فصل هشتم

انتخاب مشخصه‌ها

1-6: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 طبقه‌بندی) اهمیت بسزائی دارد، زیرا در این کاربردها تعداد زیادی ویژگی وجود دارند که بسیاری از آنها یا بلا استفاده هستند و یا 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 می-شود 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 از قدمت سی یا چهل ساله برخوردارند. مشکل بعضی از الگوریتم‌ها در زمانی که ارائه شده بودند، بار محاسباتی زیاد آنها بود، اگر چه امروزه با ظهور کامپیوترهای سریع و منابع ذخیره‌سازی بزرگ این مشکل به چشم نمی‌آید ولی از طرف دیگر، مجموعه داده‌های 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 دسته‌بندی می‌کنیم. سپس تعدادی از روش‌های معرفی شده در هر دسته را معرفی و بر اساس اهمیت، تا جائی‌که مقدور باشد، آنها را تشریح و الگوریتم برخی از آنها را ذکر می‌کنیم. لازم به ذکر است که به دلیل اینکه مبحث انتخاب ویژگی به مبحث طبقه‌بندی بسیار نزدیک است، بعضی از مسائلی که در اینجا مطرح می‌شود مربوط به مبحث طبقه‌بندی می‌باشد. توضیحات ارائه‌شده برای الگوریتم‌های مختلف در حد آشنائی است. شما می‌توانید برای کسب اطلاعات بیشتر به منابع معرفی شده مراجعه کنید.

2-6: تعاریف

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

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 تابع معیار[1] 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، هدف انتخاب ویژگی، کاهش اندازه ساختار بدون کاهش قابل ملاحظه در دقت پیشگوئی طبقه-بندی‌کننده‌ای که با استفاده از ویژگی‌های داده شده بدست می‌آید، می‌باشد.

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.

---

[1] Criterion Function

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 می‌شود، 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 دسته‌های مختلفی تقسیم‌بندی 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 بخش‌های زیر تقسیم می‌کنیم:

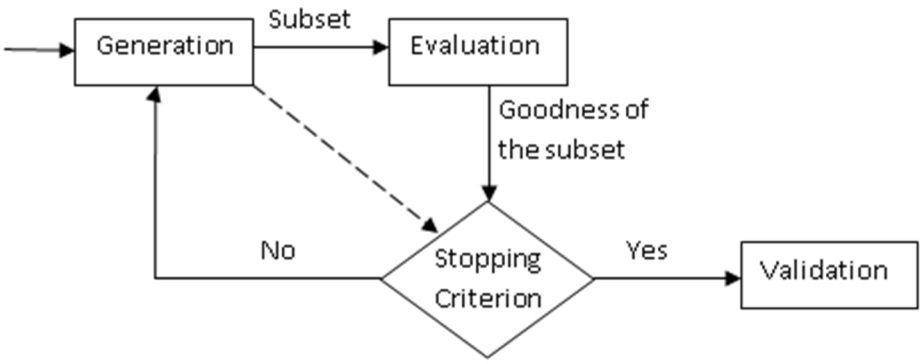
1. **تابع تولید کننده[2]**: این تابع زیر مجموعه‌های کاندید را برای روش مورد نظر پیدا می‌کند.
2. **تابع ارزیابی[3] :** زیرمجموعه مورد نظر را بر اساس روش داده شده، ارزیابی و یک عدد به عنوان میزان خوبی روش باز می‌گرداند. روش‌های مختلف سعی در یافتن زیرمجموعه‌ای دارند که این مقدار را بهینه کند.

---

[2] Generation procedure
[3] Evaluation function

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

4. **تابع تعیین اعتبار[4] :** تصمیم می‌گیرد که آیا زیر مجموعه انتخاب شده معتبر است یا خیر؟



شکل 6-1 فرآیند انتخاب ویژگی

تابع تولیدکننده در واقع تابع جستجو است. این تابع زیرمجموعه‌های مختلف را به ترتیب تولید می‌کند، تا به‌وسیله تابع ارزیابی، مورد ارزیابی قرا بگیرد. تابع تولیدکننده از یکی از حالت‌های زیر شروع به کار می‌کند:

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.

[4] Validation procedure

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 خوب‌بودن زیرمجموعه موردنظر بازمی‌گرداند. این مقدار با بهترین زیرمجموعه قبلی مقایسه می‌شود. 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 تولیدکننده 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) هر زمان که اضافه یا حذف کردن ویژگی، زیرمجموعه بهتری را تولید نکند
2) هر زمان که 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 داده‌های شبیه‌سازی شده استفاده کنیم.

3-6: روش‌های مختلف انتخاب ویژگی

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

1-3-6: توابع تولیدکننده

اگر تعداد کل ویژگی‌ها برابر N باشد، تعداد کل زیرمجموعه‌های ممکن برابر 2^N می‌شود. این تعداد حتی برای N های متوسط 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.

1-1-3-6: 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 "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 که از مرتبه $O(2^N)$ می‌باشد، این روش‌ها باعث می‌شوند که فضای کمتری جستجو شود. روش‌ها و تکنیک‌های مختلفی برای این‌کار استفاده شده‌اند، بعضی از آنها از تکنیک بازگشت به عقب[5] نیز در جریان کار استفاده کرده‌اند، مانند: branch and bound، best first search و beam search.

6-3-1-2: جستجوی مکاشفه ای

در روش‌هایی 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 پیاده‌سازی آن‌ها نیز بسیار ساده است.

6-3-1-3: 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 تولیدکننده تعدادی از زیرمجموعه‌های ممکن از فضای جستجو را به صورت تصادفی انتخاب می‌کند و در اختیار تابع ارزیابی قرار می‌دهد. تابع تولیدکننده تصادفی پارامتر-هایی دارد که بایستی تنظیم شوند، تنظیم مناسب این پارامترها در سرعت رسیدن به جواب و یافتن جواب‌های بهتر موثر است.

---

[5] Backtracking

2-3-6: تابع ارزیابی

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

به‌طور کلی توابع ارزیابی را می‌توان به 5 گروه تقسیم کرد :

1. **معیارهای مبتنی بر فاصله**[6]: در این معیارها، مثلا برای یک مساله دو کلاسه، یک ویژگی یا یک مجموعه ویژگی مانند X بر یک ویژگی یا یک مجموعه ویژگی دیگر مانند Y ارجحیت دارد، اگر 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**[7]: این معیارها میزان اطلاعاتی را که به‌وسیله یک ویژگی به‌دست می‌آید، درنظرمی‌گیرند. ویژگی X در این روش‌ها بر ویژگی Y اولویت دارد، اگر اطلاعات به‌دست آمده از ویژگی X بیشتر از اطلاعاتی باشد، که از ویژگی Y به‌دست می‌آید. نمونه‌ای از این معیارها همان معیار آنتروپی می‌باشد.

3. **معیارهای مبتنی بر وابستگی**[8]: این معیارها که با عنوان معیارهای همبستگی[9] نیز شناخته می‌شوند، قابلیت پیشگوئی مقدار یک متغیر به‌وسیله یک متغیر دیگر را اندازه‌گیری می‌کنند. ضریب[10] یکی از معیارهای وابستگی کلاسیک است که می‌توان از

---

[6] Distance Measures
[7] Information Measures
[8] Dependence Measures
[9] Correlation
[10] Coefficient

آن برای یافتن همبستگی بین یک ویژگی و یک کلاس استفاده کرد. 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 ویژگی‌های دیگر را اندازه‌گیری کنیم. این مقدار درجه افزونگی این ویژگی را نشان می‌دهد. همه توابع ارزیابی بر پایه معیار وابستگی را می‌توانیم به دو دسته معیارهای مبتنی بر فاصله و اطلاعات تقسیم کنیم. اما به خاطر این‌که این روش‌ها از یک وجه دیگر به مساله نگاه می‌کنند، این کار را انجام نمی‌دهیم.

4. معیارهای مبتنی بر سازگاری[11]: این معیارها جدیدتر هستند و اخیرا توجه بیشتری به آنها شده است. این معیارها خصوصیات متفاوتی نسبت به سایر معیارها دارند، زیرا به شدت به داده‌های آموزشی تکیه دارند و برای انتخاب یک زیرمجموعه از ویژگی‌ها تمایل دارند که مجموعه ویژگی‌های کوچکتری را انتخاب کنند. این روش‌ها زیرمجموعه‌هایی با کمترین اندازه را بر اساس از دست دادن یک مقدار قابل قبول سازگاری که توسط کاربر تعیین می‌شود، پیدا می‌کنند.

5. معیارهای مبتنی بر خطای طبقه‌بندی‌کننده[12]: 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 نسبتا زیاد است.

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

---

[11] Consistency Measures
[12] Classifier Error Rate Measures

1. **عمومیت**[13]: اینکه بتوان زیرمجموعه انتخاب شده را برای طبقه‌بندی‌کننده‌های 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. به‌جز خطای طبقه‌بندی‌کننده، دقت سایر توابع ارزیابی به مجموعه داده مورد استفاده و طبقه‌بندی‌کننده‌ای 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

جدول6-1 مقایسه توابع ارزیابی مختلف

6-3-3: دسته بندی و تشریح الگوریتم های مختلف انتخاب ویژگی

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

[13] Generality

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.

1-3-3-6: تابع ارزیابی مبتنی بر فاصله - تابع تولیدکننده مکاشفه ای

مهم‌ترین روش در این گروه Relief است. در اینجا ما ابتدا این روش را به عنوان نماینده این گروه شرح می‌دهیم، سپس یک مرور مختصر 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 نمونه‌ها 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[14] و نزدیکترین شکست[15] را بر اساس معیار اقلیدسی پیدا می‌کند.

[14] Nearest Hit
[15] Nearest Miss

نزدیکترین برخورد نمونه‌ای است که کمترین فاصله اقلیدسی را در میان سایر نمونه‌های هم-کلاس 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 به‌علاوه یک ویژگی خوب آن است که اختلاف بین اندازه آن ویژگی و نزدیکترین شکست وی بیشتر باشد. دلیل کار خیلی ساده است، ویژگی‌هایی که به خوبی دو کلاس (یا یک کلاس از سایر کلاس‌ها) را از هم تمیز می‌دهند، برای نمونه‌های متعلق به دو کلاس متفاوت، مقادیری نزدیک به‌هم نمی‌دهند بلکه یک فاصله معنی‌دار، 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 می‌کند، 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 آستانه است، حذف می‌کند و سایر ویژگی‌ها را به‌عنوان زیرمجموعه ویژگی جواب باز می‌گرداند. مقدار حد آستانه توسط کاربر تعیین می‌گردد. البته ممکن است که بصورت اتوماتیک بوسیله 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 2-6 الگوریتم Relief

الگوریتم Relief برای ویژگی‌های دارای نویز یا ویژگی‌های دارای همبستگی، خوب کار می‌کند و پیچیدگی زمانی آن به‌صورت 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 می‌توان 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 مرتفع شده است، با الگوریتم جدید مشکل داده‌های غیر کامل (نمونه‌های آموزشی غیرکامل) نیز حل شده است.

روش دیگری که در این زمینه برای انتخاب ویژگی مطرح است، از یک تابع ارزیابی استفاده می‌کند که مجموع یک معیار اختلاف آماری و یک معیار پیچیدگی ویژگی را محاسبه کرده و آن‌را مینیمم می‌کند. این الگوریتم، اولین ویژگی را که بهتر بتواند کلاس‌ها را از هم تمیز دهد پیدا می‌کند. سپس ویژگی‌هایی را پیدا می‌کند که در ترکیب با ویژگی‌های انتخاب شده، جدائی-

پذیری کلاس‌ها را افزایش دهند. این فرآیند زمانی متوقف می‌شود که به حداقل معیار بازنمائی مورد انتظار برسیم.

2-3-3-6: تابع ارزیابی مبتنی بر فاصله - تابع تولیدکننده کامل

استفاده از این ترکیب در روش‌های قدیمی نظیر B&B[16] یافت می‌شود. سایر روش‌های این گروه، نسخه‌های متفاوتی از B&B هستند. به این ترتیب که از یک تابع تولیدکننده دیگر استفاده کرده‌اند (BFF) و یا این‌که از یک تابع ارزیابی متفاوت استفاده کرده‌اند (Bobrowski's method). در اینجا ابتدا به شرح B&B می‌پردازیم و سپس یک شرح مختصر در مورد دو روش دیگر ارائه می‌دهیم.

تعریف کلاسیک ارائه شده در این زمینه از انتخاب ویژگی، احتیاج دارد که تابع ارزیابی یکنوا باشد. یعنی اگر دو زیرمجموعه ویژگی A و B با اندازه‌های M و N موجود و $B \subset A$ باشد 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 می‌گوئیم: 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 در نظر بگیرد. مانند بهترین زیرمجموعه ویژگی سه عنصری)

[16] (Branch and Bound

به‌وسیله 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 مجموعه‌های تک‌عنصری (یا تعداد ویژگی‌های تعیین شده به‌عنوان کران) برسیم. یعنی برگ‌های درخت مجموعه‌های تک‌عنصری هستند و ریشه درخت یک مجموعه شامل همه ویژگی‌های موجود می‌باشد.

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 قطعا تابع ارزیابی مقدار کمتری را برای آن‌ها باز می‌گرداند.عموماً توابع ارزیابی زیر برای این‌کار استفاده می‌شوند:

- فاصله ماهالانوبیس[17]
- تابع جداساز[18]
- معیار فیشر[19]
- فاصله باتاچاریا [۲۰]
- Divergence

یک الگوریتم مشابه برای انتخاب ویژگی، BFF است، در این الگوریتم، تابع جستجو مشابه حل مساله جستجوی یک مسیر بهینه در یک درخت وزن‌دار، و با یک استراتژی تغییر یافته از Best

[17] Mahalanobis Distance
[18] Discriminant Function
[19] (Fisher Criterion
[20] Bhattacharya

first search عمل می‌کند. این الگوریتم تضمین می‌کند که بهترین هدف، (زیرمجموعه بهینه) بدون از دست دادن جامعیت و با ارضای معیار یکنوا بودن تابع ارزیابی مساله، پیدا شود.

```
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 6-3 الگوریتم Branch and Bound

6-3-3-3: 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، 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 به‌عنوان جواب مساله باز می‌گرداند.

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:

$$(,) = \frac{-}{+} \log \frac{}{+} - \frac{}{+} \log \frac{}{+}$$

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

$$( ) = \frac{+}{+} (,) + \frac{+}{+} (,)$$

الگوریتم درخت تصمیم به صورت زیر است :

( )
(1) = ∅
(2) 4.5 h ,
(3) h
(4)

شکل6-4 الگوریتم درخت تصمیم

2) روش استفاده شده توسط Koller و Sahami

روش استفاده شده توسط Koller و Sahami که اخیرا ارائه شده است، بر این پایه استوار است که ویژگی‌هایی که داده مفید چندانی را در بر ندارند و یا اصلا داده مفیدی را در اختیار قرار نمی‌دهند و می‌توان آن‌ها را با سایر ویژگی‌ها نمایش داد، یا ویژگی-هایی 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 به‌صورت مشروط هم از کلاس و هم از سایر ویژگی‌هایی که در T نیستند، مستقل باشد.

6-3-3-4: تابع ارزیابی مبتنی بر اطلاعات- تابع تولید کننده کامل

مهمترین روشی که در این گروه می‌توانیم پیدا کنیم، روشMDLM[۲۱] است. محققان در این روش تلاش می‌کنند تا همه ویژگی‌های بدون استفاده (بی‌ربط یا اضافی) را حذف نمایند، در این صورت اگر ویژگی‌های زیرمجموعه V را بتوانیم به‌صورت یک تابع ثابت مانند F که وابسته به کلاس نیست و بر اساس یک زیرمجموعه ویژگی دیگر مانند U، بیان کنیم آن‌گاه 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، یک 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 MDLC [۲۲] استفاده کرده‌اند. این روش شامل تعداد بیتهای لازم برای انتقال کلاس‌ها، پارامترهای بهینه‌سازی، ویژگی‌های مفید و ویژگی‌های غیرمفید است. الگوریتم تمام زیرمجموعه‌های ممکن (2N) را جستجو می‌کند و به‌عنوان خروجی زیرمجموعه‌ای را بازمی‌گرداند که معیار MDLC را ارضا کند. این روش می‌تواند تمام ویژگی‌های مفیدی را پیدا کند که دارای توزیع نرمال باشند.

---

[21] Minimum Description Length Method
[22] Minimum Description Length Criterion

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 6-5 الگوریتم روش (MDLM)

6-3-3-5: تابع ارزیابی مبتنی بر وابستگی-تابع تولید کننده مکاشفه‌ای

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، ویژگی است که مجموع وزن‌دار P_e و میانگین ضریب همبستگی (ACC) با ویژگی(های) انتخاب‌شده 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 می‌شوند.

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 6-6 الگوریتم (POE1ACC**)** Probability Of Error & Average Correlation Coefficient

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

6-3-3-6: تابع ارزیابی مبتنی بر سازگاری - تابع تولیدکننده کامل

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

Focus (1

این روش یک حداقل‌گرا 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 را که بتواند سازگاری را با حداقل تعداد ویژگی ممکن برقرار کند به‌عنوان جواب بازمی‌گرداند.

در ساده‌ترین پیاده‌سازی 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 یادشده 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.

(,)
(1) =
(2) = 0
h
h h
=

شکل 6-7 الگوریتم روش Focus

2) Schlimmer

این روش از یک شمارش سیستماتیک برای تابع تولیدکننده 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 شده، نسبتی از درصد مقادیری است که زیاد مشاهده شده‌اند.

MIFES1 (3

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 وجود 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 این است که فقط برای مسائل دو کلاسه و ویژگی‌های منطقی قابل استفاده است.

6-3-3-7؛ تابع ارزیابی مبتنی بر سازگاری - تابع تولیدکننده تصادفی

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

این روش برای هر زیرمجموعه کاندید، تعداد ناسازگاری را محاسبه می‌کند و بر این ایده استوار است که کلاس محتمل‌تر آن است که در میان نمونه‌های این زیرمجموعه ویژگی، تعداد بیشتری متعلق به آن کلاس باشند. یک حد آستانه برای ناسازگاری در نظر گرفته می‌شود، که در ابتدا ثابت و به‌صورت 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 6-8 الگوریتم روش **LVF**

این روش می‌تواند هر زیرمجموعه بهینه را حتی برای داده‌های دارای نویز پیدا کند، ، 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 به‌دست آوردن یک زیرمجموعه بهینه منتظر بماند، زیرا الگوریتم هر زیرمجموعه‌ای که بهتر از بهترین جواب قبلی باشد (هم از لحاظ اندازه زیرمجموعه انتخاب‌شده 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 ویژگی-های درون آن کمتر یا مساوی بهترین زیرمجموعه‌ای است که تا کنون پیدا شده است. پیاده‌سازی 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 تولیدکننده مکاشفه‌ای استفاده می‌کنند، نیاز دارد، چون این الگوریتم از دانش مربوط به زیرمجموعه‌های قبلی استفاده نمی‌کند.

6-3-3-8: تابع ارزیابی مبتنی بر خطای طبقه‌بندی کننده-تابع تولید کننده مکاشفه ای

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

1) SFS [23]

این روش، 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، این است که ویژگی اضافه‌شده در صورتی‌که مناسب نباشد، از مجموعه جواب حذف نمی‌شود.

2) SBS [24]

این روش برعکس SFS با مجموعه‌ای شامل تمام ویژگی‌ها شروع می‌کند و در هر بار تکرار الگوریتم، ویژگی که به‌وسیله 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 این است که ویژگی حذف شده را دیگر به مجموعه اضافه نمی‌کند، حتی اگر مناسب باشد.

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

---

[23] Sequential Forward Selection
[24] Sequential Backward Selection

3) SBS-Slash

این روش بر پایه این مشاهده است که هنگامی که تعداد زیادی ویژگی وجود دارد، بعضی از طبقه‌بندی‌کننده‌ها (مانند ID3 یا C4.5) مکررا تعداد زیادی از آنها را استفاده نمی‌کنند. الگوریتم با یک مجموعه ویژگی شروع می‌کند (مانند SBS)، اما بعد از یک مرحله تمام ویژگی‌هایی را که در این مرحله یاد گرفته است و استفاده نشده‌اند، را حذف[25] می‌کند.

4) PQSS[26]

در اینجا از بعضی از خواص بازگشت به عقب استفاده می‌کنیم. عملکرد الگوریتم به این صورت است که در هر مرحله p ویژگی را به مجموعه اضافه و q ویژگی از آن را 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.

5) BDS[27]

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

6) Schemata Search

الگوریتم 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 می-

---

[25] Slashes
[26] (p,q) Sequential Search
[27] Bi-Directional Search

کند. برای اینکه هر زیرمجموعه را ارزیابی کند، از تعیین اعتبار LOOCV[۲۸] 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.

7) RC[29]

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

8) Queiros and Gelsema

این الگوریتم شبیه SFS است اما پیشنهاد می‌کند که در هر تکرار، هر ویژگی با تنظیمات متفاوتی بوسیله اثرات متفاوت ناشی از ویژگی‌های قبلی ارزیابی شود. 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 طبقه‌بندی‌کننده استفاده می‌کنیم.

---

[28] Leave-One-Out Cross Validation (

[29] Relevance in Context

6-3-3-9: تابع ارزیابی مبتنی بر خطای طبقه بندی‌کننده-تابع تولیدکننده 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 به‌وسیله برنامه نویسی صفر و یک حل‌شده است.

3) **AMB&B**[30]

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

4) **BS**[31]

این روش یک نمونه از جستجوی Best-First است که از صف محدودشده 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 تولیدکننده 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

[30] Approximate monotonic branch and bound
[31] Beam Search

محل مناسب خود 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.

6-3-3-10: 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 کاملا تصادفی با استفاده از الگوریتم Las Vegas تولید می‌کند.

2) GA[32]

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[33]

در اینجا نیز مانند الگوریتم ژنتیک، تابع تولیدکننده آن از تولید تصادفی استفاده می-کند ولی در تولید تصادفی از یک جریان خاصی پیروی می‌نماید.

---

[32] Genetic Algorithm
[33] Simulated Annealing

4) RGSS[34]

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 کارایی آنها را افزایش دهد.

5) RMHC-PF1[35]

نمونه‌های اولیه و ویژگی‌ها در اینجا به‌طور همزمان برای استفاده در طبقه‌بندی‌کننده نزدیک‌ترین همسایه، انتخاب می‌شوند، همچنین برای ثبت نمونه‌های اولیه و ویژگی‌ها از یک بردار شرطی استفاده می‌شود. تابع ارزیابی نیز، نرخ خطای طبقه‌بندی‌کننده نزدیکترین همسایه می‌باشد. در هر تکرار، به‌صورت تصادفی یکی از بیتهای بردار جهش داده می‌شوند تا یک بردار جدید برای تکرار بعدی تولید شود.

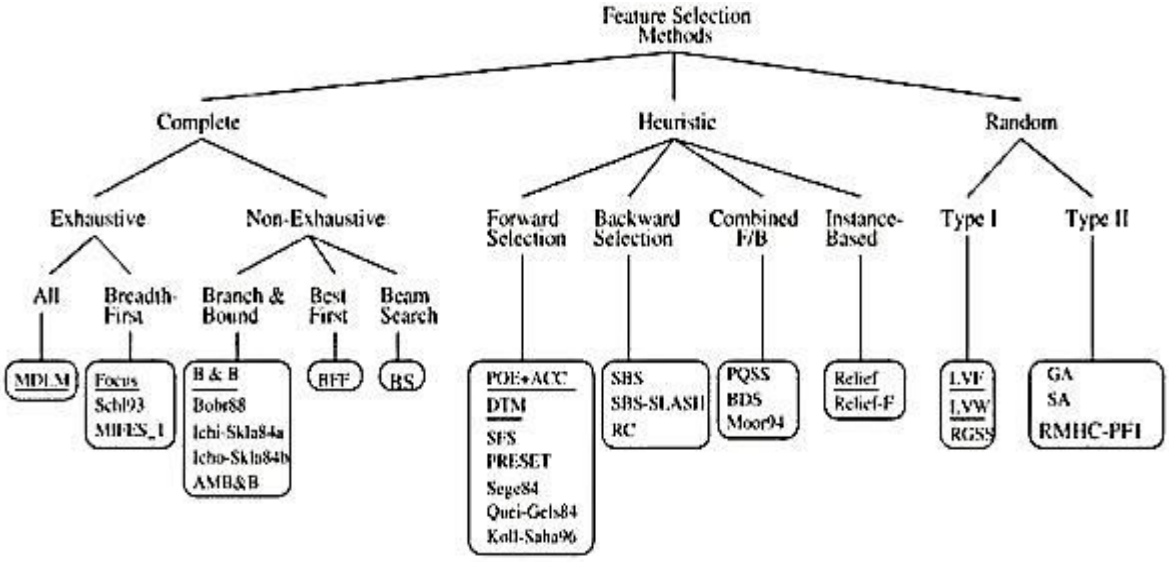
تمام روش‌های این گروه پارامترهای زیادی دارند که بایستی تنظیم شود، مثلا در LVW حد آستانه برای نرخ ناسازگاری، در الگوریتم‌های ژنتیک اندازه جمعیت اولیه، نرخ بازترکیبی و نرخ جهش و یا در SA، تعداد تکرار حلقه، دمای اولیه و احتمال جهش بایستس تنظیم شوند. تنظیم دقیق این پارامترها عملکرد این الگوریتم‌ها را بهبود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

---

[34] Random Generation plus Sequential Selection
[35] Random Mutation Hill Climbing-Prototype and Feature selection

4-3-6: جمع بندی روش‌های انتخاب ویژگی

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



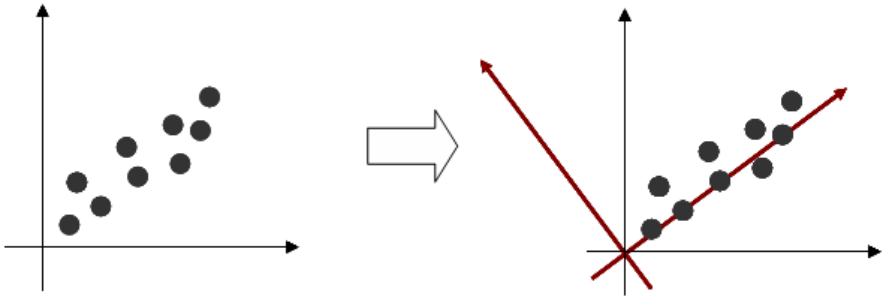
شکل 6-9 طبقه بندی روش‌های مختلف انتخاب ویژگی

4-6: PCA[36]

تکنیک PCA بهترین روش برای کاهش ابعاد داده به صورت خطی می‌باشد. یعنی با حذف ضرایب کم‌اهمیت به‌دست آمده از این تبدیل، اطلاعات از دست رفته نسبت به روش‌های دیگر کمتر است. البته کاربرد PCA محدود به کاهش ابعاد داده، نمی‌شود و در زمینه‌های دیگری مانند شناسایی الگو و تشخیص چهره نیز مورد استفاده قرار می‌گیرد. در این روش محورهای

[36] Principal Component Analysis

مختصات جدیدی برای داده‌ها تعریف شده و داده‌ها براساس این محورهای مختصات جدید بیان می‌شوند. اولین محور باید در جهتی قرار گیرد که واریانس داده‌ها ماکزیمم شود (یعنی در جهتی که پراکندگی داده‌ها بیشتر است). دومین محور باید عمود بر محور اول و به همین ترتیب محورهای بعدی عمود بر تمامی محورهای قبلی به گونه‌ای قرار می‌گیرند که داده‌ها در آن جهت دارای بیشترین پراکندگی باشند. در شکل زیر این مطلب برای داده‌های دو بعدی نشان داده شده است.



شکل 6-10 انتخاب محورهای جدید برای داده‌های دو بعدی

روش PCA به نام‌های دیگری نیز معروف است. مانند:

- SVD[37]
- KLT[38]
- Hotelling Transform
- EOF[39]

[37] Singular Value Decomposition
[38] Karhunen Loeve Transform
[39] Empirical Orthogonal Function

قبل از این‌که به جزئیات این روش بپردازیم ابتدا مفاهیم ریاضی و آمار مرتبط با این روش را به‌طور مختصر بیان می‌کنیم. این مفاهیم شامل انحراف از معیار استاندارد، کواریانس، بردارهای ویژه و مقادیر ویژه می‌باشد.

1-4-6: مفاهیم مقدماتی مورد نیاز در PCA

1-1-4-6: مفاهیم آماری

فرض کنید X رشته‌ای از مقادیر است. میانگین این مقادیر از رابطه زیر به‌دست می‌آید.

$$= \frac{\sum}{}$$

انحراف از معیار نیز از رابطه زیر محاسبه می‌شود.

$$= \sqrt{\frac{\sum_{=1} - ^2}{(-1)}}$$

علت اینکه در مخرج رابطه فوق از عبارت n-1 استفاده شده (و نه n) این است که فرض شده X شامل تمام مقادیر موجود نیست. بلکه تعدادی از این مقادیر انتخاب شده‌اند و در X قرار گرفته‌اند. یعنی X مجموعه نمونه است و شامل کل داده‌ها نمی‌باشد. با این فرض اگر از n-1 در رابطه فوق استفاده شود، انحراف از معیار به‌دست آمده به انحراف از معیار داده‌های واقعی نزدیک‌تر خواهد بود نسبت به این‌که از n استفاده شود. با بتوان 2 رساندن انحراف از معیار، واریانس به‌دست می‌آید.

$$= \sqrt{\frac{\sum_{=1} - ^2}{(-1)}}$$

معیارهایی که در بالا ذکر شد فقط اطلاعات مربوط به یک بُعد را ارائه می‌کنند و دانشی در مورد ارتباط بین ابعاد مختلف به ما نمی‌دهند. با استفاده از کواریانس می‌توانیم ارتباط بین ابعاد

مختلف داده‌ها را پیدا کنیم. فرض کنید یک رشته دیگر از اعداد داریم که آن را با Y نشان می-دهیم. کواریانس بین X و Y از رابطه زیر بدست می‌آید.

$$(,) = \frac{\sum (-)(-)}{(- 1)}$$

مقداری که از رابطه بالا بدست می‌آید در بازه [-1,1] قرار خواهد داشت که یکی از سه حالت زیر را به‌وجود می‌آورد:

- اگر مقدار به‌دست آمده مثبت باشد آنگاه X و Y با هم افزایش یا کاهش می‌یابند.
- اگر مقدار به‌دست آمده منفی باشد آنگاه با افزایش X مقدار Y کاهش می‌یابد و بالعکس.
- اگر مقدار به‌دست آمده صفر باشد آنگاه X و Y از یکدیگر مستقلند.

کواریانس بین تمامی ابعاد داده‌ها را می‌توان دوبه‌دو محاسبه کرده و در یک ماتریس ذخیره کرد. به این ماتریس، ماتریس کواریانس می‌گویند. ماتریس کواریانس یک ماتریس مربعی متقارن است. مثلا اگر سه بعد به نامهای x، y و z داشته باشیم، ماتریس کواریانس آنها برابر است با:

$$= \begin{matrix} (,) & (,) & (,) \\ (,) & (,) & (,) \\ (,) & (,) & (,) \end{matrix}$$

2-1-4-6: مفاهیم جبر ماتریسها

در این بخش مفهوم بردار ویژه و مقادیر ویژه را بیان می‌کنیم. همانطور که می‌دانید برای این‌که بتوان دو ماتریس را در یکدیگر ضرب کرد، آن دو باید از نظر اندازه با هم سازگار باشند. بردارهای ویژه نوع خاصی از ضرب ماتریسها را ارائه می‌کنند. به مثال زیر توجه کنید.

$$\begin{matrix}2 & 3\\ 2 & 1\end{matrix} \times \begin{matrix}1\\ 3\end{matrix} = \begin{matrix}11\\ 5\end{matrix}$$

$$\begin{matrix}2 & 3\\ 2 & 1\end{matrix} \times \begin{matrix}3\\ 2\end{matrix} = \begin{matrix}12\\ 8\end{matrix} = 4 \times \begin{matrix}3\\ 2\end{matrix}$$

در مثال اول بردار به‌دست آمده مضرب صحیحی از بردار اولیه نیست. اما در مثال دوم، بردار به‌دست آمده چهار برابر بردار اولیه می‌باشد. ماتریس 2×2 که در این دو بردار ضرب کرده‌ایم را می‌توان یک ماتریس تبدیل[40] در نظر گرفت که با ضرب آن در یک بردار می‌توان اندازه و راستای آن بردار را تغییر داد. در میان تمام بردارهایی که می‌توان ماتریس تبدیل را در آنها ضرب کرد، بردارهایی وجود دارند که پس از تبدیل، راستای آنها تغییر نمی‌کند و فقط اندازه آنها ممکن است عوض شود، مانند بردار [3;2] در مثال فوق. این بردارها را بردارهای ویژه می‌نامند. برای هر بردار ویژه یک مقدار ویژه نیز وجود دارد که بیان می‌کند اندازه آن بردار (و تمام بردارهای دیگر که در راستای آن بردار هستند) پس از تبدیل، چند برابر خواهد شد. در مثال فوق مقدار ویژه برای بردار [3;2] و البته تمام بردارهای هم‌راستا با آن مانند [6;4] برابر با 4 می‌باشد.

بردارهای ویژه و مقادیر ویژه فقط برای ماتریس‌های مربعی معنی پیدا می‌کنند. یک ماتریس n×n می‌تواند دارای n بردار ویژه باشد. به منظور استانداردکردن بردارهای ویژه، پس از یافتن بردارهای ویژه اندازه آن‌ها را به گونه‌ای تغییر می‌دهند تا طول آن‌ها برابر با یک شود. مثلا برای بردار [3;2] داریم:

$$\overline{(3 \quad + 2\ )} = \sqrt{13} \qquad \begin{matrix}3\\ 2\end{matrix} \div \overline{\qquad 13} = \begin{matrix}3/\sqrt{13}\\ 2/\sqrt{13}\end{matrix}$$

ویژگی مهم بردارهای ویژه این است که آن‌ها برهم عمودند. مثلا ماتریس تبدیل زیر را در نظر بگیرید.

[40] Transformation matrix

$$= \begin{matrix} -1 & 0 \\ 0 & 1 \end{matrix}$$

با ضرب ماتریس A در هر بردار می‌توان قرینه آن بردار نسبت به خط $y=0$ را به‌دست آورد. بردارهای ویژه این ماتریس عبارتند از [1;0] و [0;1]. مقادیر ویژه این بردارها نیز به ترتیب 1- و 1 می‌باشند. همانطور که گفتیم این دو بردار ویژه برهم عمودند.

به‌دست آوردن بردارهای ویژه برای ماتریس‌های بزرگتر از 3×3 کار نسبتا دشواری است. این کار توسط یک الگوریتم بازگشتی انجام می‌شود.

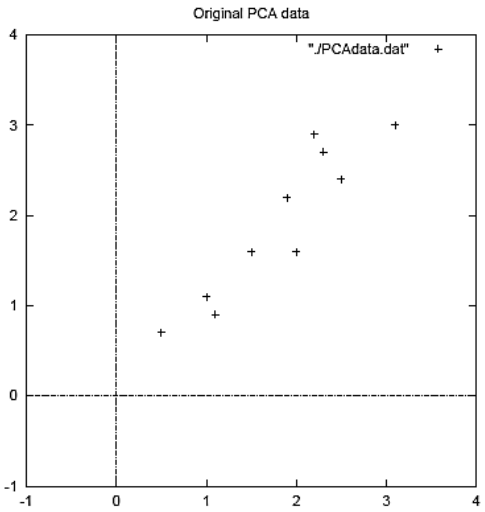
2-4-6: الگوریتم PCA

در این بخش الگوریتم PCA را با ذکر یک مثال توضیح می‌دهیم.

مرحله 1- انتخاب داده

در اینجا ما قصد داریم PCA را بر روی یک مجموعه داده دو بعدی اعمال کنیم. این داده‌ها در شکل زیر نشان داده شده است.

x	y
2.5	2.4
0.5	0.7
2.2	2.9
1.9	2.2
3.1	3.0
2.3	2.7
2	1.6
1	1.1
1.5	1.6
1.1	0.9



شکل 6-8 داده‌های دوبعدی اولیه که قرار است PCA برروی آنها اعمال شود

مرحله 2- کم کردن میانگین از داده‌ها

در این مرحله، میانگین هر بُعد را از مقادیر آن بُعد کم می‌کنیم تا میانگین داده‌ها در هر بُعد صفر شود.

مرحله 3- محاسبه ماتریس کواریانس

ماتریس کواریانس را به طریقی که در بالا ذکر شد برای داده‌ها به‌دست می‌آوریم. برای مثال ما این ماتریس، یک ماتریس 2×2 است:

$$= \begin{matrix} .616555556 & .615444444 \\ .615444444 & .716555556 \end{matrix}$$

مرحله 4- محاسبه بردارهای ویژه و مقادیر ویژه

اکنون بردارهای ویژه و مقادیر ویژه ماتریس کواریانس را محاسبه می‌کنیم. ماتریس کواریانس، یک ماتریس نیمه قطعی مثبت متقارن[41] است. طبق قضایای جبر خطی، یک ماتریس متقارن n×n دارای n بردار ویژه مستقل و n مقدار ویژه می‌باشد. همچنین یک ماتریس نیمه قطعی مثبت، دارای مقادیر ویژه غیر منفی است. این مقادیر برای مثال ما برابر است با:

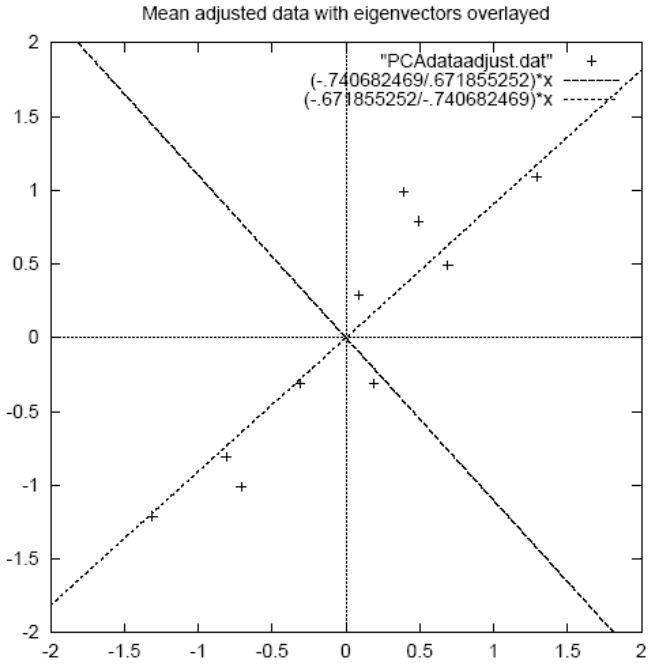
$$= \begin{matrix} -0490833989 \\ 1.28402771 \end{matrix}$$

$$= \begin{matrix} -735178656 & -.677873399 \\ .677873399 & -.735178656 \end{matrix}$$

توجه داشته باشید که این دو بردار ویژه به گونه‌ای انتخاب شده‌اند که طول هر دو برابر با 1 باشد. اما این دو بردار چه چیزی به ما می‌دهد؟ ما راستای این دو بردار را در شکل 9-6 نشان داده‌ایم. همانطور که می‌بینید یکی از این دو بردار در جهتی قرار گرفته است که داده‌ها در آن جهت بیشترین پراکندگی را دارند. بردار دیگر نیز عمود بر بردار اول است. اما مقادیر ویژه چه چیزی را ارائه می‌دهند؟ در این مثال برداری که در راستای بیشترین پراکندگی داده‌ها قرار گرفته است دارای مقدار ویژه 1.284 و بردار دیگر دارای مقدار ویژه 0.049 می‌باشد. در واقع مقادیر ویژه، میزان پراکندگی داده‌ها در راستای بردار ویژه مربوطه را نشان می‌دهد. می‌توان گفت بردار ویژه‌ای که دارای بزرگترین مقدار ویژه است مؤلفه اصلی[42] داده‌های موجود می‌باشد.

---

[41] Symmetric positive semidefinite matrix
[42] Principle component



شکل 6-11 داده‌های نرمال‌سازی شده (با کم شدن میانگین) به همراه بردارهای ویژگی ماتریس کواریانس

مرحله 5- انتخاب مؤلفه‌ها و ساختن بردار ویژگی

در این مرحله مفهوم کاهش ابعاد داده وارد می‌شود. بردارهای ویژه‌ای که در مرحله قبل به‌دست آوردیم را بر اساس مقادیر ویژه آنها از بزرگ به کوچک مرتب می‌کنیم (توجه داشته باشید که مقادیر ویژه ماتریس کواریانس همگی بزرگتر یا مساوی صفر هستند). بدین ترتیب مؤلفه‌های داده‌ها از پر اهمیت به کم اهمیت مرتب می‌شوند. در اینجا اگر بخواهیم ابعاد داده‌ها را کاهش دهیم می‌توانیم مؤلفه‌های کم اهمیت را حذف نماییم. البته این کار با از دست دادن مقدار کمی اطلاعات همراه است.

کاری که باید در این مرحله انجام دهیم ایجاد یک بردار ویژگی است که در واقع ماتریسی از بردارها می‌باشد. این ماتریس شامل بردارهای ویژگی است که ما می-خواهیم آن‌ها را نگه داریم.

$$= (\quad , \quad , \quad , \dots , \quad)$$

اگر همه بردارهای ویژگی را در این ماتریس قرار دهیم، هیچ اطلاعاتی از دست نخواهد رفت و دوباره می‌توانیم دقیقا همان داده‌های اولیه را به‌دست آوریم. در ادامه مثال فوق، یک بردار ویژگی را برابر با مقدار زیر در نظر می‌گیریم.

$$= \begin{matrix} -.677873399 \\ -.735178656 \end{matrix}$$

مرحله 6- بدست آوردن داده‌های جدید

در آخرین مرحله از PCA فقط باید ترانهاده ماتریس بردار ویژگی که در مرحله قبل به‌دست آوردیم را در ترانهاده داده‌های نرمال‌سازی‌شده ضرب نماییم.

$$= \qquad\qquad \times$$

که RowFeatureVector ماتریسی است که بردارهای ویژه در سطرهای آن به ترتیب مقادیر ویژه از بالا به پایین قرار گرفته‌اند و RowDataAdjust ماتریسی است که شامل داده‌هایی است که میانگین هر بُعد از آن بُعد کم شده است. در این ماتریس، داده‌ها در ستون‌های آن ذخیره شده و هر سطر آن مربوط به یک بُعد است. در مثال ذکر شده به‌دلیل اینکه ما فقط یکی از بردارهای ویژه را انتخاب کردیم داده‌های به‌دست آمده از PCA، داده‌های یک بُعدی می‌باشند.

x
-.827970186
1.77758033
-.992197494
-.274210416
-1.67580142
-.912949103
.0991094375
1.14457216
.438046137
1.22382056

شکل 6-12 داده‌های بدست آمده از تبدیل PCA با انتخاب مهم‌ترین بردار ویژگی

با استفاده از رابطه زیر می‌توانیم مقادیری که از تبدیل PCA بدست آورده‌ایم را به داده‌های اولیه که مقدار میانگین از آنها کم شده بازگردانیم.

= ×

به‌دلیل اینکه ماتریس حاوی بردارهای ویژه یکه است، معکوس آن با ترانهاده آن برابر است. بنابراین:

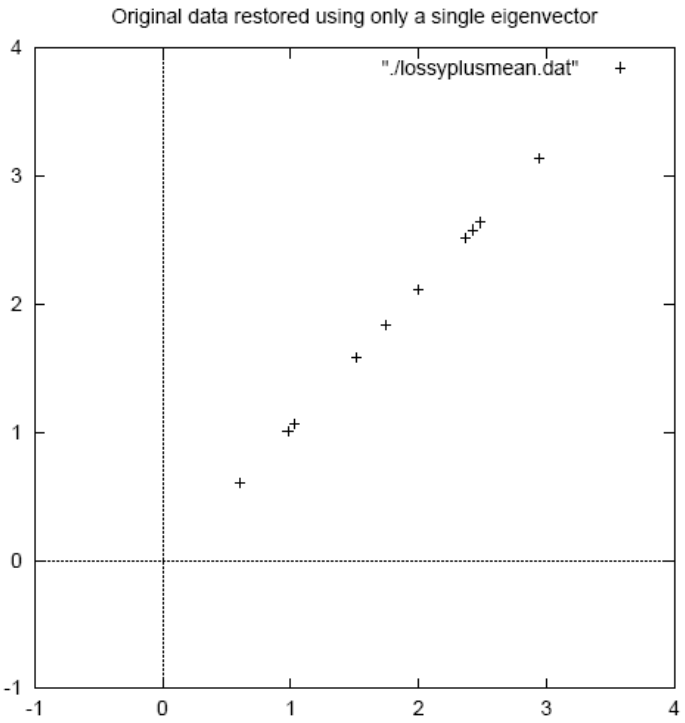
= ×

با اضافه کردن میانگین، داده‌های اولیه را خواهیم داشت:

=

(×) +

در شکل زیر داده‌هایی که پس از تبدیل PCA بازیابی شده‌اند را مشاهده می‌کنید. همانطور که می‌بینید مقدار کمی اطلاعات از بین رفته است و باعث شده داده‌ها همگی در امتداد یک خط راست قرار گیرند.



شکل 6-13 داده‌های بازیابی شده از تبدیل PCA با انتخاب مهم‌ترین بردار ویژگی

6-4-3: آنالیز تفکیک پذیری خطی (LDA)

بدون تردید یکی از پرطرفدارترین و باسابقه ترین گرایش‌های تحقیقاتی در چند سال اخیر استفاده از نسخه‌های مختلف بسط آنالیز تفکیک‌پذیری خطی یا به اختصارLDA است. در بسط LDA هدف تولید فضایی است که در آن پراکندگی درون کلاسی مینیمم و پراکندگی برون

کلاسی ماکزیمم باشد. اما مقصود از پراکنـدگی درون کلاسـی و بـرون کلاسـی چیسـت؟ بطـور شهودی چنانچه داده‌های هر کلاس متمرکز و توده‌های کلاسی تـا حـد امکـان از یکـدیگر دور باشند طراحی یک ماشین یادگیری برای طبقه‌بندی داده‌ها با سهولت بیشـتری همـراه بـوده و ماشین یادگیری از پتانسیل بالاتری برای طبقه‌بندی داده‌ها برخوردار اسـت. پراکنـدگی درون کلاسی (رابطه1-4) شاخصی برای میزان تمرکز داده‌ها در هر کلاس و پراکندگی برون کلاسـی (رابطه1-5) شاخصی برای میزان دور بودن توده‌های کلاسی از یکدیگر است.

$$= -\sum \quad \sum_{\in} (\quad - \quad)(\quad - \quad) \qquad (4\text{-}1)$$

$$= -\sum \quad (\quad - \quad)(\quad - \quad) \qquad (5\text{-}1)$$

در رابطه1-4 و رابطه1-5 فرض کرده‌ایم که با یک مسأله شناسایی الگـو بـا N داده و c کـلاس مواجه هستیم. در این روابط mi و m به ترتیب میانگین تمام داده‌ها و میانگین داده‌هـای درون کلاس i هستند. Ni تعداد داده‌های درون کلاس i را نشان مـی دهـد. در بسـط LDA بـرای آن‌که دو شاخص پراکندگی درون و بـرون کلاسـی بـه طـور همزمـان و بـه ترتیـب مینـیمم و ماکزیمم شوند باید رابطه (6-1) که شاخص فیشر نامیده میشود ماکزیمم شود. می‌توان نشـان داد که نگاشتW از کنار هم چیدن بردارهای ویژه ماتریس $S_w^{-1}S_b$ بدست می‌آید.

$$= \arg\max \ \left| \frac{\quad}{\quad} \right| \qquad (6\text{-}1)$$

نسخه های جدیدتر الگوریتم نیز با تغییر در معیار فیشر حاصل شده انـد. بـه عنـوان مثـال در الگوریتم F-LDA ماتریس پراکندگی برون کلاسـی را بـا ترکیبـی وزن‌دار محاسـبه مـی کننـد. استدلال استفاده از ترکیب وزن‌دار، جریمه‌کردن کلاس‌هایی است که تفکیـک‌پـذیری کمتـری داشته و مراکز آن‌ها به یکدیگر نزدیک‌تر است. با دادن وزن در محاسبه ماتریس پراکندگی برون کلاسی، نقش این کلاس‌ها در تولید فضای نهایی کمتر می‌شود.

نسخه معروف و جدید دیگری از الگوریتم LDA با نام Regularized LDA توسط لو و همکاران در سال 2005 میلادی با اصلاح معیار فیشر مطابق با رابطه (7-1) معرفی شد.

$$= \qquad \frac{| \qquad |}{| \quad | د | \quad |} \qquad (7\text{-}1)$$

اگرچه باور عمومی بر این است که کارایی بسط LDA همواره از بسط PCA بهتر است اما مشاهدات مختلفی گزارش شدندکه عکس این باور را در برخی سناریوها نشان می دادند. بعدها مارتینز و کاک در یک قالب جامع ریاضی نشان دادند که همواره نمی‌توان انتظار داشت بسط LDA بهتر از بسط PCA عمل نماید.