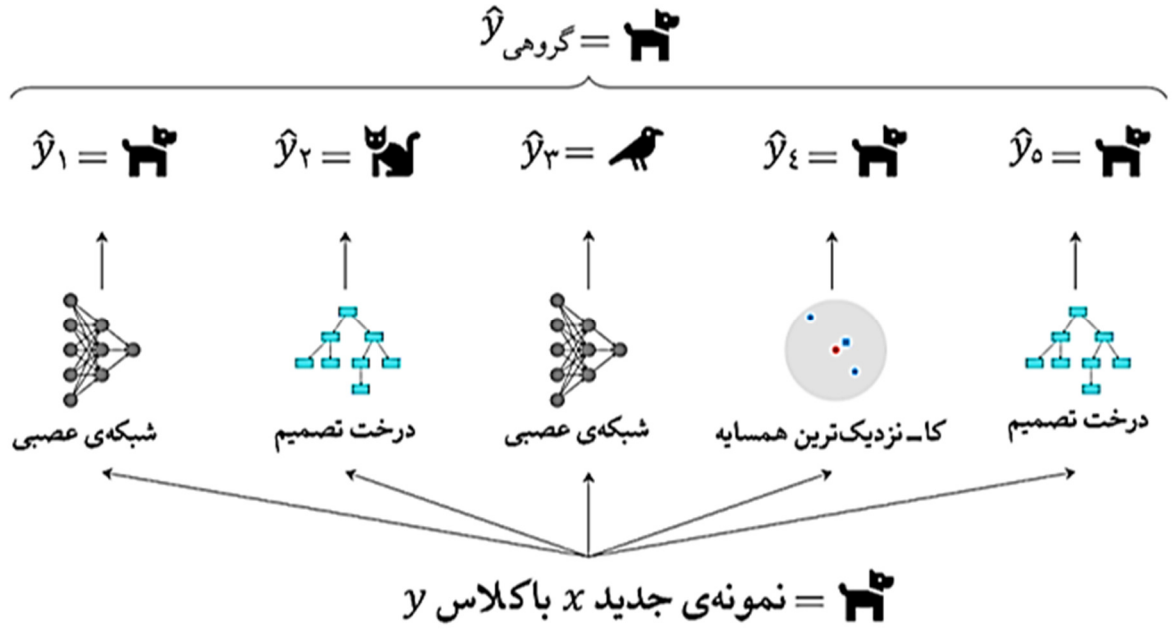
# مصطفی سبیلو - مهدی محمدی - پروژهٔ داده‌کاوی

## یادگیری گروهی

در بسیاری از مواقعی که نیاز به تصمیم‌گیری مهم داریم، این طبیعت ما است که ازنظر یک متخصص برای کمک به تصمیم‌گیری بهره ببریم. در بیشتر موارد، ما یک گام فراتر می‌رویم و به دنبال نظر دوم و سوم نیز هستیم. ازاین‌جهت که فکر می‌کنیم هیچ فردی به‌تنهایی نمی‌تواند در مورد موضوعی که ازنظر عینی دشوار است (مثلاً تشخیص پزشکی)، دانش کاملی داشته باشد. با در نظر گرفتن تعدادی از عوامل، ما سپس به‌طور شهودی، به‌نوعی توصیه‌های دریافت شده را برای اطلاع از تصمیم نهایی خود وزن کرده و ترکیب می‌کنیم. ما از همهٔ این تصمیم‌ها استفاده می‌کنیم، چراکه انتظار داریم این تصمیم بهتر از تصمیمی باشد که خودمان می‌گیریم.

علیرغم مهارت ما در حل مسائل، انسان‌ها هنوز از مشورت با منابع متعدد سود می‌برند؛ بنابراین، طبیعی است که حوزه یادگیری ماشین از این عادت انسان الهام گرفته باشد، زمانی که به دنبال راه‌هایی برای بهبود مدل‌های خود می‌گردند. حوزه‌ای از یادگیری ماشین که شامل رویه‌هایی است تا با آموزش مدل‌های متعدد و روش‌های ترکیب خروجی‌های آن‌ها به بهبود عملکرد مدل دست یابد، به‌عنوان یادگیری گروهی شناخته می‌شود. به عبارت ساده‌تر، یادگیری گروهی، یعنی هنر استفاده از چندین مدل، برای به دست آوردن عملکرد پیش‌بینی بهتر.

اصل اساسی پشت مدل‌های گروهی، این است که گروهی از یادگیرنده‌های ضعیف گرد هم می‌آیند تا به کمک یکدیگر یک یادگیرنده قوی تشکیل دهند. شهود پشت مدل‌سازی گروهی، مترادف با چیزی است که ما در زندگی روزمره به آن عادت کرده‌ایم، مانند درخواست نظر از چندین متخصص قبل از اتخاذ یک تصمیم خاص به‌منظور به حداقل رساندن احتمال تصمیم بد یا نتیجه نامطلوب. یک سناریوی نمونه که در آن یک گروه سودمند تشکیل شده است در شکل ۱ قابل‌مشاهده است. در این مثال، ما ۵ مدل را بر روی‌داده‌های آموزشی یکسان آموزش می‌دهیم هر مدل به کلاس تصویر رأی می‌دهد و گروه، کلاسی که بیشترین رأی را دارد پیش‌بینی می‌کند.




شکل ۱ - مثالی از یادگیری گروهی.

هیچ مدل یادگیری ماشین کامل نیست. همهٔ آن‌ها محدودیت‌هایی دارند و اشتباه می‌کنند؛ همان‌طور که یک انسان ممکن است قبل از تصمیم‌گیری دشوار از چندین متخصص، مشاوره بخواهد. همچنین، نشان داده‌شده که ترکیب پیش‌بینی‌های چندین مدل، پیش‌بینی‌های بسیار دقیق‌تری را ارائه می‌دهد. بسیاری از مطالعات نظری و تجربی موقعیت‌هایی را شناسایی کرده‌اند که در آن سیستم‌های گروهی سودمند هستند. به‌طورکلی، سه دلیل وجود دارد که چرا یادگیری گروهی می‌تواند نتایج بهتری نسبت به یک یادگیرنده منفرد ایجاد کند:

۱. داده‌های آموزشی محدود: برای مسائل پیچیده، داده‌ها ممکن است اطلاعات کافی را برای همهٔ ویژگی‌های احتمالی داده‌هایی که یادگیرنده ممکن است با آن‌ها مواجه شود، ارائه نکند؛ بنابراین، یک فرضیهٔ منحصربه‌فرد برای یک مجموعه دادهٔ محدود، بعید است به دلیل اطلاعات ناکافی برای مجموعه دادهٔ دیگری کار کند. در داده‌های آموزشی با ترکیب فرضیه‌های یادگیرندگان، شانس بیشتری وجود دارد که یادگیرندهٔ گروهی با موفقیت، داده‌هایی با ویژگی‌های ناشناخته را مدیریت کند.

۲. جبران فرآیندهای جستجوی ناقص: فرآیندهای جستجوی الگوریتم‌های یادگیری، ممکن است ناقص باشند. ازاین‌رو، حتی اگر یک فرضیهٔ بهینهٔ منحصربه‌فرد وجود داشته باشد، ممکن است هرگز پیدا نشود. یادگیری گروهی می‌تواند به جبران چنین فرآیندهای جستجوی ناقصی کمک کند.

۳. هیچ فرضیهٔ بهینهٔ منحصربه‌فردی وجود ندارد: فضای مورد جستجو برای مجموعه دادهٔ معین، ممکن است حاوی یک فرضیهٔ بهینهٔ واحد نباشد. ازاین‌رو یادگیری گروهی می‌تواند تقریب‌های خوبی را با ترکیب فرضیه‌های متعدد ارائه دهد.

## تا یادگیری عمیق هست، چرا یادگیری گروهی؟!

پیشرفت‌های محاسباتی و روش‌شناختی اخیر، در آنچه که عموماً به‌عنوان یادگیری عمیق شناخته می‌شود، سودمندی درک شده از روش‌های یادگیری گروهی را تغییر داده است. با یک معماری کافی و تنظیم ابرپارمترها، شبکه‌های عصبی عمیق می‌توانند عملکردی نزدیک به کامل در طیف وسیعی از مجموعه داده‌ها داشته باشند. علاوه بر این، بسیاری از تدبیرهای منظم سازی برای اطمینان از همگرایی این مدل‌های پیچیده بدون بیش برازش در دسترس هستند.

ازاین‌رو، با این عملکرد فوق‌العادهٔ یادگیری عمیق، آیا روش‌های گروهی هنوز نقش مفیدی دارند؟! اول اینکه، آموزش شبکه‌های عصبی عمیق، هزینهٔ محاسباتی زیادی دارد. تنظیم ابرپارمترهای جامع برای این مدل‌ها به دلیل کمبود منابع محاسباتی به‌ندرت امکان‌پذیر است. بااین‌حال، یافتن معماری مناسب برای تولید مدل‌هایی با عملکرد پیش‌بینی بالا ضروری است. برخلاف این، یک سیستم گروهی می‌تواند عملکرد پیش‌بینی مشابهی با چنین مدل‌های منفرد ایجاد کند، اما بدون متحمل شدن هزینه‌های محاسباتی مشابه. ممکن است در نگاه اول غیرشهودی به نظر برسد، چراکه چندین مدل برای تولید یک گروه نیز، نیاز به آموزش دارند! بااین‌حال، مدل‌های فردی می‌توانند پیچیدگی کمتری باشند (لایه‌های کمتر) که امکان کاوش بهتر در فضای دسته‌بند را در همان بودجه محاسباتی فراهم می‌کند. علاوه بر این، بسیاری از این تنظیمات را می‌توان در سیستم نهایی گروهی (به‌جای انتخاب بهترین مدل) در نظر گرفت.

دوم اینکه، روش‌های منظم سازی به‌صورت کامل از بیش برازش جلوگیری نمی‌کنند؛ بنابراین، ترکیب مدل‌هایی که همگرا نشده‌اند، یا از مشکل بیش برازش داده‌ها رنج می‌برند، می‌توانند مدل‌های بهتری را بدون آموزش مجدد تولید کنند. درنهایت، هنگام کاوش مجموعه داده‌های دنیای واقعی، در بسیاری از مسابقات دیده می‌شود که سیستم‌های گروهی بهترین نتایج را ایجاد می‌کنند.

درحالی‌که شبکه‌های عصبی عمیق در برخی وظایف به عملکردی نزدیک به انسان دست‌یافته‌اند، واقعیت این است که این مدل‌ها بسیار تخصصی هستند. ازاین‌رو، استفاده از یک شبکهٔ عصبی که بتواند در هنگام تشخیص بین تصاویر گربه‌ها و سگ‌ها به‌دقت کامل برسد و آن را در یک مسئله متفاوت اما مرتبط (مثلاً شناسایی کلاس‌های مشابه بر اساس نمونه‌های ویدیویی) بکار

برد، عملکرد بسیار ضعیفی را در پی خواهد داشت. این به یک موضوع بسیار موردبحث معروف، به نام قضیه ناهار مجانی نیست مربوط می‌شود: هیچ مدل یادگیری ماشین به‌تنهایی، در همهٔ انواع داده‌ها و مجموعه داده‌ها بهترین عملکرد را نخواهد داشت. این نشان می‌دهد که مجموعه‌ای از مدل‌های بسیار تخصصی، راه‌حل بهتری برای این موضوع هستند.

## تکنیک‌های یادگیری گروهی

یادگیری گروهی یک الگوی یادگیری ماشین است که در آن چندین مدل که اغلب "یادگیرندهٔ ضعیف" نامیده می‌شوند برای حل یک مسئله آموزش داده می‌شوند و برای به دست آوردن نتایج بهتر ترکیب می‌شوند. فرضیهٔ اصلی این است که وقتی مدل‌های ضعیف به‌درستی ترکیب شوند، می‌توانیم مدل‌های دقیق‌تری/قوی‌تری به دست آوریم.

در نظریهٔ یادگیری گروهی، ما یادگیرندگان ضعیف (مدل‌های پایه) را مدل‌هایی می‌نامیم که می‌توانند با ترکیب چندین مدل از آن‌ها به‌عنوان بلوک‌هایی برای طراحی مدل‌های پیچیده‌تر استفاده شوند. اغلب اوقات، این مدل‌های پایه به‌خودی‌خود چندان خوب عمل نمی‌کنند به این دلیل که بایاس یا واریانس بالایی دارند. ازاین‌رو، ایده روش‌های گروهی این است که سعی کنیم بایاس یا واریانس چنین یادگیرندگان ضعیفی را با ترکیب چند مورد از آن‌ها، کاهش دهیم تا یک یادگیرندهٔ قوی (مدل گروهی) ایجاد کنیم که به عملکرد بهتری دست یابد.

برای ایجاد یک مدل یادگیری گروهی، ابتدا باید مدل‌های پایه خود را برای تجمع انتخاب کنیم. بیشتر اوقات از یک الگوریتم یادگیری پایه استفاده می‌شود تا یادگیرندگان ضعیف همگنی داشته باشیم که به روش‌های مختلف آموزش‌دیده‌اند. به این نوع مدل‌ها "همگن" گفته می‌شود.

بااین‌حال، روش‌هایی نیز وجود دارد که از انواع مختلف الگوریتم‌های یادگیری پایه استفاده می‌کنند. سپس برخی از یادگیرندگان ضعیف ناهمگن در یک "مدل مجموعه‌های ناهمگن" ترکیب می‌شوند.

یک نکته مهم این است که انتخاب یادگیرندگان ضعیف ما، باید با روش تجمع این مدل‌ها هماهنگ باشد. اگر مدل‌های پایه با بایاس کم اما واریانس بالا را انتخاب کنیم، باید با روش تجمیع‌کننده‌هایی باشد که تمایل به کاهش واریانس دارد، درحالی‌که اگر مدل‌های پایه با واریانس کم اما بایاس بالا را انتخاب کنیم، باید با روش تجمعی باشد که تمایل به کاهش بایاس دارد.

تکنیک‌های گروهی زیادی برای ترکیب چندین یادگیرندهٔ ماشین در راستای ایجاد یک مدل پیش‌گویانه وجود دارد. رایج‌ترین تکنیک‌ها عبارت‌اند از: تجمیع پردازی برای کاهش واریانس و توان‌افزایی برای کاهش بایاس.

## توان‌افزایی (Boosting)

توان‌افزایی، دسته‌ای از الگوریتم‌ها است که یادگیرندگان ضعیف را به یادگیرندگان قوی تبدیل می‌کند. الگوریتم توان‌افزایی با آموزش یک یادگیرندهٔ پایه شروع می‌شود و سپس توزیع نمونه‌های آموزشی را با توجه به نتیجهٔ یادگیرندهٔ پایه تنظیم می‌کنند تا نمونه‌های دسته‌بندی نادرست موردتوجه بیشتر یادگیرندگان پایهٔ بعدی قرار گیرد. پس از آموزش اولین یادگیرندهٔ پایه، یادگیرندهٔ پایه دوم با نمونه‌های آموزشی تنظیم‌شده آموزش می‌بیند و از نتیجه برای تنظیم مجدد توزیع نمونه آموزشی استفاده می‌شود. چنین فرآیندی تکرار می‌شود تا زمانی که تعداد یادگیرندگان پایه به مقدار از پیش تعریف‌شده T برسد و درنهایت این یادگیرندگان پایه وزن و ترکیب شوند.

شناخته‌شده‌ترین الگوریتم توان‌افزایی AdaBoost است که الگوریتم توان‌افزایی ساده را از طریق یک فرآیند تکراری بهبود می‌بخشد. ایدهٔ اصلی پشت این الگوریتم تمرکز بیشتر بر روی الگوهایی است که دسته‌بندی آن‌ها سخت‌تر است. مقدار تمرکز با وزنی که به هر الگوی مجموعه آموزشی اختصاص داده می‌شود، تعیین می‌شود. در ابتدا، وزن یکسانی به همهٔ الگوها اختصاص داده می‌شود. در هر تکرار وزن تمام نمونه‌های دسته‌بندی اشتباه افزایش می‌یابد درحالی‌که وزن نمونه‌های دسته‌بندی صحیح کاهش می‌یابد. درنتیجه، یادگیرندهٔ ضعیف مجبور می‌شود با انجام تکرارهای اضافی و ایجاد دسته‌بندهای بیشتر، بر روی نمونه‌های دشوار

مجموعه آموزشی تمرکز کند. علاوه بر این، وزنی به هر دسته‌بند اختصاص داده می‌شود. این وزن دقت کلی دسته‌بند را اندازه‌گیری می‌کند و تابعی از وزن کل الگوهای دسته‌بندی صحیح است؛ بنابراین، وزن‌های بالاتری به دسته‌بندهای دقیق‌تر داده می‌شود. از این وزن‌ها برای دسته‌بندی الگوهای جدید استفاده می‌شود.

از منظر تجزیه‌وتحلیل بایاس-واریانس، توان‌افزایی عمدتاً بر کاهش بایاس تمرکز دارد. به همین دلیل است که مجموعه‌ای از یادگیرندگان با توانایی تعمیم ضعیف می‌توانند بسیار قدرتمند باشند.

### تجمیع پردازی (Bagging)

تجمیع پردازی یک روش ساده و درعین‌حال مؤثر برای ایجاد مجموعه‌ای از دسته‌بندها است. دسته‌بند گروهی که با این روش ایجاد می‌شود، خروجی‌های دسته‌بندهای مختلف را در یک دسته‌بند واحد ادغام می‌کند. این منجر به یک دسته‌بند می‌شود که دقت آن بالاتر از دقت هر دسته‌بند منفردی است.

ایدهٔ پشت تجمیع پردازی، ترکیب نتایج چندین مدل برای به دست آوردن یک نتیجه کلی است. در اینجا یک سؤال وجود دارد: اگر همهٔ مدل‌ها را بر روی یک مجموعه داده ایجاد کنیم و آن‌ها را ترکیب کنید، مفید خواهد بود؟ احتمال زیادی وجود دارد که این مدل‌ها نتیجه یکسانی داشته باشند، زیرا ورودی یکسانی دارند. پس چگونه می‌توانیم این مشکل را حل کنیم؟ یکی از این راه‌ها ایجاد یادگیرندگان پایهٔ مختلف، با تقسیم مجموعهٔ آموزشی اصلی به چندین زیرمجموعه غیر همپوشان و استفاده از هر زیرمجموعه برای آموزش یک یادگیرندهٔ پایه است.

ازآنجایی‌که زیرمجموعه‌های آموزشی متفاوت هستند، احتمالاً یادگیرندگان پایهٔ آموزش‌دیده نیز متفاوت هستند. بااین‌حال، اگر زیرمجموعه‌ها کاملاً متفاوت باشند، به این معنی است که هر زیرمجموعه فقط بخش کوچکی از مجموعه آموزشی اصلی را در برمی‌گیرد که احتمالاً منجر به یادگیری ضعیفی می‌شود. ازآنجایی‌که یک گروه خوب مستلزم آن است که هر یادگیرندهٔ پایه به‌طور معقولی خوب باشد، ما اغلب اجازه می‌دهیم زیرمجموعه‌ها به‌گونه‌ای همپوشانی داشته باشند که هر یک از آن‌ها شامل نمونه‌های کافی باشد.

تجمیع پردازی بر اساس نمونه‌برداری بوت‌استرپ کار می‌کند. با توجه به مجموعه داده با $m$ نمونه، به‌طور تصادفی یک نمونه را انتخاب کرده و در مجموعه نمونه‌برداری رونوشت می‌کند. سپس، آن را در مجموعه داده‌های اصلی نگه می‌داریم به‌طوری‌که هنوز فرصتی برای برداشتن آن در دفعه بعد وجود داشته باشد. با تکرار $m$ مرتبهٔ این فرآیند یک مجموعه داده حاوی $m$ نمونه به دست می‌آید که در آن برخی از نمونه‌های اصلی ممکن است بیش از یک بار ظاهر شوند درحالی‌که برخی ممکن است هرگز ظاهر نشوند (تقریباً ۶۳٫۲% از نمونه‌های اصلی در مجموعه داده‌ها ظاهر می‌شوند).

اعمال فرآیند بالا در T مرتبه، T مجموعه داده را تولید می‌کند که هرکدام حاوی $m$ نمونه است. سپس، یادگیرندگان پایه بر روی این مجموعه داده‌ها آموزش دیده و ترکیب می‌شوند. چنین رویه‌ای، روند کار اولیهٔ تجمیع پردازی است. هنگام ترکیب پیش‌بینی‌های یادگیرندگان پایه، تجمیع پردازی روش رأی‌گیری ساده را برای وظایف دسته‌بندی و روش میانگین‌گیری ساده را برای وظایف رگرسیونی اتخاذ می‌کند. هنگامی‌که چندین کلاس تعداد آرای یکسانی را دریافت می‌کنند، می‌توانیم به‌طور تصادفی یکی را انتخاب کنیم یا اطمینان آرا را بیشتر بررسی کنیم.

نمونه‌گیری بوت‌استرپ مزیت دیگری را برای تجمیع پردازی به ارمغان می‌آورد: ازآنجایی‌که هر یادگیرندهٔ پایه فقط از ۶۳٫۲% از نمونه‌های آموزشی اصلی برای آموزش استفاده می‌کند، ۳۶٫۸% نمونه‌های باقیمانده (خارج از کیسه) را می‌توان به‌عنوان مجموعهٔ اعتبارسنجی برای به دست آوردن توانایی تعمیم استفاده کرد.

## زنگ تفریح

بیایید روش‌های گروهی را با استفاده از قیاس زیر تجسم کنیم: تصور کنید که باید در امتحانی شرکت کنیم که شامل ۱۰۰ سؤال درست/غلط در موضوعات مختلف ازجمله ریاضی، جغرافیا، علوم، تاریخ و موسیقی است. خوشبختانه، ما اجازه داریم با پنج دوست خود - آدریانا، باب، کارلوس، دانا و امیلی - تماس بگیریم تا به ما کمک کنند. یک محدودیت کوچک وجود دارد و آن این است که همه آن‌ها تمام‌وقت کار می‌کنند و وقت ندارند به همه ۱۰۰ سؤال پاسخ دهند، اما خوشحال هستند که در زیرمجموعه‌ای از آن‌ها به ما کمک می‌کنند. از چه تکنیک‌هایی می‌توانیم برای کمک گرفتن از آن‌ها استفاده کنیم؟ دو تکنیک ممکن به شرح زیر است:

تکنیک ۱: برای هر یک از دوستان، چندین سؤال تصادفی انتخاب کنید و از آن‌ها بخواهید که به آن‌ها پاسخ دهند (مطمئن شوید که هر سؤال حداقل توسط یکی از دوستان ما پاسخ داده شود). پس از دریافت پاسخ‌ها، با انتخاب گزینه‌ای که در بین کسانی که به آن سؤال پاسخ داده‌اند، بیشترین محبوبیت را داشت، به آزمون پاسخ دهید. به‌عنوان‌مثال، اگر دو نفر از دوستان ما در مورد سؤال ۱ "درست" و یکی "نادرست" پاسخ داد، ما به سؤال ۱ به‌عنوان "درست" پاسخ می‌دهیم (در صورت وجود مساوی، می‌توانیم یکی از پاسخ‌های برنده را به‌طور تصادفی انتخاب کنیم).

تکنیک ۲: امتحان را به آدریانا می‌دهیم و از او می‌خواهیم که فقط به سؤالاتی که در مورد آن‌ها مطمئن است پاسخ دهد. ما فرض می‌کنیم که آن پاسخ‌ها خوب هستند و آن‌ها را از آزمون حذف می‌کنیم. اکنون سؤالات باقی‌مانده را با همان دستورالعمل به باب می‌دهیم. ما به این روش ادامه می‌دهیم تا آن را به هر پنج دوست منتقل کنیم.

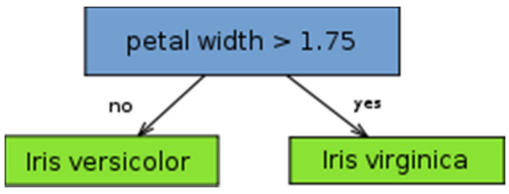
تکنیک ۱ شبیه یک الگوریتم Bagging است و تکنیک ۲ شبیه یک الگوریتم Boosting است. برای دقیق‌تر شدن، Bagging و Boosting از مجموعه‌ای از مدل‌ها به نام یادگیرندگان ضعیف استفاده می‌کنند و ترکیب آن‌ها به‌صورت یک یادگیرنده قوی عمل می‌کند.

## Decision Stump:

استامپ تصمیم یک مدل یادگیری ماشینی است که از درخت تصمیم یک سطحی تشکیل شده است. یعنی یک درخت تصمیم با یک گره داخلی (ریشه) است که بلافاصله به گره های پایانی (برگ های آن) متصل می شود. یک استامپ تصمیم بر اساس مقدار تنها یک ویژگی ورودی پیش بینی می کند. گاهی به آنها ۱-قواعد نیز گفته می شود. معادلهٔ یک استامپ تصمیم به شکل زیر قابل بیان است:

$$h(x; j, \theta)\begin{cases} +1 & x_j > \theta \\ -1 & else \end{cases}$$

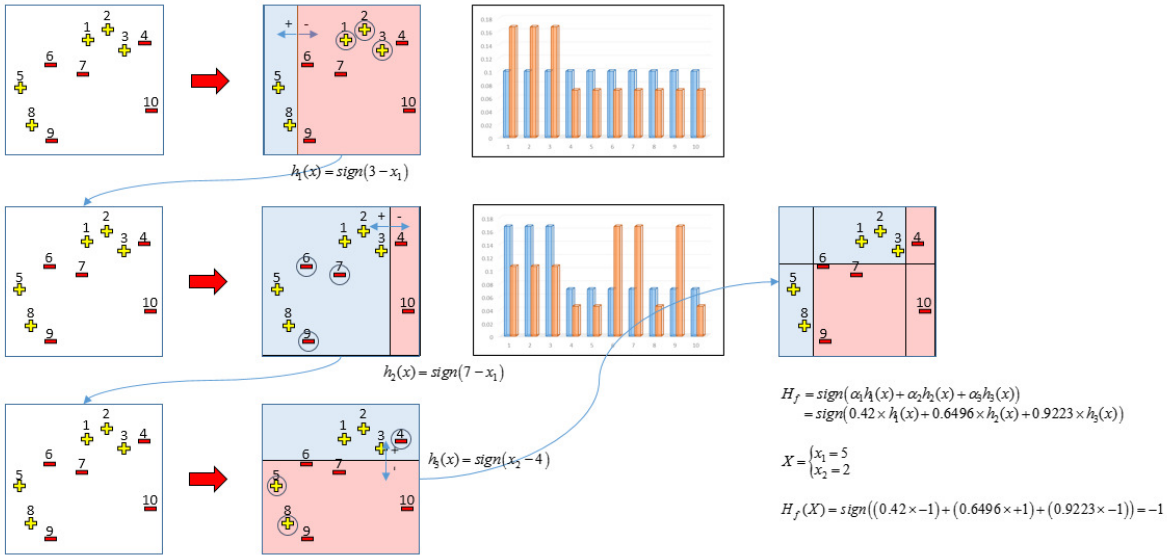
در شکل ۲ نمونه ای از یک استامپ تصمیم که بین دو تا از سه کلاس مجموعه داده گل زنبق تمایز قائل می شود به تصویر کشیده شده است.




شکل ۲ - تمایز دو کلاس از داده‌های گل زنبق با یک حد آستانه.

## مثالی شهودی:

در ابتدا احتمال انتخاب تمامی نمونه‌ها یکسان فرض می‌شود که در این مثال $1/10$ فرض شده است. سپس با استفاده از استامپ تصمیم $h_1(x)$ نمونه‌های برچسب خورده را جدا می‌کنیم. مشاهده می‌شود که سه نمونهٔ ۱، ۲ و ۳ در سمت اشتباهی قرار گرفته‌اند؛ بنابراین باید احتمال انتخاب این سه نمونه جهت استفاده در مرحلهٔ بعد افزایش پیدا کند و احتمال انتخاب سایر نمونه‌ها کاهش یابد. در مرحلهٔ بعد با استفاده از استامپ تصمیم $h_2(x)$ نمونه‌های برچسب خورده از جدا می‌شوند که مشاهده می‌شود سه نمونهٔ ۷، ۶ و ۹ در سمت اشتباهی قرارگرفته‌اند؛ بنابراین احتمال انتخاب این نمونه‌ها جهت استفاده در مرحلهٔ بعد افزایش یافته و احتمال انتخاب سایر نمونه‌ها کاهش می‌یابد. همین رویه برای مرحلهٔ سوم تکرار می‌شود. در انتها با ترکیب این سه ناحیه می‌توان به مرزهای جداکنندهٔ نمونه‌های آموزشی دست یافت. حال می‌توانیم با استفاده از علامت مجموع وزن‌دار خطوط تفکیک‌کننده، برچسب نمونه جدید را مشخص کنیم. درواقع بر اساس یک رأی‌گیری که نسبت مشارکت هر تفکیک‌کننده به‌واسطهٔ ضریب $\alpha_i$ مشخص خواهد شد.




## زنگ تفریح

برای درک این موضوع تصور کنید که ما سه دوست داریم: ترزا راست‌گو، اومبرت غیرقابل‌پیش‌بینی، و لنی دروغ‌گو. ترزا راست‌گو تقریباً همیشه حقیقت را می‌گوید، لنی دروغ‌گو تقریباً همیشه دروغ می‌گوید و اومبرت غیرقابل‌پیش‌بینی حقیقت را تقریباً نیمی از زمان می‌گوید و نیمه دیگر پنهان می‌کند. از بین این سه دوست، کدام‌یک کم مفید است؟

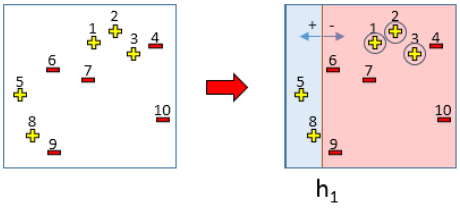
اونطوری که من می‌بینم ترزا راست‌گو خیلی قابل اعتماده چون اون تقریباً همیشه حقیقت رو می‌گه پس ما می‌تونیم بهش اعتماد کنیم در میان دو نفر دیگر، من لنی دروغ‌گو را ترجیح می‌دهم. اگر چه او تقریباً همیشه دروغ می‌گوید اما زمانی که ما از او یک سؤال بله یا خیر بپرسیم، ما به‌سادگی به‌عنوان حقیقت مخالف آنچه او به ما می‌گوید، و ما بیشتر اوقات درست خواهد شد! از سوی دیگر، اومبرت غیرقابل‌پیش‌بینی هیچ هدفی به ما خدمت نمی‌کند اگر ما هیچ تصوری از اینکه آیا او حقیقت را می‌گوید یا دروغ می‌گوید. در آن صورت، اگر ما به اختصاص نمره به آنچه هر دوست می‌گوید، من می‌خواهم راست‌گو ترزا نمره مثبت بالا، لنی دروغ‌گو نمره منفی بالا، و اومبرت غیرقابل‌پیش‌بینی نمره صفر است.

## نگاهی دقیق‌تر به محاسبات:

ابتدا مجموع وزن نمونه‌های به‌اشتباه دسته‌بندی‌شده محاسبه می‌شود ($J_1$). مقدار خطای $\varepsilon_1$ با تقسیم مقدار $J_1$ بر مجموع وزن تمامی نمونه‌ها محاسبه‌شده و ضریب دسته‌بند $h_1$ توسط رابطهٔ $\alpha_1 = 1/2 \ln(1-\varepsilon_1/\varepsilon_1)$ به دست می‌آید. پس‌ازآن وزن

نمونه‌هایی که به‌صورت نادرست دسته‌بندی شده‌اند با استفاده از رابطهٔ $W_{t+1,i} = W_{t,i} \times e^{+\alpha_1}$ افزایش و وزن نمونه‌هایی که به‌صورت درست دسته‌بندی شده‌اند با استفاده از رابطهٔ $W_{t+1,i} = W_{t,i} \times e^{-\alpha_1}$ کاهش می‌یابند. در انتها، تمامی وزن‌های به‌روزرسانی شده، جهت استفاده در مرحلهٔ بعد باید نرمالایز شوند. مرحلهٔ دوم و سوم به نحوی مشابه با مرحلهٔ اول انجام می‌پذیرند و در انتها با استفاده از ضرایب $\alpha_i$-های به‌دست‌آمده از مراحل اول تا سوم، می‌توان تابع $H_f = sign\left(0.42 \times h_1(x) + 0.6496 \times h_2(x) + 0.9223 \times h_3(x)\right)$ را به دست آورد.

## مرحلهٔ اول:




مجموع وزن داده‌های به اشتباه دسته‌بندی‌شده $J_1 = 0.1 + 0.1 + 0.1 = 0.3$

خطای مناسب $\varepsilon_1 = \frac{J_1}{\sum_{i=1}^{10} Wn_{t,i}} = \frac{0.3}{1.0} < 0.5$

ضریب دسته‌بند $h_1$ $\alpha_1 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_1}{\varepsilon_1}\right) = \frac{1}{2}\ln\left(\frac{1-0.3}{0.3}\right) = 0.42$

محاسبهٔ وزن‌های جدید:

دسته‌بندی نادرست $W_{t+1,i} = W_{t,i} \times e^{+\alpha_1}, i \in \{1,2,3\}$ $0.1 \times e^{+0.42} = 0.1528$

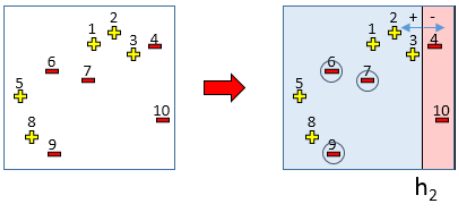
دسته‌بندی درست $W_{t+1,j} = W_{t,j} \times e^{-\alpha_1}, j \in \{4,5,6,7,8,9,10\}$ $0.1 \times e^{-0.42} = 0.0655$

نرمال کردن وزن‌ها $Wn_{t+1,j} = \frac{Wun_{t+1,j}}{Z_1}, j = 1 \ldots 10$ $Z_1 = \sum_{i=1}^{10} Wun_{t+1,i} = 3 \times 0.1528 + 7 \times 0.0655 = 0.9169$

$Wn_{t+1,1..3} = \frac{0.1528}{0.9169} = 0.1666$ $Wn_{t+1,4..10} = \frac{0.0655}{0.9169} = 0.0714$

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| وزن‌های اولیهٔ | 0.10 | 0.10 | 0.10 | 0.10 | 0.10 | 0.10 | 0.10 | 0.10 | 0.10 | 0.10 |
| وزن‌های جدید | 0.1528 | 0.1528 | 0.1528 | 0.0655 | 0.0655 | 0.0655 | 0.0655 | 0.0655 | 0.0655 | 0.0655 |
| نرمال کردن وزن‌ها | 0.1666 | 0.1666 | 0.1666 | 0.0714 | 0.0714 | 0.0714 | 0.0714 | 0.0714 | 0.0714 | 0.0714 |

## مرحلهٔ دوم:




مجموع وزن داده‌های به اشتباه دسته‌بندی‌شده $J_2 = 0.0714 + 0.0714 + 0.0714 = 0.2142$

خطای مناسب $\varepsilon_2 = \frac{J_2}{\sum_{i=1}^{10} Wn_{t,i}} = \frac{0.2142}{0.9996} < 0.5$

ضریب دسته‌بند $h_2$ $\alpha_2 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_2}{\varepsilon_2}\right) = \frac{1}{2}\ln\left(\frac{1-0.2143}{0.2143}\right) = 0.6496$

محاسبهٔ وزن‌های جدید:

دسته‌بندی نادرست $W_{t+1,i} = W_{t,i} \times e^{+\alpha_2}, i \in \{6,7,9\}$ $0.0714 \times e^{+0.6496} = 0.1367$

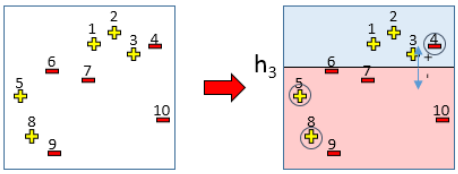
دسته‌بندی درست $W_{t+1,j} = W_{t,j} \times e^{-\alpha_2}, j \in \{1,2,3,4,5,8,10\}$ $0.0714 \times e^{-0.6496} = 0.0373$ $0.1666 \times e^{-0.6496} = 0.0870$

نرمال کردن وزن‌ها $Wn_{t+1,j} = \frac{Wun_{t+1,j}}{Z_2}, j = 1 \ldots 10$ $Z_2 = \sum_{i=1}^{10} Wun_{t+1,i} = 3 \times 0.0870 + 4 \times 0.0373 + 3 \times 0.1367 = 0.8203$

$Wn_{t+1,1..3} = \frac{0.0870}{0.8203} = 0.1061$ $Wn_{t+1,4,5,8,10} = \frac{0.0373}{0.8203} = 0.0455$ $Wn_{t+1,6,7,9} = \frac{0.1367}{0.8203} = 0.1666$

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| وزن‌های اولیهٔ | 0.1666 | 0.1666 | 0.1666 | 0.0714 | 0.0714 | 0.0714 | 0.0714 | 0.0714 | 0.0714 | 0.0714 |
| وزن‌های جدید | 0.0870 | 0.0870 | 0.0870 | 0.0373 | 0.0373 | 0.1367 | 0.1367 | 0.0373 | 0.1367 | 0.0373 |
| نرمال کردن وزن‌ها | 0.1061 | 0.1061 | 0.1061 | 0.0455 | 0.0455 | 0.1666 | 0.1666 | 0.0455 | 0.1666 | 0.0455 |

## مرحلهٔ سوم:




$J_3 = 0.0455 + 0.0455 + 0.0455 = 0.1365$ مجموع وزن داده‌های به اشتباه دسته‌بندی‌شده

$\varepsilon_3 = \frac{J_3}{\sum_{i=1}^{10} Wh_{t,i}} = \frac{0.1365}{1.0001} < 0.5$ خطای مناسب

$\alpha_3 = \frac{1}{2}\ln\left(\frac{1-\varepsilon_3}{\varepsilon_3}\right) = \frac{1}{2}\ln\left(\frac{1-0.1365}{0.1365}\right) = 0.9223$ ضریب دسته‌بند $h_3$

محاسبهٔ وزن‌های جدید:

$0.0455 \times e^{+0.9223} = 0.1144 \leftarrow W_{t+1,i} = W_{t,i} \times e^{+\alpha_3},\ i \in \{4,5,8\}$ ← دسته‌بندی نادرست

$0.1061 \times e^{-0.9223} = 0.0422$
$0.1666 \times e^{-0.9223} = 0.0662 \leftarrow W_{t+1,j} = W_{t,j} \times e^{-\alpha_3},\ j \in \{1,2,3,6,7,9,10\}$ ← دسته‌بندی درست
$0.0455 \times e^{-0.9223} = 0.0181$

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0.1061 | 0.1061 | 0.1061 | 0.0455 | 0.0455 | 0.1666 | 0.1666 | 0.0455 | 0.1666 | 0.0455 | وزن‌های اولیهٔ |
| 0.0422 | 0.0422 | 0.0422 | 0.1144 | 0.1144 | 0.0662 | 0.0662 | 0.1144 | 0.0662 | 0.0181 | وزن‌های جدید |

## شبه‌کد

Given: $(x_1; y_1), \ldots, (x_m; y_m), x_i \in X, y_i \in Y = \{-1, 1\}$.
Initialize $D_1(i) = 1/\mathrm{m}$.
For $t = 1 \ldots T$:

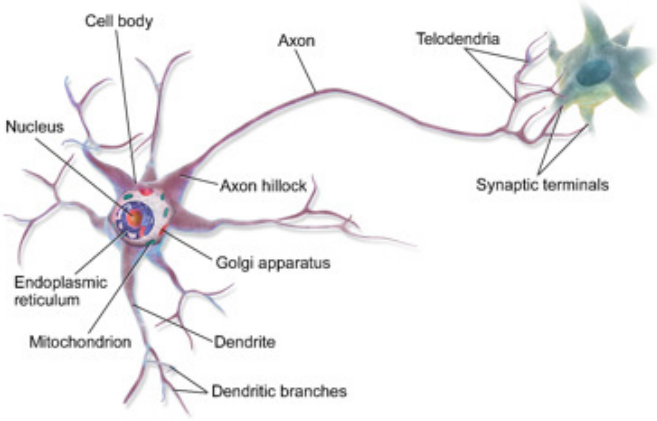
1. Train weak classifier using distribution $D_t$.
2. Get weak hypothesis $h_t : X \to \{-1, 1\}$ with error $\varepsilon_t = \sum_{i:h_t(x_i) \neq y_i} D_t(x_i)$
3. Choose $\alpha_t = \frac{1}{2}\log\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$
4. Update:

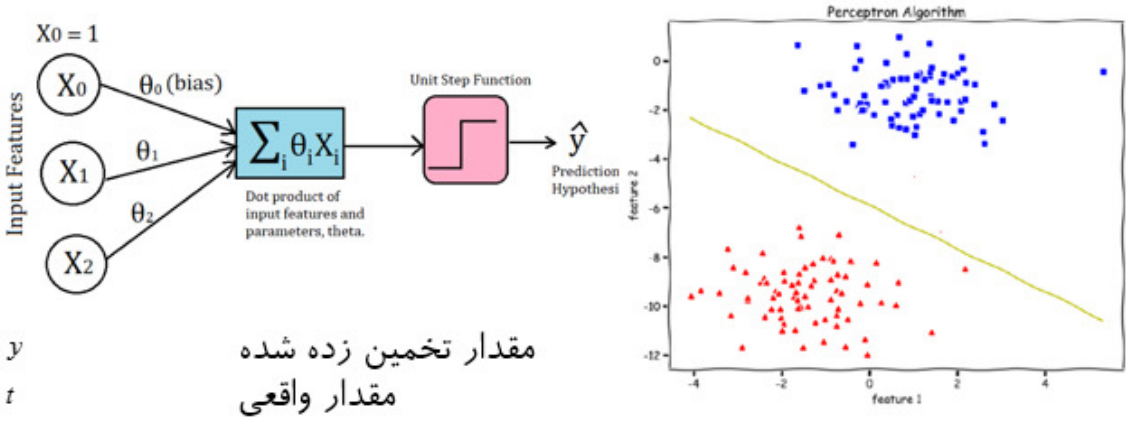
$$D_{t+1}(i) = \frac{D_t(i)}{Z_t} \cdot = \begin{cases} e^{-\alpha_t} & \text{if instance } i \text{ is correctly classified} \\ e^{\alpha_t} & \text{if instance } i \text{ is not correctly classified} \end{cases}$$

where $Z_t$ is a normalization factor (chosen so that $\sum_{i=1}^{m} D_{t+1} = 1$).
Output the final hypothesis: $H(x) = \mathrm{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right)$.

## نگاهی گذرا به پرسپترون

شبکهٔ عصبی پرسپترون تک لایه، یک یادگیرنده ضعیف است که می‌تواند در الگوریتم آدابوست مورد استفاده قرار گیرد. در شبکهٔ عصبی پرسپترون مجموع وزن‌دار ورودی‌های شبکه عصبی از یک تابع فعالیت گذر کرده و بدین ترتیب برچسبی به نمونهٔ موردنظر اختصاص می‌یابد و درصورتی‌که مقدار برچسب با برچسب واقعی متفاوت باشد تخمین صحیحی صورت نپذیرفته و بنابراین با استفاده از قانون دلتا، در یک فرایند تکراری، وزن‌ها به‌روزرسانی خواهند شد تا میزان خطا به حداقل برسد.



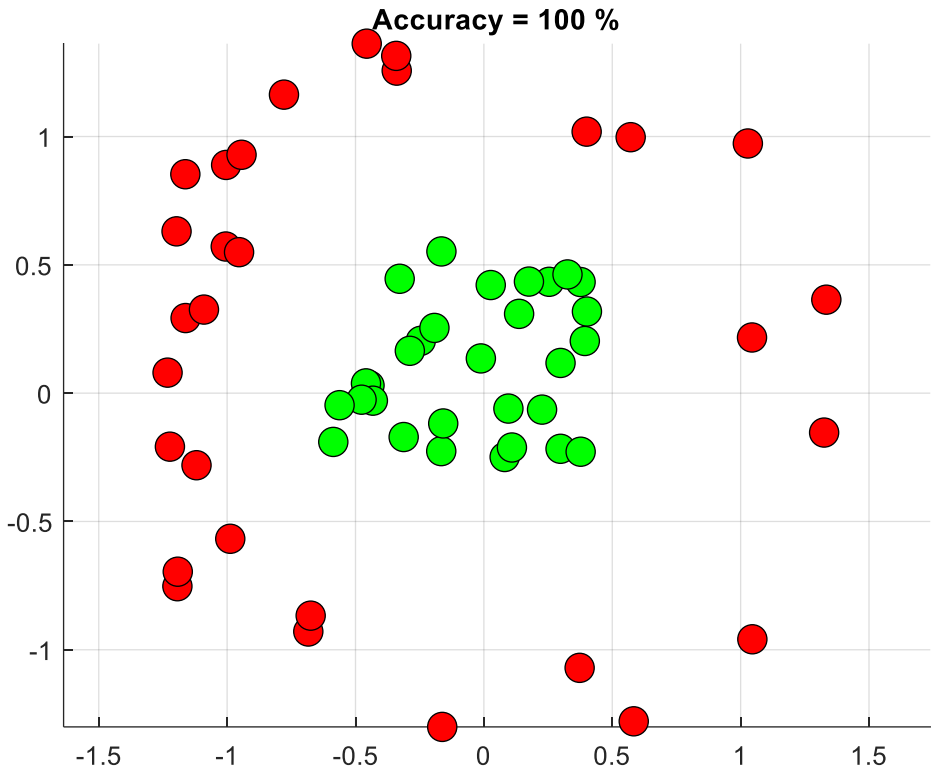




$y$ مقدار تخمین زده شده

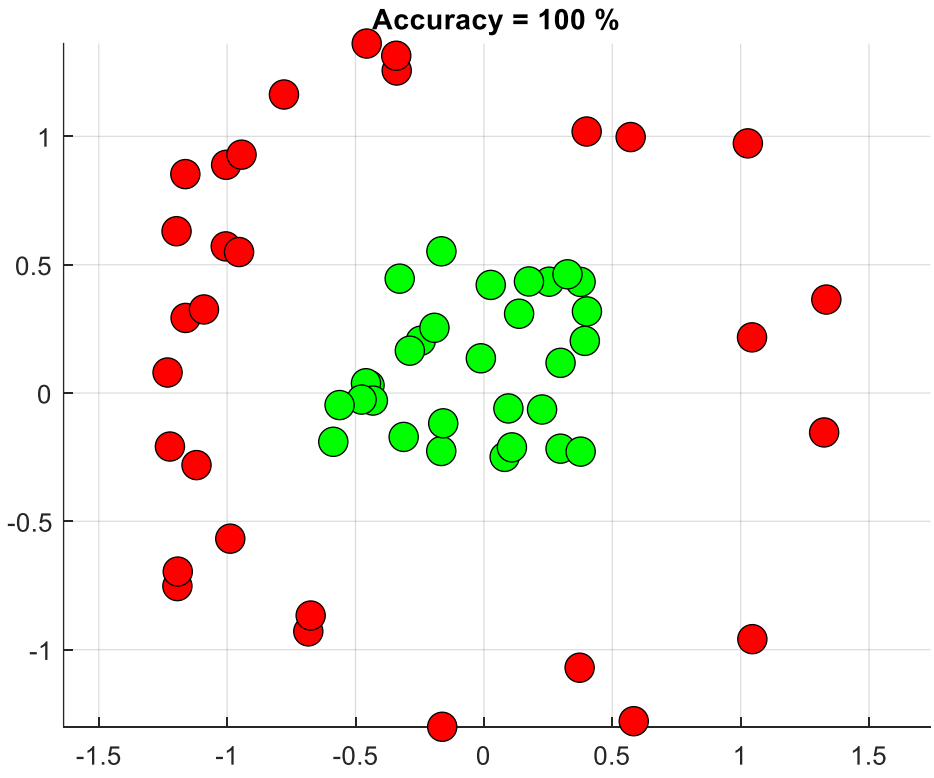
$t$ مقدار واقعی

$e = t - y$

$\theta_i \leftarrow \theta_i + \alpha . e . x_i$ به‌روزرسانی وزن‌ها با استفاده از قانون دلتا




دقت آموزش روی داده‌های آموزشی.




دقت تخمین روی داده‌های تست.

```
clc;
clear;
close all;
%% Generate nonlinear dataset
N = 100;
[classA, classB] = generateRings(N);

figure;
scatter(classA(1, :), classA(2, :), 120, 'g', 'filled', 'MarkerEdgeColor', 'k');
hold on;
scatter(classB(1, :), classB(2, :), 120, 'r', 'filled', 'MarkerEdgeColor', 'k');
axis equal; grid on;
title('Nonlinear Dataset');
%% Split data into train and test
ratio = 0.7;
nA = round(size(classA, 2)*ratio);

trainA = classA(:, 1:nA);
testA = classA(:, nA+1:end);

trainB = classB(:, 1:nA);
testB = classB(:, nA+1:end);

Xtrain = [trainA, trainB];
Ytrain = [ones(1, size(trainA, 2)), -ones(1, size(trainB, 2))];

Xtest = [testA, testB];
Ytest = [ones(1, size(testA, 2)), -ones(1, size(testB, 2))];
%% AdaBoost settings
T = 230;
weights = ones(1, size(Xtrain, 2)) / size(Xtrain, 2);
models = cell(1, T);
alpha  = zeros(1,T);
%% Boosting loop
for t = 1:T

    % Weighted sampling
    cumW = cumsum(weights);
    r = rand(1, size(Xtrain, 2));
    idx = arrayfun(@(x) find(cumW >= x, 1), r);

    Xb = Xtrain(:, idx);
    Yb = Ytrain(idx);

    % Train weak learner (SLP)
    models{t} = trainSLP(Xb, Yb);

    % Test on training set
    pred = testSLP(models{t}, Xtrain);
    err = (pred ~= Ytrain);

    eps_t = sum(weights.*err);
    eps_t = max(min(eps_t, 1-1e-10), 1e-10); % avoid division issues

    % Compute alpha
    alpha(t) = 0.5 * log((1 - eps_t)/eps_t);

    % Update weights
    weights = weights .* exp(alpha(t)*err);
```

```
    weights = weights / sum(weights);
end
%% Final prediction
scores = zeros(T, size(Xtest, 2));

for t = 1:T
    scores(t, :) = alpha(t) * testSLP(models{t}, Xtest);
end

final = sign(sum(scores, 1));
acc = mean(final == Ytest) * 100;
%% Plot results
figure;
scatter(Xtest(1, final == 1), Xtest(2, final == 1), 120, 'g', 'filled',
'MarkerEdgeColor', 'k');
hold on;
scatter(Xtest(1, final == -1), Xtest(2, final == -1), 120, 'r', 'filled',
'MarkerEdgeColor', 'k');
axis equal; grid on;
title(['Accuracy = ', num2str(acc), ' %']);

function [A, B] = generateRings(N)
% Generate two-ring nonlinear dataset
r1 = sqrt(0.4*rand(N, 1));
t1 = 2 * pi * rand(N, 1);
A = [r1 .* cos(t1), r1 .* sin(t1)]';

r2 = sqrt(rand(N, 1)+1.1);
t2 = 2 * pi * rand(N, 1);
B = [r2 .* cos(t2), r2 .* sin(t2)]';

A = A(:, randperm(N));
B = B(:, randperm(N));
End

function mdl = trainSLP(X, Y)
% Simple least-squares SLP training
X = [-ones(1, size(X, 2)); X];
R = X * X';
P = Y * X';
mdl.w = P / R;
end

function out = testSLP(mdl, X)
% Apply SLP model
X = [-ones(1, size(X, 2)); X];
out = sign(mdl.w*X);
end
```