعه P به این معنی است که نمونهها بوسیله آن به خوبی مجموعه P طبقه بندی شوند. پس از پیدا کردن یک کاهش، تمام ویژگیهایی که در مجموعه کاهش داده شده وجود ندارند، را از مجموعه ویژگی حذف میکنیم. سپس ویژگیها را بر اساس اهمیت آنها درجه-بندی میکنیم. اهمیت یک ویژگی بر این اساس بیان میشود که یک ویژگی در جریان کلاسبندی چقدر اهمیت دارد. این معیار بر پایه صفات وابستگی ویژگی تعیین میگردد [19].

6-3-2-3- تابع ارزیابی مبتنی بر سازگاری - تابع تولید کننده کامل

روشهایی که در این گروه قرار دارند، در سالهای اخیر ارائه شدهاند. ما به صورت مختصر سه روش این گروه را بررسی میکنیم ولی بحث اصلی ما بر روی روش اول است.

1) Focus

این روش یک حداقل گرا است، به این معنی که سعی میکند که حداقل تعداد ویژگی ممکن را برای ارائه پیدا کند. این روش فرضیههای قابل تعریف را بررسی کرده و فرضیهای که بتواند سازگاری را با حداقل تعداد ویژگی ممکن برقرار کند را بعنوان جواب بازمیگرداند.

در سادهترین پیاده سازی برای این روش، برای یافتن یک ناسازگاری با یک زیرمجموعه از ویژگیهای انتخاب شده، درخت جستجو را به صورت سطح به سطح (جستجو در پهنا)، پیمایش میکنیم. در این جریان که از مجموعههای کوچکتر شروع میشود، در صورتی که به یک ناسازگاری برسیم، مجموعه انتخاب شده رد میشود و جستجو با مجموعه بعدی ادامه مییابد. به محض اینکه به یک مجموعه برسیم که ناسازگاری نداشته باشد، جستجو متوقف شده و مجموعه یاد شده به عنوان جواب انتخاب میشود.

میتوان گفت که این روش به نویز حساس است و نمیتواند نویز را مدیریت کند، زیرا در صورتی که نویز وجود داشته باشد، هیچ زیرمجموعهای را نمیتوان پیدا کرد که ناسازگاری نداشته باشد و الگوریتم تمام ویژگیها را به عنوان جواب بازمیگرداند. با یک تغییر کوچک میتوانیم این مساله را حل کنیم، به اینصورت که اجازه میدهیم یک میزان معینی از ناسازگاری در مجموعه انتخاب شده وجود داشته باشد.

```
Focus(D, S)
  (1) T = S
  (2) For i = 0 to N
        For each subset L of size i
          If no inconsistency in the training set D then
            T = L
            Return T
```

شکل 19- الگوریتم روش Focus

2) Schlimmer

این روش از یک شمارش سیستماتیک برای تابع تولید کننده و یک معیار ناسازگاری نیز به عنوان تابع ارزیابی استفاده میکند. همچنین با استفاده از یک تابع مکاشفه‌ای سرعت جستجو را برای یافتن زیرمجموعه بهینه افزایش میدهد. این تابع مکاشفه‌ای یک معیار قابلیت اعتماد است، بر پایه این ایده که احتمال مشاهده یک ناسازگاری مشاهده شود، نسبتی از درصد مقادیری است که زیاد مشاهده شده‌اند[7].

3) MIFES1

این روش در انتخاب ویژگی شباهت زیادی به روش Focus دارد. در اینجا مجموعه نمونه‌ها را به شکل یک ماتریس ارائه میدهیم، هر عنصر نماینده یک ترکیب یکتا از یک نمونه منفی و یک نمونه مثبت است. یک ویژگی مانند f یک پوشش برای یک نمونه از ماتریس نامیده میشود، اگر برای نمونه‌های منفی و نمونه‌های مثبت، مقادیر عکسی داشته باشد. این روش از یک پوشش با همه ویژگیها شروع میکند، و تکرار میشود تا وقتی که هیچ کاهشی نتوانیم برای پوشش داشته باشیم. مشکل اساسی این روش اینست که فقط برای مسائل دو کلاسه و ویژگیهای منطقی قابل استفاده است. توضیحات کاملتر راجع به پوشش و نحوه اجرای الگوریتم را میتوانید در مرجع شماره [20] پیدا کنید.

7-3-2-3- تابع ارزیابی مبتنی بر سازگاری - تابع تولید کننده تصادفی

نماینده این گروه که جدیدتر هستند، روش LVF است[21]. این روش فضای جستجو را بصورت تصادفی با استفاده از یک الگوریتم Las Vegas جستجو میکند، که یکسری انتخابهای احتمالی انجام میدهد تا با استفاده از آنها سریعتر به جواب بهینه برسیم، همچنین از یک معیار سازگاری که با معیار استفاده شده در الگوریتم Focus متفاوت است.

این روش برای هر زیرمجموعه کاندید، تعداد ناسازگاری را محاسبه میکند، که بر این ایده استوار است که کلاس محتملتر آن است که در میان نمونه‌های این زیرمجموعه ویژگی، تعداد بیشری متعلق به آن کلاس باشند. یک حد آستانه‌ای برای ناسازگاری در نظر گرفته میشود، که در ابتدا ثابت و بصورت پیش فرض صفر است و هر زیرمجموعه‌ای که مقدار ناسازگاری آن بیشتر باشد، رد میشود.

```
LVF(D, S, MaxTries, δ)
  (1) T = S
  (2) For i = 1 to MaxTries {
        randomly choose a subset of features, S_j
        if card(S_j) ⩽ card(T)
          if inConCal(S_j, D) ⩽ δ
            if card(S_j) < card(T)
              T = S_j
              output S_j
            else
              append S_j to T
              output S_j as 'yet another solution' }
  (3) return T
```

شکل 20- الگوریتم روش LVF

این روش میتواند هر زیرمجموعه بهینه را پیدا کند، حتی برای دادههای دارای نویز، به شرط آنکه سطح نویز درست را در ابتدا مشخص کنیم. یک مزیب این روش اینست که احتیاجی نیست که کاربر مدت زیادی را برای بدست آوردن یک زیرمجموعه بهینه منتظر بماند، زیرا الگوریتم هر زیرمجموعهای که بهتر از بهترین جواب قبلی باشد (هم از لحاظ اندازه زیرمجموعه انتخاب شده و هم از لحاظ نرخ سازگاری)، را به عنوان جواب باز میگرداند.

این الگوریتم کارا است، زیرا تنها مجموعههایی برای ناسازگاری تست میشوند، که تعداد ویژگیهای درون آن کمتر یا مساوی بهترین زیرمجموعهای است که تا کنون پیدا شده است. پیاده سازی آن آسان است و پیدا شدن زیرمجموعه بهینه را اگر منابع موجود اجازه دهند، تضمین میکند. یکی از مشکلات این الگوریتم این است که برای پیدا کردن جواب بهینه زمان بیشتری نسبت به الگوریتمهایی که از توابع تولید کننده مکاشفهای استفاده میکنند، نیاز دارد، چون این الگوریتم از دانش مربوط به زیرمجموعههای قبلی استفاده نمیکند.

8-3-2-3- تابع ارزیابی مبتنی بر خطای طبقه بندی کننده- تابع تولید کننده مکاشفه ای

همانطور که قبلاً نیز اشاره کردیم، به مجموعه روشهایی که از تابع ارزیابی مبتنی بر نرخ خطای طبقه-بندی کننده استفاده میکنند، (بدون توجه به نوع تابع تولید کننده استفاده شده) روشهای wrapper میگویند. در این گروه روشهای مشهور زیر را میتوانیم ببینیم:

1) SFS (Sequential Forward Selection)

این روش، کارش را با یک مجموعه خالی شروع میکند، سپس در هر تکرار یک ویژگی با استفاده از تابع ارزیابی مورد استفاده، به مجموعه جواب اضافه میکند، این کار را تکرار میکند تا زمانیکه تعداد ویژگی لازم انتخاب شود. مشکلی که این روش با آن روبروست، اینست که ویژگی اضافه شده در

صورتیکه مناسب نباشد، از مجموعه جواب حذف نمیشود[23].

2) SBS (Sequential Backward Selection)

این روش برعکس SFS کارش را با مجموعهای شامل تمام ویژگیها شروع میکند و در هر بار تکرار الگوریتم، ویژگی که بوسیله تابع ارزیابی انتخاب میشود، را از مجموعه مورد نظر حذف میکند. این کار را تا زمانی ادامه میدهد که تعداد ویژگیها برابر یک تعداد معینی شود. مانند روش قبل مشکل این روش اینست که ویژگی حذف شده را دیگر به مجموعه اضافه نمیکند، حتی اگر مناسب باشد[23].

روشهای دیگری که در این گروه وجود دارند، نسخههای متفاوتی از دو روش قبلی یا ترکیب آنها هستند.

3) SBS-Slash

این روش بر پایه این مشاهده است که هنگامی که تعداد زیادی ویژگی وجود دارد، بعضی از طبقه بندی کنندهها (مانند ID3 یا C4.5) مکرراً تعداد زیادی از آنها را استفاده نمیکنند. الگوریتم با یک مجموعه ویژگی کار خودش را شروع میکند (مانند SBS)، اما بعد از یک مرحله تمام ویژگیهایی را که در این مرحله یاد گرفته است و استفاده نشدهاند، را حذف (Slashes) میکند[24].

4) PQSS ((p,q) Sequential Search)

در اینجا از بعضی از خواص بازگشت به عقب (Backtracking) استفاده میکنیم. عملکرد الگوریتم به این صورت است که در هر مرحله p ویژگی به مجموعه اضافه و q ویژگی از آن حذف میکند. حال اگر الگوریتم از مجموعه خالی شروع کرده باشد، بایستی اندازه p بزرگتر از اندازه q باشد. ولی اگر از مجموعه تمام ویژگیها شروع شده باشد، بایستی اندازه p کوچکتر از q باشد[25].

5) BDS (Bi-Directional Search)

مانند روشهای قبل است با این تفاوت که جستجو را از دو طرف انجام میدهد[25].

6) Schemata Search

الگوریتم کارش را با مجموعه خالی و یا مجموعه تمام ویژگیها شروع میکند و در هر تکرار، بهترین زیرمجموعه را با حذف یا اضافه تنها یک ویژگی به مجموعه ویژگی، پیدا میکند. برای اینکه هر زیرمجموعه را ارزیابی کند، از تعیین اعتبار Leave-One-Out Cross Validation (LOOCV) استفاده میکند. در هر تکرار زیرمجموعهای انتخاب میشود که کمترین خطای LOOCV را داشته باشد. کار به اینصورت ادامه مییابد تا هیچ تغییر با تک ویژگی نتواند باعث بهتر شدن زیرمجموعه شود[26].

7) RC (Relevance in Context)

در اینجا این واقعیت تشریح شده است که بعضی از ویژگیها فقط در قسمتی از فضای کار مربوط هستند. روش کار مشابه SBS است، اما با تغییرات عمدهای که باعث محلی شدن آن شده است(انتخاب ویژگیهای مرتبط بر اساس تصمیم گیری بوسیله نمونهها)[27] .

8) Queiros and Gelsema

شبیه SFS است اما پیشنهاد میکند که در هر تکرار، هر ویژگی در با تنظیمات متفاوتی بوسیله اثرات

متفاوت ناشی از ویژگیهای قبلی ارزیابی شود[28]. دو نمونه از این تنظیمات به اینصورت هستند:

i) همیشه فرض کنیم که ویژگیها مستقل هستند (ویژگیهای قبلی را در نظر نمیگیریم).

ii) هیچگاه فرض نمیکنیم که ویژگیها مستقل هستند (ویژگیهای قبلی را در نظر میگیریم).

در این روش و تعدادی از روشهای قبلی در این گروه از نرخ خطای بیز به عنوان خطای طبقه بندی کننده استفاده میکنیم.

3-2-3-9- تابع ارزیابی مبتنی بر خطای طبقه بندی کننده - تابع تولید کننده کامل

در این گروه چهار روش وجود دارد، که دو روش اول آن بوسیله Ichino و Sklansky ارائه شده است.

1) Linear Classifier

2) Box Classifier

در دو روش فوق مساله انتخاب ویژگی بوسیله برنامه نویسی صفر و یک حل شده است[29,30,31].

3) AMB&B (Approximate monotonic branch and bound)

این روش برای حل مشکلات B&B ارائه شده است، به این صورت که به تابع ارزیابی اجازه داده میشود که غیر یکنوا باشد[32]. در اینجا به تابع تولید کننده اجازه داده میشود که زیرمجموعههایی تولید کند که محدودیت تعیین شده را نقض میکنند، اما زیرمجموعهای که بعنوان جواب انتخاب میشود، نباید محدودیت ذکر شده را نقض کرده باشد.

4) BS (Beam Search)

این روش یک نمونه از جستجوی Best-First، است که از صف محدود شده برای محدود کردن فضای جستجو استفاده میکند. صف از بهترین به بدترین مرتب میشود، در اینصورت، بهترین زیرمجموعه در ابتدای صف قرار داده میشود. تابع تولید کننده به این صورت عمل میکند که زیرمجموعه موجود در ابتدای صف را انتخاب و کلیه زیرمجموعههای ممکن با اضافه کردن یک ویژگی به آن را تولید میکند و آنها را در محل مناسبشان در صف قرار میدهد. در صورتی که هیچ محدودیتی در اندازه صف نداشته باشیم، این روش یک جستجوی جامع است. در حالتی که محدودیت طول برابر یک را برای صف داشته باشیم، این روش با SFS برابر است.

3-2-3-10- تابع ارزیابی مبتنی بر خطای طبقه بندی کننده - تابع تولید کننده تصادفی

در این گروه پنج روش وجود دارد، که به شرح ذیل میباشند.

1) LVW

این روش زیرمجموعههایی به صورت کاملاً تصادفی با استفاده از الگوریتم Las Vegas تولید میکند[33].

2) GA (Genetic Algorithm)

در این روش یک جمعیت از زیرمجموعههای کاندید تولید میکنیم. در هر بار تکرار الگوریتم، با استفاده از عملگرهای جهش و بازترکیبی بر روی عناصر جمعیت قبلی، عناصر جدیدی تولید میکنیم. با استفاده از یک تابع ارزیابی، میزان شایستگی عناصر جمعیت فعلی را مشخص کرده و عناصر بهتر را به عنوان

جمعیت نسل بعد انتخاب میکنیم. پیدا شدن بهترین جواب در این روش تضمین نمیشود ولی همیشه یک جواب خوب به نسبت مدت زمانی که به الگوریتم اجازه اجرا داده باشیم، پیدا میکند.

3) SA (Simulated Annealing)

در اینجا نیز مانند الگوریتم ژنتیک، تابع تولید کننده آن از تولید تصادفی استفاده میکند ولی در تولید تصادفی از یک جریان خاصی پیروی میکند.

4) RGSS (Random Generation plus Sequential Selection)

این روش مشابه SBS و SFS است با این تفاوت که یک زیرمجموعه تصادفی تولید میکند و سپس SBS و SFS را با استفاده از این زیرمجموعه تولید شده اجرا میکند. در واقع فاکتور تصادف را به دو روش ذکر شده تزریق میکند، تا کارآئی آنها را افزایش دهد.

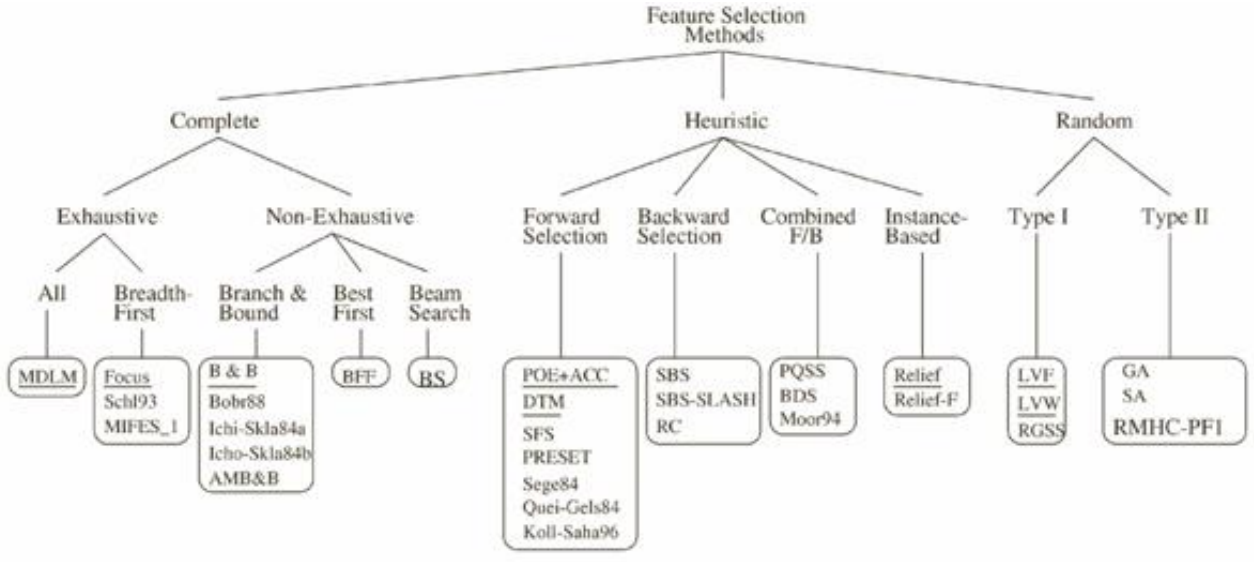
5) RMHC-PF1 (Random Mutation Hill Climbing-Prototype and Feature selection)

نمونههای اولیه (Prototype) و ویژگیها در اینجا همزمان برای استفاده در طبقه بندی کننده نزدیکترین همسایه انتخاب میشوند، همچنین برای ثبت نمونههای اولیه و ویژگیها از یک بردار شرطی استفاده میشود. تابع ارزیابی نیز، نرخ خطای طبقهبندی کننده نزدیکترین همسایه میباشد. در هر تکرار، بصورت تصادفی یکی از بیتهای بردار جهش داده میشوند، تا یک بردار جدید برای تکرار بعدی تولید شود.

تمام روشهای این گروه پارامترهای زیادی دارند که بایستی تنظیم شود، مثلاً LVW حد آستانهای برای نرخ ناسازگاری، در الگوریتمهای ژنتیک اندازه جمعیت اولیه، نرخ بازترکیبی و نرخ جهش و یا در SA، تعداد تکرار حلقه، دمای اولیه و احتمال جهش. تنظیم دقیق این پارامترها عملکرد این الگوریتمها را بهبود می-بخشد.

3-2-4- جمع بندی روشهای انتخاب ویژگی

برای اینکه یک جمعبندی از کلیه روشهای انتخاب ویژگی داشته باشیم، نمودار آنها را برحسب سه نوع تابع تولید کننده در شکل زیر نشان دادهایم[6].



شکل21 - طبقه بندی روشهای مختلف انتخاب ویژگی