به نام خداوند

تئوری یادگیری ماشین

دکتر سیدصالحی

جلسه سوم

دانشکده ریاضی و علوم کامپیوتر

اسفند ماه ۱۴۰۲

---

۱ مسئله $Classification$

مسئله $Classification$ یا طبقه‌بندی، یکی از مسائل اصلی در یادگیری ماشین و داده‌کاوی است که هدف آن تخصیص دادن هر نمونه یا داده به یکی از دسته‌ها یا کلاس‌های مشخص است بر اساس ویژگی‌ها یا خصوصیات آن نمونه. به زبان ساده‌تر، در یک مسئله طبقه‌بندی، ما می‌خواهیم پیش‌بینی کنیم که یک مورد خاص به کدام دسته تعلق دارد. به عنوان مثال، طبقه‌بندی ایمیل‌ها به «اسپم» یا «غیراسپم»، تشخیص اینکه یک تصویر حاوی گربه است یا سگ، یا تعیین اینکه یک تراکنش مالی مشکوک به تقلب است یا خیر، همگی از مثال‌هایی برای مسائل طبقه‌بندی هستند.

در طبقه‌بندی، مجموعه آموزشی یا $TrainingSet$ شامل داده‌هایی است که برای آموزش مدل یادگیری ماشین استفاده می‌شوند. این داده‌ها شامل نمونه‌هایی با ویژگی‌های مشخص هستند که هر کدام به یکی از کلاس‌های مورد نظر ما مربوط می‌شوند. هر نمونه در این مجموعه دارای یک برچسب یا کلاس معین است که مشخص می‌کند به کدام دسته تعلق دارد.

دو نوع اصلی مسائل طبقه‌بندی در یادگیری ماشین عبارت اند از: طبقه‌بندی دودویی (باینری) و طبقه‌بندی چندکلاسه.

- در حالت دو کلاسی (باینری)، خروجی‌ها به صورت $y \in \{0, 1\}$ نمایش داده می‌شوند. به این معنی که هر نمونه می‌تواند به یکی از دو کلاس موجود تعلق داشته باشد، معمولاً ۰ نمایانگر کلاس منفی و ۱ نمایانگر کلاس مثبت است. به عنوان مثال، در تشخیص ایمیل‌های اسپم، یک ایمیل می‌تواند اسپم (۱) یا غیراسپم (۰) باشد.

- در حالت چند کلاسی، خروجی‌ها اغلب به صورت بردارهایی از اعداد ۰ و ۱ نمایش داده می‌شوند که به آنها بردارهای یک‌داغ یا $one-hot\,vectors$ گفته می‌شود. برای مثال، $y = [0, 1, 0, 0, 0]$ نشان می‌دهد که نمونه به کلاس دوم تعلق دارد (کلاس‌ها از ۱ تا K شماره‌گذاری شده‌اند، و در این مثال $K \geq 5$). در این روش، هر عنصر بردار نمایانگر یک کلاس است و مقدار ۱ نشان‌دهنده تعلق نمونه به آن کلاس و مقادیر ۰ نشان‌دهنده عدم تعلق به سایر کلاس‌ها است.

تابع تمییزدهنده یا $Discriminant\,Function$، در $Classification$، یک تابع ریاضی است که برای تعیین کلاس یک نمونه داده شده بر اساس ویژگی‌های آن استفاده می‌شود. این تابع ورودی را دریافت می‌کند و مستقیما مقداری را تولید می‌کند که نشان‌دهنده کلاسی است که نمونه به آن تعلق دارد. در واقع، این تابع به ما کمک می‌کند تا بین دو یا چند کلاس تمایز قائل شویم.

به عنوان مثال، در طبقه‌بندی دو کلاسه، یک تابع تمییزدهنده ممکن است به این صورت باشد که اگر مقدار تابع برای یک نمونه بیشتر از یک آستانه خاص باشد، نمونه به کلاس ۱ تعلق دارد، و در غیر این صورت، به کلاس ۰ تعلق دارد. در موارد چند کلاسی، ممکن است برای هر کلاس یک تابع تمییزدهنده داشته باشیم و کلاسی که بیشترین مقدار تابع تمییزدهنده را دارد، به عنوان کلاس نمونه انتخاب می‌شود.

۲ $Linear\ classifiers$

- فرض می‌کنیم فضای ورودی به نواحی تصمیم تقسیم می‌شود که مرزهای آنها به نام مرزهای تصمیم می‌شناسیم.

- سطوح تصمیم به صورت توابع خطی از بردار ورودی x هستند.

- توسط هایپرپلین‌هایی با بعد $d-1$ درون فضای ورودی d بعدی تعریف می‌شوند.

به عبارت دیگر در مسئله $Classification$، فضایی که داده‌های ما در آن قرار دارند (فضای ورودی) را می‌توان به قسمت‌های مختلف (نواحی تصمیم) تقسیم کرد که هر کدام نشان‌دهنده یک کلاس خاص هستند. مرزهایی که این نواحی را از هم جدا می‌کنند، مرزهای تصمیم نامیده می‌شوند که می‌توانند خطی (یا به صورت سطوح در فضاهای با بعد بالاتر) باشند. این مرزها بر اساس ویژگی‌های داده‌ها تعریف می‌شوند و هدف از تعریف آنها این است که بتوان با دیدن ویژگی‌های یک نمونه جدید، تعیین کرد که این نمونه به کدام کلاس تعلق دارد.

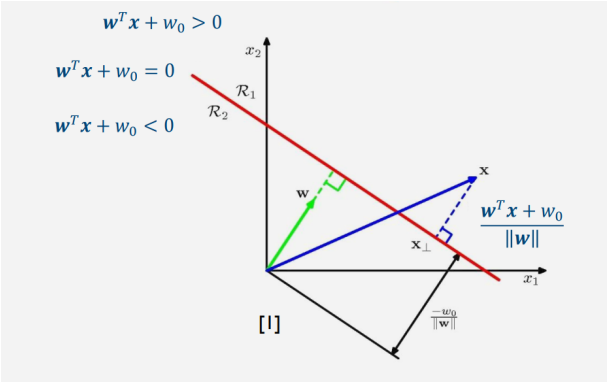
۱-۲ $Binary\ classification$

- $h(x;w) = w^T x + w_0 = w_0 + w_1 x_1 + ... + w_d x_d$ در دسته‌بندی باینری با با این تابع نمونه‌ها را از هم جدا می‌کنیم، که در آن x بردار ویژگی‌های ورودی و w وزن‌های مرتبط با این ویژگی‌ها است. w_0، که به عنوان $bias$ شناخته می‌شود، یک جمله ثابت است که به تنظیم مقدار تابع تمییزدهنده کمک می‌کند.

- $x = [x_1, x_2, ..., x_d]$ نشان‌دهنده بردار ویژگی‌های ورودی است.

- $w = [w_1, w_2, ..., w_d]$ نشان‌دهنده وزن‌های اختصاص داده شده به هر ویژگی است.

- تابع تمییزدهنده خطی به این صورت عمل می‌کند که اگر $w^T x + w_0 \geq 0$، آنگاه نمونه به کلاس C_1 تعلق دارد، در غیر این صورت به کلاس C_2.

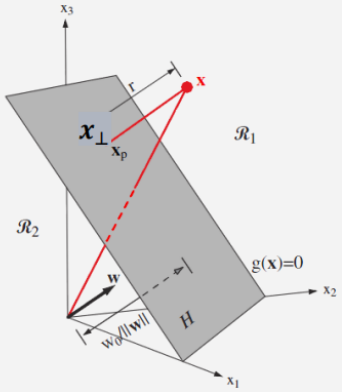


شکل ۱: مرز خطی جداکننده

$$x = x_\perp + r\frac{w}{\|w\|}$$

$$w^T x + w_0 = r\|w\| \Rightarrow r = \frac{w^T x + w_0}{\|w\|}$$

gives a signed measure of the perpendicular distance r of the point x from the decision surface



شکل ۲: مرز جداکننده در داده‌های سه بعدی

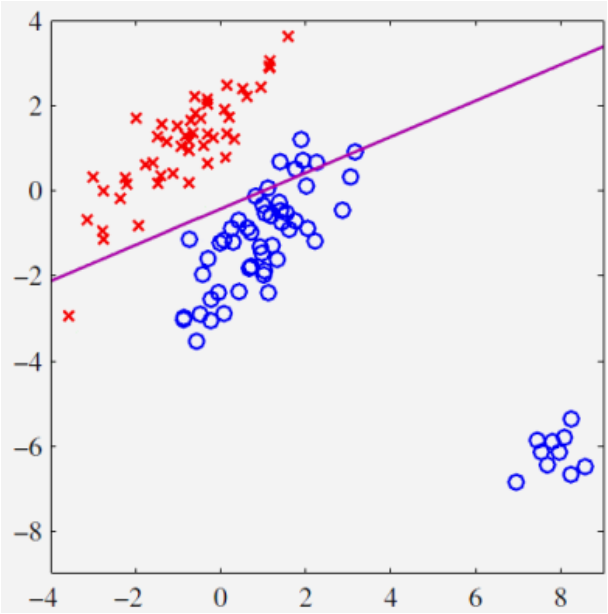
$$h(x; w, w_0) = sign(w^t x + w_0) = \begin{cases} +1 & if\, w^t x + w_0 > 0 \\ -1 & otherwise \end{cases}$$

- سطح تصمیم (یا مرز تصمیم): $w^T x + w_0 = 0$ معادله‌ای است که مرز بین دو کلاس را مشخص می‌کند. این سطح یا خط، نواحی تصمیم را در فضای ویژگی جدا می‌کند، به طوری که نمونه‌هایی که در یک طرف این مرز قرار دارند به یک کلاس و نمونه‌هایی که در طرف دیگر قرار دارند به کلاس دیگر تعلق دارند.

۲-۲ *Cost function*

برای طبقه‌بندی‌کننده‌های خطی، تابع هزینه به شکل یک مسئله بهینه‌سازی تعریف می‌شود:

- ابتدا باید معیاری برای اندازه‌گیری خطای پیش‌بینی انتخاب شود.



شکل ۳: دسته بند دوتایی

- بر اساس مجموعه آموزشی $D = \{(x^{(i)}, y^{(i)})\}_{i=1}^{n}$، تابع هزینه $J(w)$ تعریف می‌شود که در آن $x^{(i)}$ نمونه‌های ورودی و $y^{(i)}$ برچسب‌های مرتبط با این نمونه‌ها هستند.

- سپس، مسئله بهینه‌سازی حاصل برای یافتن بهترین پارامترها حل می‌شود: پارامترهای بهینه $\hat{w}$ از طریق مینیمم کردن تابع هزینه $J(w)$ بدست می‌آیند، یعنی $\hat{w} = \arg\min_w J(w)$.

- معیارها یا توابع هزینه برای طبقه‌بندی به عنوان شاخصی از کیفیت طبقه‌بندی‌کننده خطی عمل می‌کنند و در اینجا چندین تابع هزینه برای مسئله طبقه‌بندی بررسی خواهند شد.

۱-۲-۲ SSE

$$J(w) = \sum_{i=1}^{N} (w^T x^{(i)} - y^{(i)})^2$$

، تابع هزینه $Sum\ of\ Squared\ Errors\ (SSE)$ برای مسائل طبقه‌بندی مناسب نیست. علت این امر آن است که SSE حتی برای پیش‌بینی‌هایی که "بیش از حد صحیح" هستند و در واقع دیتا خوبی هستند، در طرف صحیح تصمیم (خط تصمیم) قرار دارند، جریمه‌هایی اعمال می‌کند. در واقع، این تابع هزینه فاصله پیش‌بینی‌های صحیح را از خط تصمیم نیز به عنوان خطا تلقی می‌کند، که این موضوع در مسائل طبقه‌بندی مطلوب نیست همانطور که در شکل ۳ این موضوع رو میبینیم که باعث مشکل شده.

که در این فرمول w بردار وزن‌ها، $x^{(i)}$ نمونه i-ام و $y^{(i)}$ برچسب واقعی نمونه i-ام است.

$Sign$ ۲-۲-۲

می‌توانیم در تابع هزینه ، از تابع علامت یا $Sign$ استفاده کنیم. در این حالت تابع هزینه $J(w)$ به صورت زیر تعریف خواهد شد:

$$J(w) = \sum_{i=1}^{N} (sign(w^T x^{(i)}) - y^{(i)})^2$$

مثال۱: فرض کنید h یک $classifier$ خطی تعریف‌شده باشد به صورت زیر:

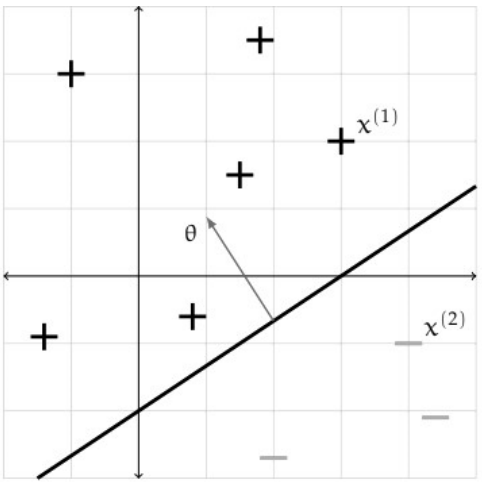
$$w = \begin{bmatrix} -1 \\ 1.5 \end{bmatrix}$$

$$w_0 = 3$$

نمودار زیر چندین نقطه را نشان می دهد که توسط h طبقه بندی شده اند. به طور خاص، ما با ۲ نقطه زیر کار داریم:

$$x^1 = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$$

$$x^2 = \begin{bmatrix} 4 \\ -1 \end{bmatrix}$$

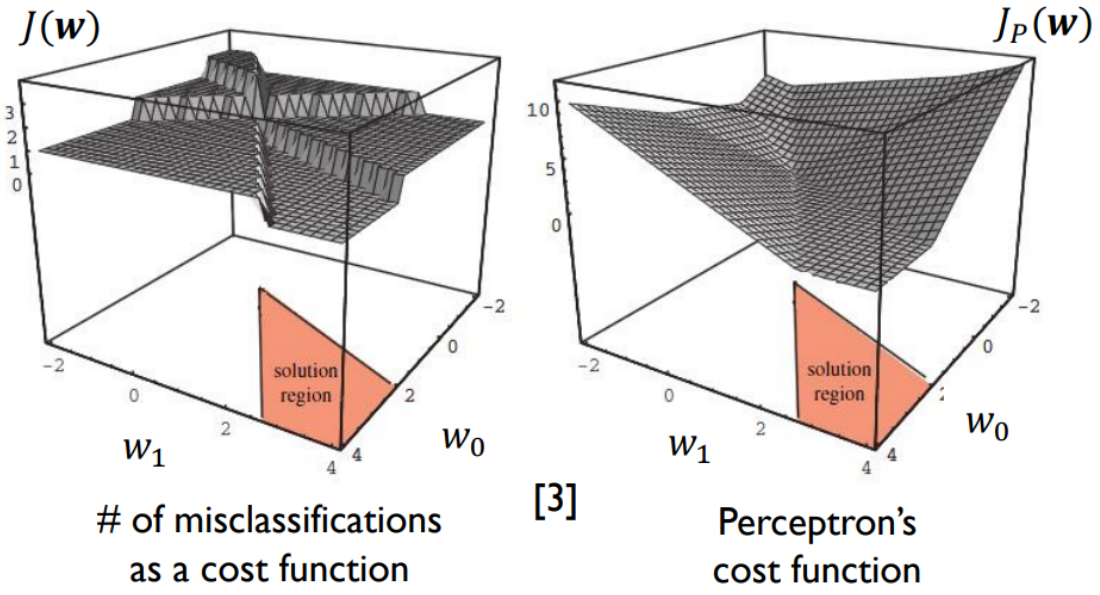


شکل ۴: شکل مثال ۱

خواهیم داشت که:

$$h(x^1; w, w_0) = sign\left(\begin{bmatrix}-1 & 1.5\end{bmatrix}\begin{bmatrix}3\\2\end{bmatrix} + 3\right) = sign(3) = +1$$

$$h(x^2; w, w_0) = sign\left(\begin{bmatrix}-1 & 1.5\end{bmatrix}\begin{bmatrix}4\\-1\end{bmatrix} + 3\right) = sign(-2.5) = -1$$



شکل ۵: مقایسه تابع هزینه پرسترون با تابع ایده‌آل

۳-۲-۲ *Preceptron*

مجدداً یک مجموعه داده آموزشی D_n با x هایی در R^dداریم. الگوریتم پرسپترون یک طبقه‌بندی‌کننده باینری $h(x; w, w_0)$ را با استفاده از یک الگوریتم آموزش می‌بینند تا w و w_0 را با استفاده از گام‌های $iterative$ پیدا کند.

معیار پرسپترون به صورت زیر تعریف میشود:

$$J_p(w) = -\sum_{i \in M} w^T x^{(i)} y^{(i)}$$

$M : subset\ of\ training\ data\ that\ are\ misclassified$

در واقع در پرسپترون میایم و در هر $iteration$ میایم یه جور گام بر میداریم که مجموع فاصله های $misclassified$ تا مرز تصمیم گیری هی کم شه.

تابع هزینه را به شکل زیر تعریف میکنیم:

$$J_p(w) = -\sum_{i \in M} w^T x^i y^i$$

دو نمودار سه‌بعدی شکل ۵ نیز نشان می‌دهند که تغییرات تابع هزینه را با تغییر وزن‌ها نشان می‌دهند. در نمودار سمت چپ، تعداد طبقه‌بندی‌های اشتباه به عنوان تابع هزینه در نظر گرفته شده است. در نمودار سمت راست، تابع هزینه پرسپترون نشان داده شده است.

در هر دو نمودار شکل ۵، منطقه‌ای که راه‌حل در آن قرار دارد (منطقه‌ای که در آن وزن‌ها می‌توانند داده‌ها را به درستی طبقه‌بندی کنند) با رنگ نارنجی مشخص شده است. این نمودارها به ما نشان می‌دهند که چگونه تابع هزینه می‌تواند بر اساس وزن‌های مختلف (w_0 و w_1) تغییر کند و به ما کمک کند تا بهترین وزن‌ها را برای مدل پرسپترون پیدا کنیم.

PERCEPTRON$(\tau, \mathcal{D}_n)$
1 $\mathrm{w} = \begin{bmatrix} 0 & 0 & \cdots & 0 \end{bmatrix}^T$
2 $\mathbf{w}_0 = 0$
3 **for** $t = 1$ **to** τ
4 **for** $i = 1$ **to** n
5 **if** $y^{(i)} \left(\mathbf{w}^T x^{(i)} + \mathbf{w}_0\right) \leqslant 0$
6 $\mathbf{w} = \mathbf{w} + y^{(i)} x^{(i)}$
7 $\mathbf{w}_0 = \mathbf{w}_0 + y^{(i)}$
8 **return** $\mathbf{w}, \mathbf{w}_0$

ایرادی که $perceptron$ دارد این است که ما وقتی به یک ریجن میرسیم که دیگه داده $misclassified$ وجود ندارد همه خط های ممکن برای $separate$ کردن دیتا خوب است و تفاوتی قاعل نیست در حالی که برای $generalization$ ما ممکن است نیاز به یکی از خط های ممکن از میان تمام خط های جواب حال

حاضر داشته باشیم که بهینه ترین است.

برای حل مساله از $gradient\ descent$ استفاده میکنیم. به صورت:

$$w^{t+1} = w^t - \mu \nabla_w J_p(w^t)$$
$$\nabla_w J_p(w) = -\sum_{i \in M} x^i y^i$$

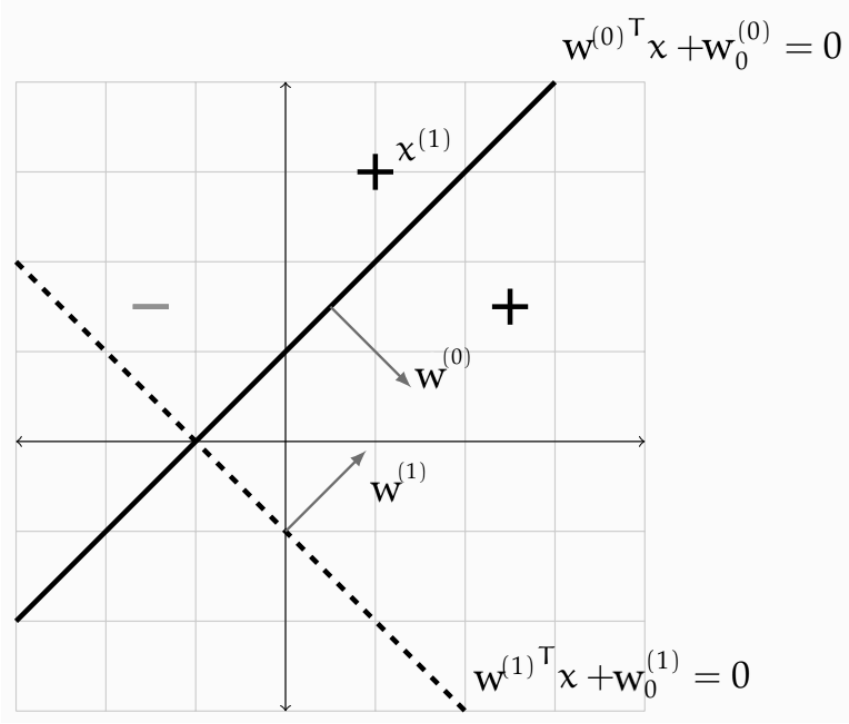
نکته ای که هست اینه که میشه اثبات کرد که در تعداد گام محدود $convergence$ رخ میدهد.

مثال۲: فرض کنید h یک $classifier$ خطی تعریف‌شده باشد به صورت زیر:

$$w^0 = \begin{vmatrix} -1 \\ 1 \end{vmatrix}$$

$$w_0^0 = 1$$

نمودار زیر چندین نقطه طبقه بندی شده توسط h را نشان می دهد.با این حال، در این مورد، h نقطه x^1 را دارد اشتباه طبقه بندی می کند که به آن لیبل ۱ را داده است.



شکل ۶: شکل مثال ۲

$$y^{(1)}(w^T x^{(1)} + w_0) = \begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 3 \end{bmatrix} + 1 = -1 < 0$$

با اجرای یک $iteration$ الگوریتم پرسپترون، ما خواهیم داشت که:

$$w^{(1)} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}, w_0^{(1)} = 2$$

طبقه‌بندی‌کننده جدید (که با خط چین نشان داده می‌شود) اکنون آن نقطه را به درستی طبقه‌بندی می‌کند، اما اکنون در نقطه با برچسب منفی اشتباه می‌کند.

۳-۲ $Pocket\ algorithm$

در $feature\ space$ کنونی اگر داده های من خطی جدایی پذیر نباشن $Pocket\ algorithm$ میگه که:

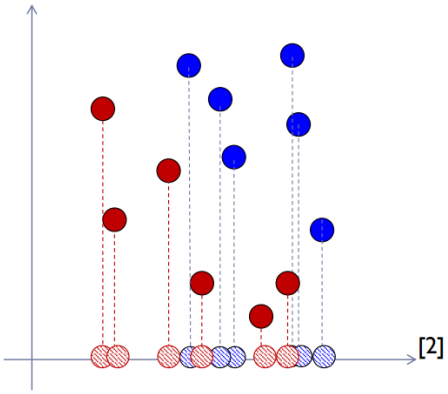
Initialize $\boldsymbol{w}$
for $t = 1, \ldots, T$
  $i \leftarrow t \bmod N$
  if $\boldsymbol{x}^{(i)}$ is misclassified then
    $\boldsymbol{w}^{new} = \boldsymbol{w} + \boldsymbol{x}^{(i)} y^{(i)}$
    if $E_{train}(\boldsymbol{w}^{new}) < E_{train}(\boldsymbol{w})$ then
      $\boldsymbol{w} = \boldsymbol{w}^{new}$
end

در $Pocket\ algorithm$ میایم یه تعداد $iteration$ محدود میگیریم و هربار یه داده $misclassified$ رو بگیر و w رو آپدیت کن و بهترین را نگه دار. به طور کلی ایده $Pocket\ algorithm$ این است که بهترین w را که تا به حال با آن روبرو شده است را نگه می دارد. صرفا یک $heuristic$ هست که میگه که اگر خطی جدایی پذیر نبودن بهترین خط کدوم هست.

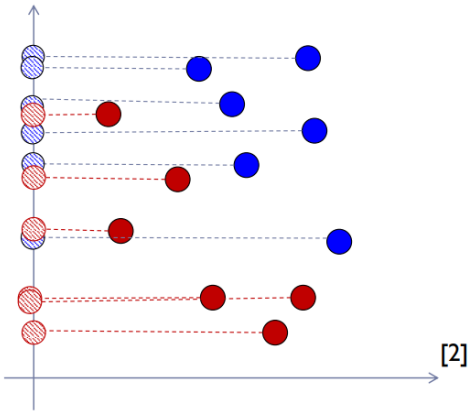
۴-۲ $Projection\ for\ Classification$

حال میخواهیم یه جهتی در فضا پیدا کنیم که در صورت map کردم نقاط به روی آن نقاط از هم خوب جدا بشوند.

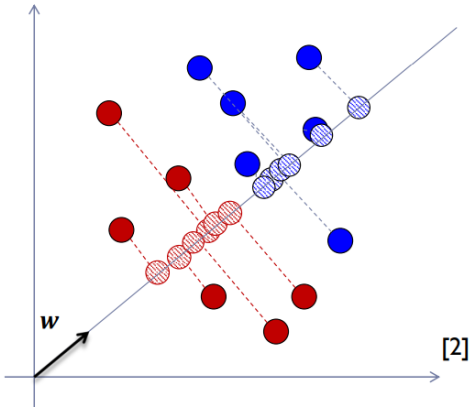
در اشکال ۷ و ۸ این جهات مناسب نیستند زیرا نقاط از هم به خوبی جدا نشده اند. ولی جهت انتخابی در شکل ۹ مناسب است و نقاط به خوبی از هم جدا شده اند.



شکل ۷: استفاده از یکی از محور های برای $mapping$



شکل ۸: استفاده از یکی از محور های برای $mapping$



شکل ۹: جهتی مناسب برای $mapping$

ما وقتی میایم و یکی از محور ها را به عنوان جهتمون برای $mapping$ انتخاب میکنیم انگار فقط یکی از $feature$ های نقاط از میان n تا را داریم برای جداسازی استفاده میکنیم در صورتی که میتوان از یک ترکیب خطی از این $feature$ ها را استفاده کرد و با یک تابع هزینه بدست آورد بهترین جهت کدام است.

۲-۴-۱ الگوریتم LDA

یک جهت w داریم که میخواهیم سمپل x را بر روی آن تصویر کنیم که تصویرش میشود $w^T x$.که خب هدف یافتن بهترین جهت w است که امیدواریم بتوانیم طبقه بندی دقیق را انجام دهیم

نوتیشن رو هم به این صورت قرار میدهیم که:

- $J(w)$: این تابع هدف را نشان می دهد که باید $maximize$ شود.
- w : این یک بردار وزن است که برای نمایش نقاط داده روی یک خط استفاده می شود.
- $_1$و $_2$: اینها نشان دهنده میانگین بردارهای دو کلاس هستند که وقتی $mapping$ بر روی w رخ داد یک پریم میگذاریم و میانگین map شده را به صورت پریم دار نمایش میدهیم.

با این اوصاف $measurement$ اینکه چقدر جهت منتخب برای جداسازی مناسب هست رو به صورت زیر تعریف میکنیم:

$$\max_{\boldsymbol{w}} J(\boldsymbol{w}) = (\mu_1' - \mu_2')^2$$
$$\text{s.t.} \|\boldsymbol{w}\| = 1$$

$$\mu_1' = \boldsymbol{w}^T \boldsymbol{\mu}_1 \qquad \boldsymbol{\mu}_1 = \frac{\sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_1} \boldsymbol{x}^{(i)}}{N_1}$$

$$\mu_2' = \boldsymbol{w}^T \boldsymbol{\mu}_2 \qquad \boldsymbol{\mu}_2 = \frac{\sum_{\boldsymbol{x}^{(i)} \in \mathcal{C}_2} \boldsymbol{x}^{(i)}}{N_2}$$