مرکز آموزش‌های الکترونیکی دانشگاه شهید بهشتی

شبکه‌های عصبی مصنوعی

Artificial Neural Networks

Ali Katanforoush*

** Department of Data and Computer Sciences, Shahid Beheshti University*

4

Perceptron Learning Rule

یادآوری و مقدمه

- به نظر می‌رسد برای بسیاری از مسائل، طراحی یک شبکه عصبی مناسب، بتواند مدل محاسباتی مورد نیاز برای حل آن را فراهم کند.
- طراحی یک شبکه عصبی در دو سطح تحقق می‌یابد:
  - تعریف ساختار یا معماری شبکه عصبی
  - تعریف مقادیر وزن پیوندها
- بدیهی است انتخاب معماری و ساختار مناسب برای یک شبکه عصبی مصنوعی به طور مستقیم بر روی قابلیت‌های محاسبه مدل مؤثر است.
- به همان اندازه مهم، انتخاب مقادیر مناسب وزن برای پیوندهای بین نورون‌ها است.

آیا قواعد مشخص و تعریف شده‌ای برای کشف مقادیر مناسب وزن وجود دارند؟

آیا همانند شبکه‌های عصبی زیستی، می‌توان مقادیر وزن را با مشاهده نمونه‌ها به شبکه آموخت؟

انواع یادگیری

- یادگیری با ناظر Supervised learning
  - برای یادگیری، نمونه‌هایی از زوج‌های ورودی/خروجی مطلوب در دسترس است:

$$\{\boldsymbol{p}_1, \boldsymbol{t}_1\} \ , \ \{\boldsymbol{p}_2, \boldsymbol{t}_2\} \ , \ \ldots \{\boldsymbol{p}_Q, \boldsymbol{t}_Q\}$$

  - یادگیری با هدف کمینه‌سازی خطای پیشگویی انجام می‌شود.
- یادگیری بدون ناظر Unsupervised learning
  - برای یادگیری، نمونه‌هایی از ورودی تنها در دسترس است.
  - یادگیری با هدف خوشه‌بندی نمونه‌های «مشابه» در تعدادی گروه انجام می‌شود.
- یادگیری تقویتی Reinforcement learning
  - یادگیری از طریق تجربه کردن – تعامل با محیط و دریافت پاداش/جریمه از محیط – صورت می‌گیرد.

Learning Rule

قاعده یادگیری

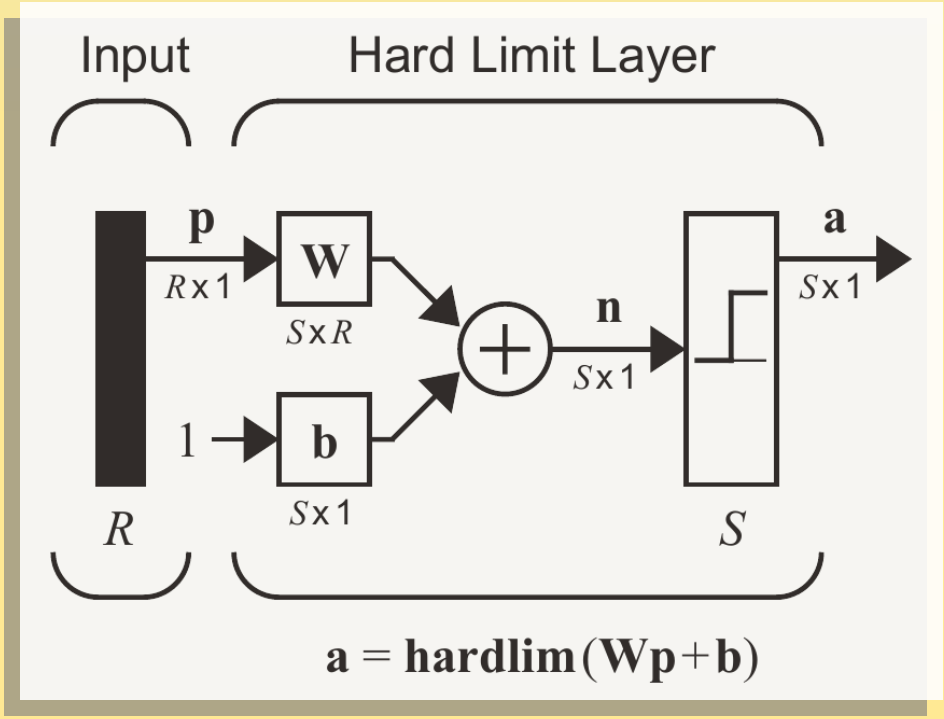
منظور از یک قاعده یادگیری، به بیان ساده، قاعده‌ای برای
تغییر مقادیر وزن پیوندهای شبکه عصبی مصنوعی
به ازای مشاهده یک نمونه یا دسته‌ای از نمونه‌ها یا دریافت پاداش یا جریمه از محیط است:

$$\boldsymbol{W}^{(new)} = \boldsymbol{W}^{(old)} \oplus \boldsymbol{H}(sample)$$

برخی قواعد یادگیری

- قاعده یادگیری پرسپترون
- قاعده یادگیری هب
- قاعده یادگیری کوهونن

معماری پرسپترون (تک لایه)

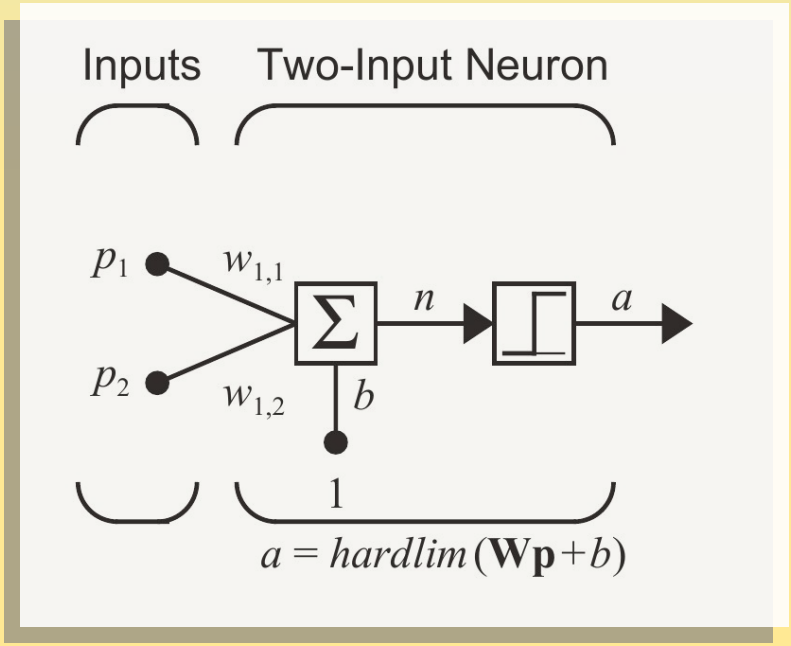


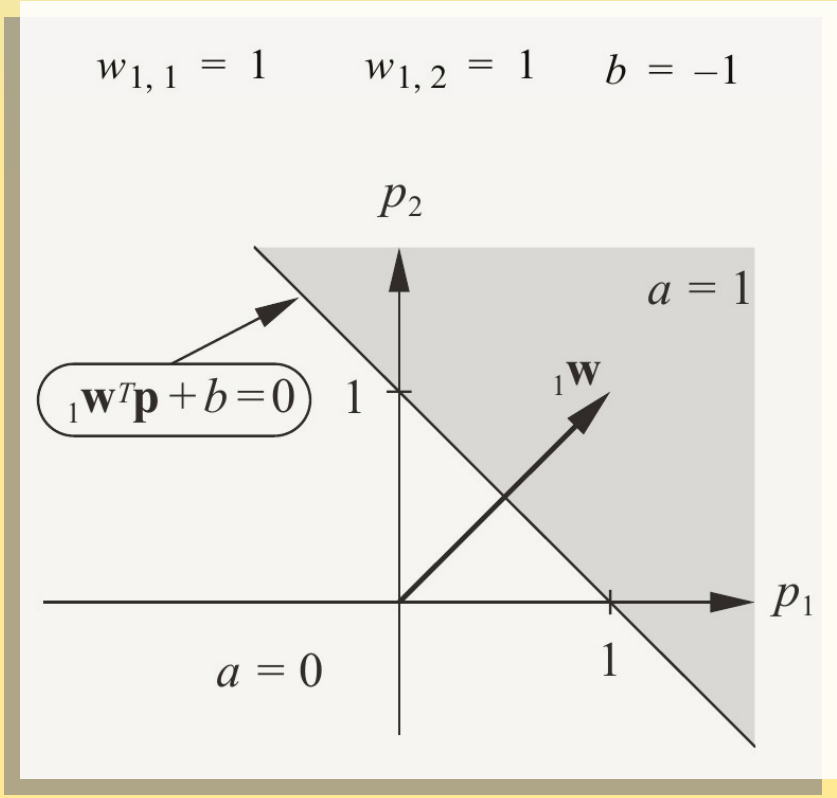
$$W = \begin{bmatrix} w_{11} & w_{12} & \cdots & w_{1R} \\ w_{21} & w_{22} & \cdots & w_{2R} \\ \vdots & \vdots & & \vdots \\ w_{S1} & w_{S2} & \cdots & w_{SR} \end{bmatrix}$$

$${}_i\boldsymbol{w} = \begin{bmatrix} w_{i1} \\ \vdots \\ w_{iR} \end{bmatrix} \qquad \boldsymbol{W} = \begin{bmatrix} {}_1\boldsymbol{w}^T \\ \vdots \\ {}_S\boldsymbol{w}^T \end{bmatrix}$$

$$a_i = hardlim(n_i) = hardlim({}_i\boldsymbol{w}^T \boldsymbol{p} + b_i)$$

پرسپترون تک نورونی

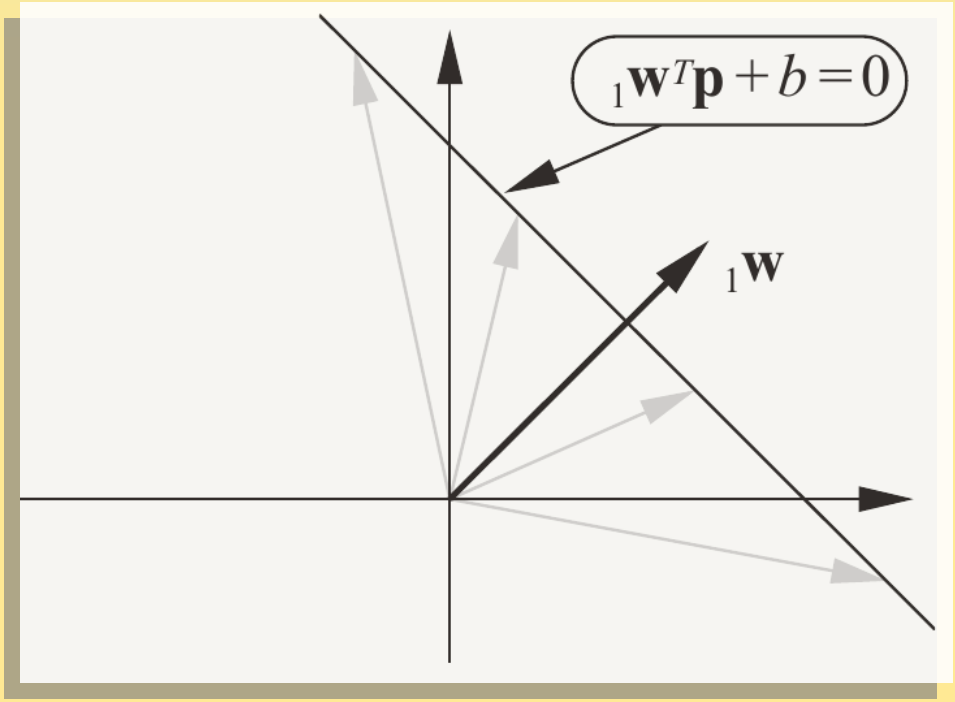




$$a = hardlim({}_1\boldsymbol{w}^T\boldsymbol{p} + b) = hardlim(w_{11}p_1 + w_{12}p_2 + b)$$

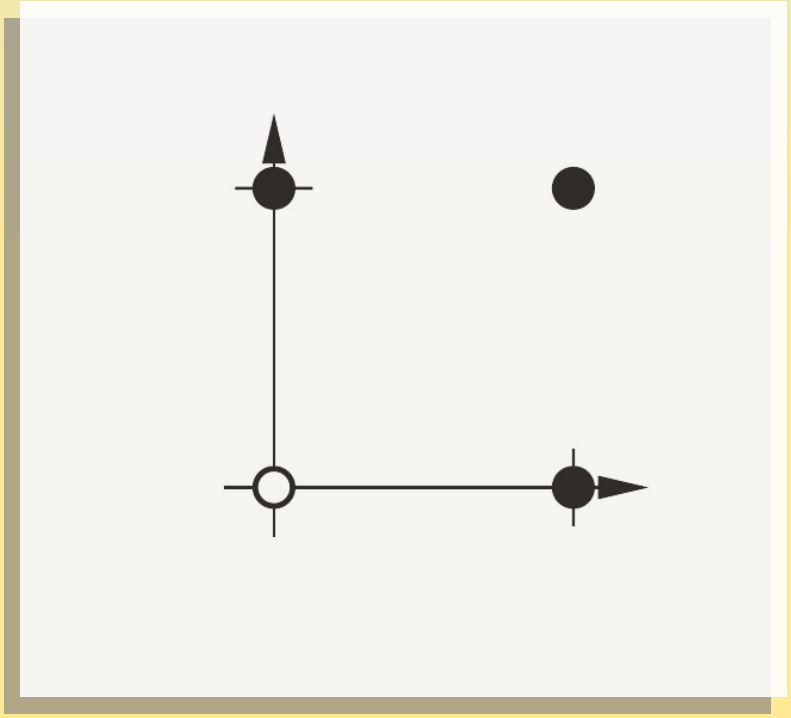
Decision Boundary

مرز تصمیم‌گیری



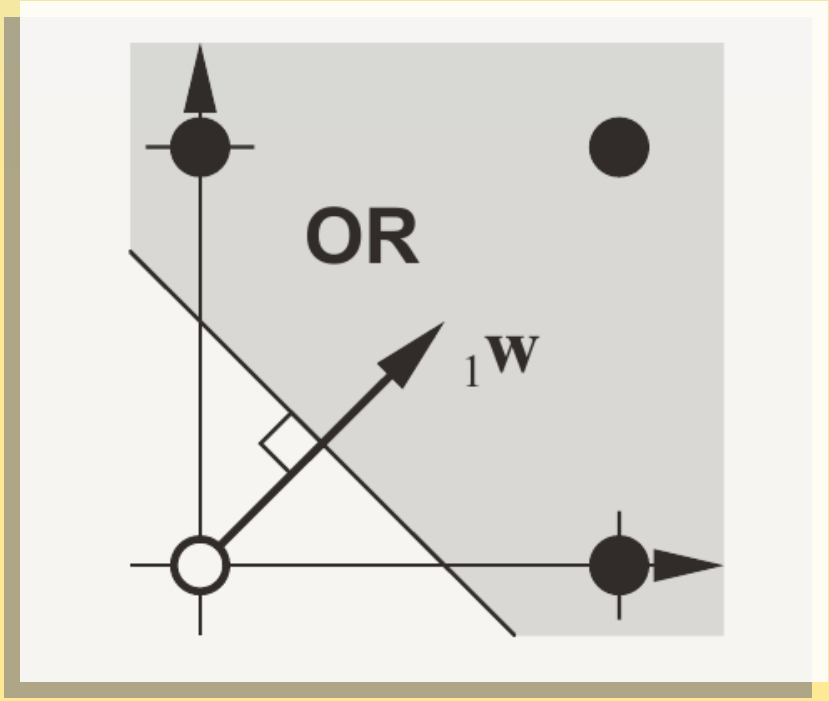
- رویهٔ تصمیم‌گیری یا به بیان کلی‌تر مرز تصمیم‌گیری در پرسپترون تک لایه، مکان هندسی نقاطی است که ضرب داخلی آن‌ها با بردار وزن، برابر با ثابت ($-b$) است.
- بنابراین بردار وزن بر ابرصفحه‌ای که مرز تصمیم‌گیری پرسپترون است عمود است.

مثال. عملگر OR



$$\left\{\mathbf{p}_1 = \begin{bmatrix}0\\0\end{bmatrix}, t_1 = 0\right\} \quad \left\{\mathbf{p}_2 = \begin{bmatrix}0\\1\end{bmatrix}, t_2 = 1\right\} \quad \left\{\mathbf{p}_3 = \begin{bmatrix}1\\0\end{bmatrix}, t_3 = 1\right\} \quad \left\{\mathbf{p}_4 = \begin{bmatrix}1\\1\end{bmatrix}, t_4 = 1\right\}$$

بردار وزن شبکهٔ OR



- بردار وزن باید بر مرز تصمیم‌گیری عمود باشد. مثلاً:

$$_1\boldsymbol{w} = \begin{bmatrix} 0.5 \\ 0.5 \end{bmatrix}$$

- حال برای تعیین مقدار بایاس (b)، کافی است رابطهٔ پرسپترون را برای یک نقطهٔ روی مرز تصمیم‌گیری، در معادله صدق دهیم:

$$_1\boldsymbol{w}^T\boldsymbol{p} + b = \begin{bmatrix} 0.5 & 0.5 \end{bmatrix}\begin{bmatrix} 0 \\ 0.5 \end{bmatrix} + b = 0.25 + b = 0 \;\Rightarrow\; b = -0.25$$

پرسپترون چند نورونی

- با هر نورون، یک مرز تصمیم‌گیری متفاوت می‌تواند تعریف شود.

$$_i\boldsymbol{w}^T\,\boldsymbol{p} + b_i = 0$$

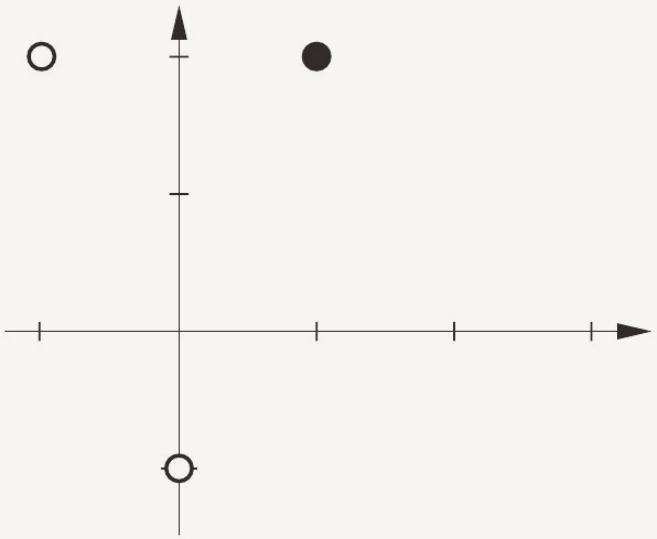
- یک نورون می‌تواند بردارهای ورودی را به ۲ کلاس دسته‌بندی کند.
- بنابراین، با S نورون می‌توان بردارهای ورودی را در 2^S کلاس دسته‌بندی کرد.

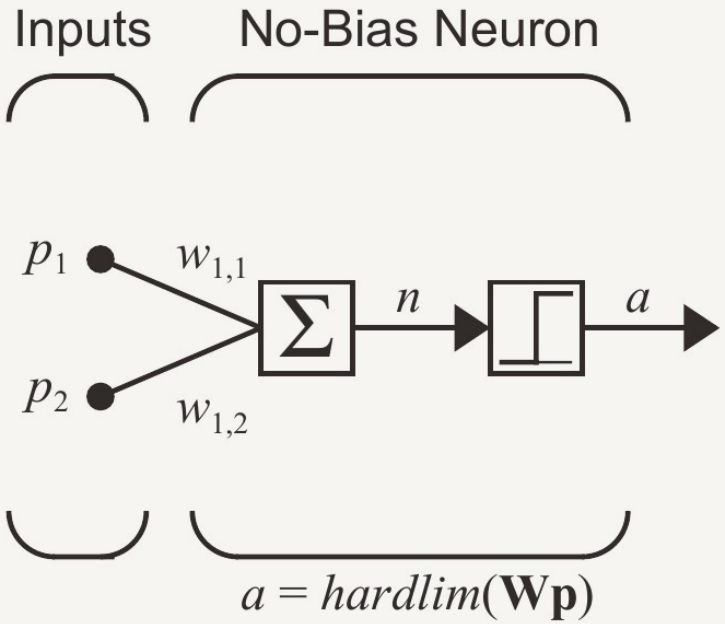
بررسی‌هایی برای بدست آوردن یک قاعده یادگیری تجربی

مثال آزمایشی

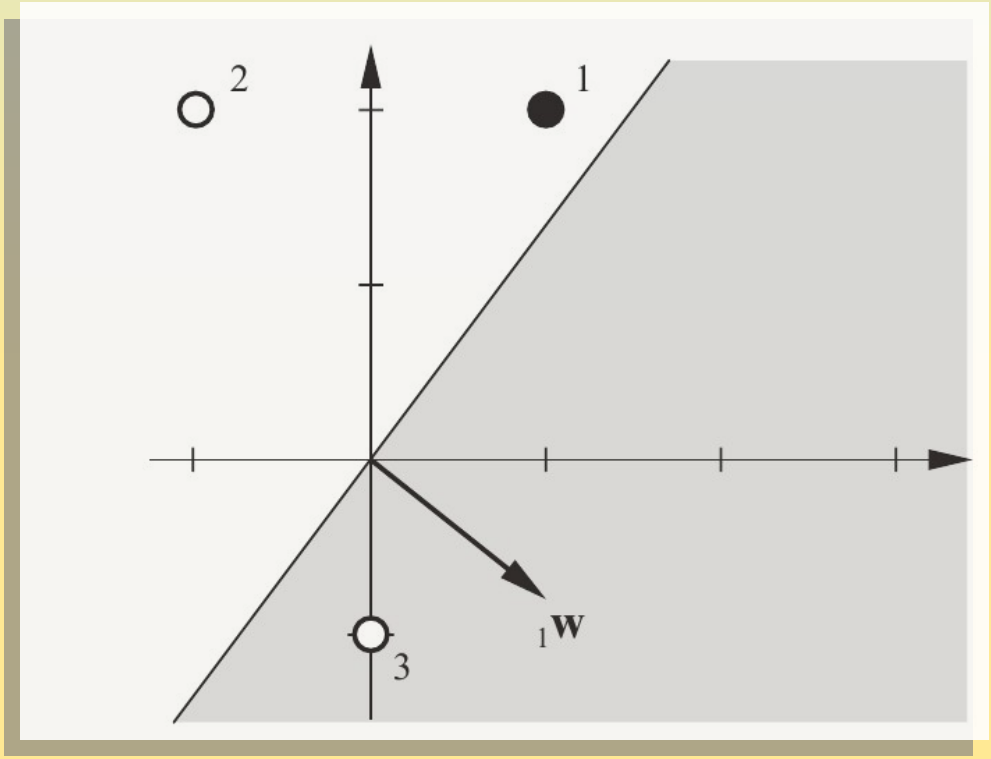
$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

$$\left\{\mathbf{p}_1 = \begin{bmatrix}1\\2\end{bmatrix}, t_1 = 1\right\} \quad \left\{\mathbf{p}_2 = \begin{bmatrix}-1\\2\end{bmatrix}, t_2 = 0\right\} \quad \left\{\mathbf{p}_3 = \begin{bmatrix}0\\-1\end{bmatrix}, t_3 = 0\right\}$$





گام آغاز



- با یک مقداردهی تصادفی برای بردار وزن آغاز می‌کنیم:

$$_1\boldsymbol{w} = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix}$$

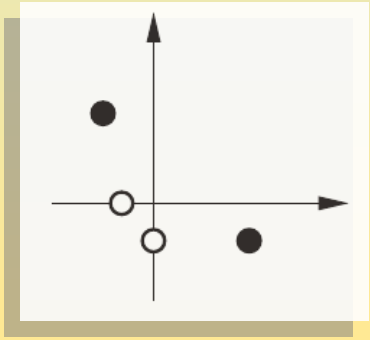
- نمونه $\boldsymbol{p}_1$ را به شبکه می‌دهیم:

$$a = hardlim(_1\mathbf{w}^T\mathbf{p}_1) = hardlim\left(\begin{bmatrix} 1.0 & -0.8 \end{bmatrix}\begin{bmatrix} 1 \\ 2 \end{bmatrix}\right)$$

$$a = hardlim(-0.6) = 0$$

دسته‌بندی ناصحیح

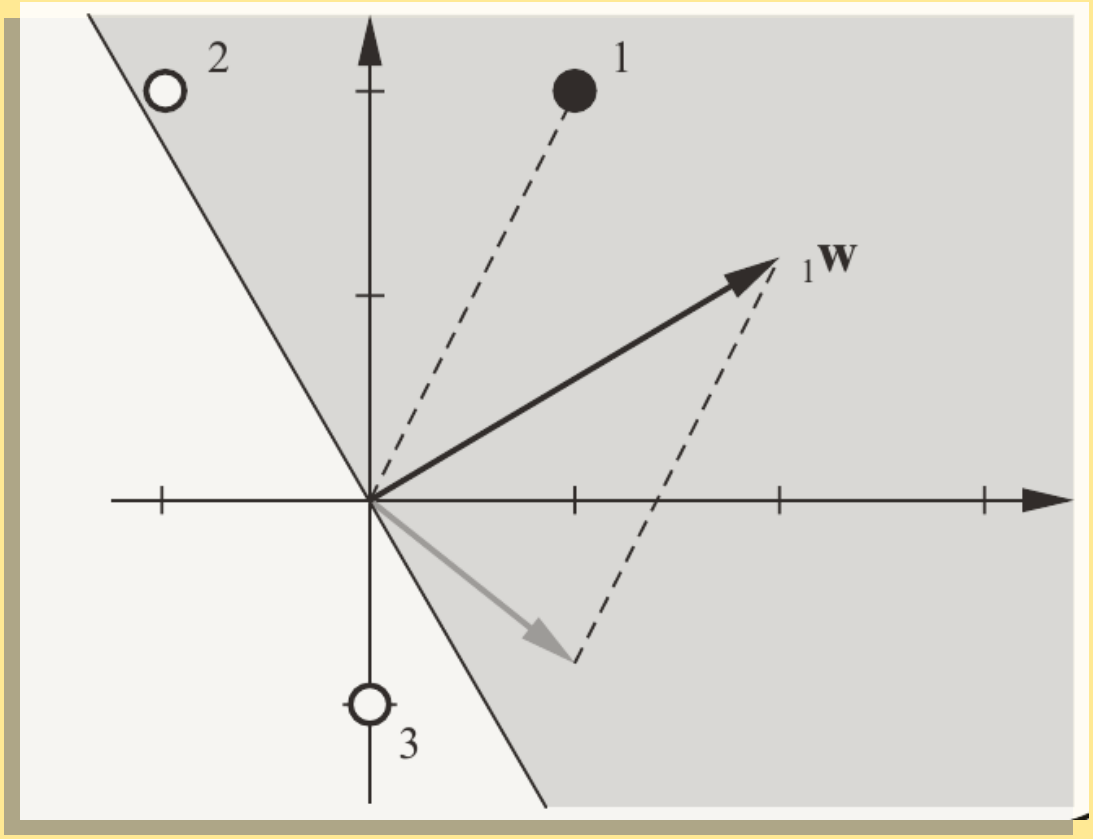
یک ایده تجربی برای تغییر مقادیر وزن



- وقتی شبکه بر روی نمونهٔ p_1 تشخیص نادرست می‌دهد:
  - ایده اول). مقدار وزن را با مقدار همان نمونه جایگزین کنید:

$$_1w^{(new)} = p_1$$

  - این روش ناپایدار است



  - ایده دوم). بردار نمونه را به بردار وزن اضافه کنید:

$$_1w^{(new)} = {}_1w^{(old)} + p_1$$

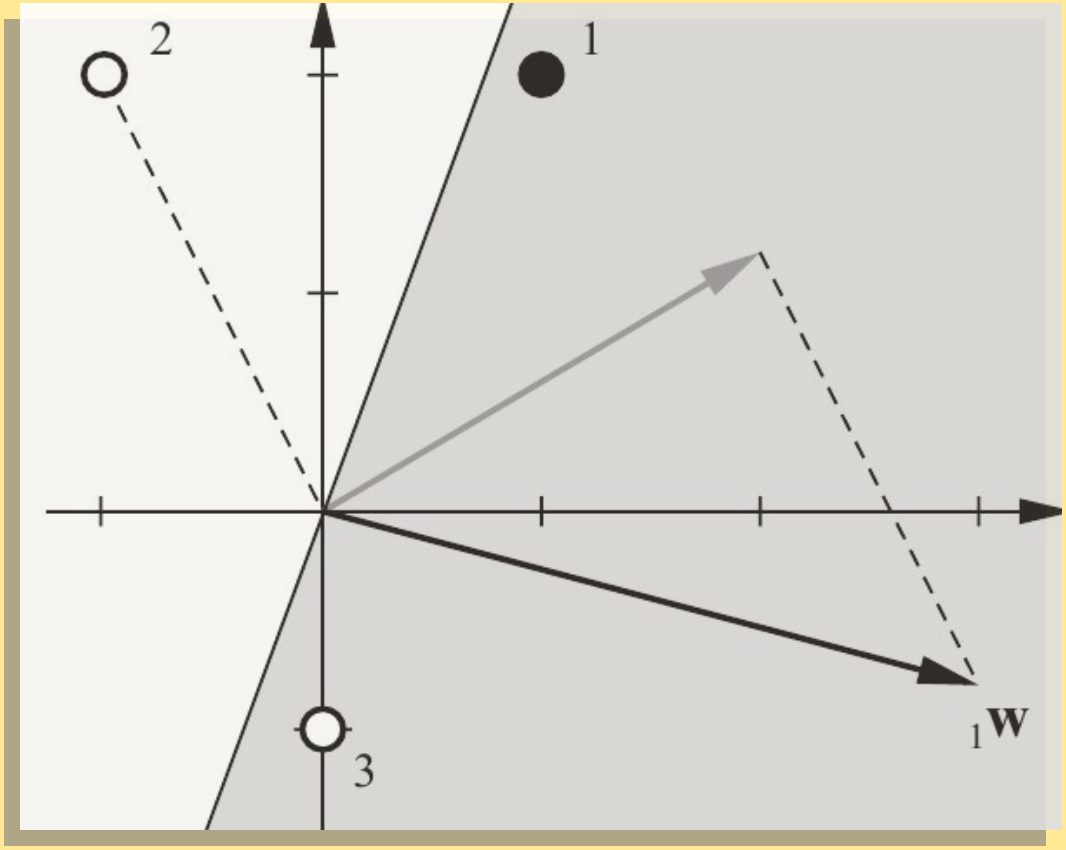
$$_1w^{(new)} = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix}$$

یک ایده تجربی برای تغییر مقادیر وزن (تکرار با بردار ورودی بعدی)

- حال با بردار وزن جدید، بردار ورودی دیگری را به شبکه می‌دهیم، مثلاً $\boldsymbol{p}_2$:

$$a = hardlim({}_1\boldsymbol{w}^T \boldsymbol{p_2}) = hardlim\left(\begin{bmatrix}2.0 & 1.2\end{bmatrix}\begin{bmatrix}-1\\2\end{bmatrix}\right) = hardlim(0.4) = 1$$

دسته‌بندی ناصحیح



- این بار، بردار ورودی می‌بایست در کلاس a=0 پیشگویی می‌شد اما شبکه اشتباه تشخیص داد.
- برای بهنگام‌سازی وزن، این بار، بردار ورودی را از بردار وزن کم می‌کنیم:

$${}_1\boldsymbol{w}^{(new)} = {}_1\boldsymbol{w}^{(old)} - \boldsymbol{p}_2$$

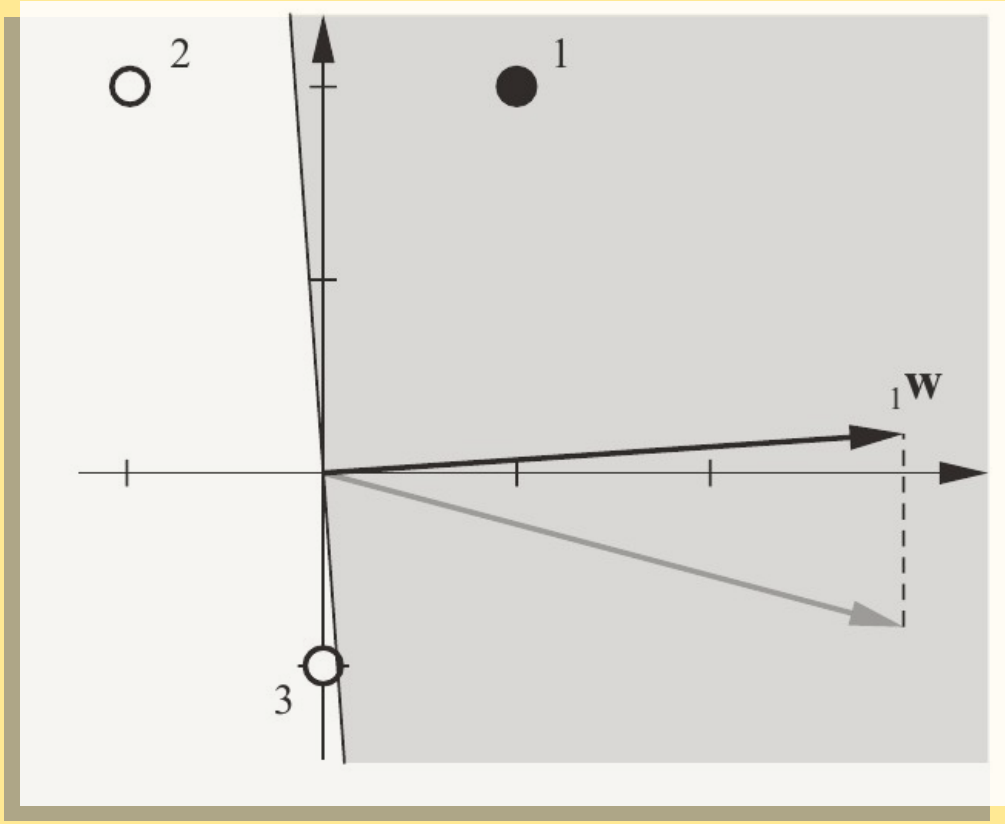
$${}_1\boldsymbol{w}^{(new)} = \begin{bmatrix}2.0\\1.2\end{bmatrix} - \begin{bmatrix}-1\\2\end{bmatrix} = \begin{bmatrix}3.0\\-0.8\end{bmatrix}$$

یک ایده تجربی برای تغییر مقادیر وزن (تکرار بعدی ...)

- خروجی شبکهٔ جدید با سومین بردار ورودی، $\boldsymbol{p}_3$ می‌شود:

$$a = hardlim({}_1\boldsymbol{w}^T \boldsymbol{p_3}) = hardlim\left(\begin{bmatrix}3.0 & -0.8\end{bmatrix}\begin{bmatrix}0\\-1\end{bmatrix}\right) = hardlim(0.8) = 1$$

دسته‌بندی ناصحیح



- با همان قاعدهٔ قبلی، وزن‌ها را تغییر می‌دهیم:

$${}_1\boldsymbol{w}^{(new)} = {}_1\boldsymbol{w}^{(old)} - \boldsymbol{p}_3$$

$${}_1\boldsymbol{w}^{(new)} = \begin{bmatrix}3.0\\-0.8\end{bmatrix} - \begin{bmatrix}0\\-1\end{bmatrix} = \begin{bmatrix}3.0\\0.2\end{bmatrix}$$

- این شبکه، دیگر تمام نمونه‌ها را به درستی دسته‌بندی می‌کند.

قاعده یادگیری کلی برای پرسپترون (تک نورونی)

$$\text{If } t = 1 \text{ and } a = 0, \text{ then } {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$$

$$\text{If } t = 0 \text{ and } a = 1, \text{ then } {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$$

$$\text{If } t = a, \text{ then } {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$$

قرارداد کنید: $e = t - a$

$$ {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + e\mathbf{p} = {}_1\mathbf{w}^{old} + (t-a)\mathbf{p}$$

$$b^{new} = b^{old} + e$$

پارامتر بایاس، مثل وزنی برای ورودیی متحد با 1 است.

قاعده یادگیری پرسپترون به فرم ماتریسی (پرسپترون چند نورونی)

بهنگام‌سازی سطر i اُم

$$_i\mathbf{w}^{new} = {}_i\mathbf{w}^{old} + e_i\mathbf{p}$$

$$b_i^{new} = b_i^{old} + e_i$$

بهنگام‌سازی تمام ماتریس وزن

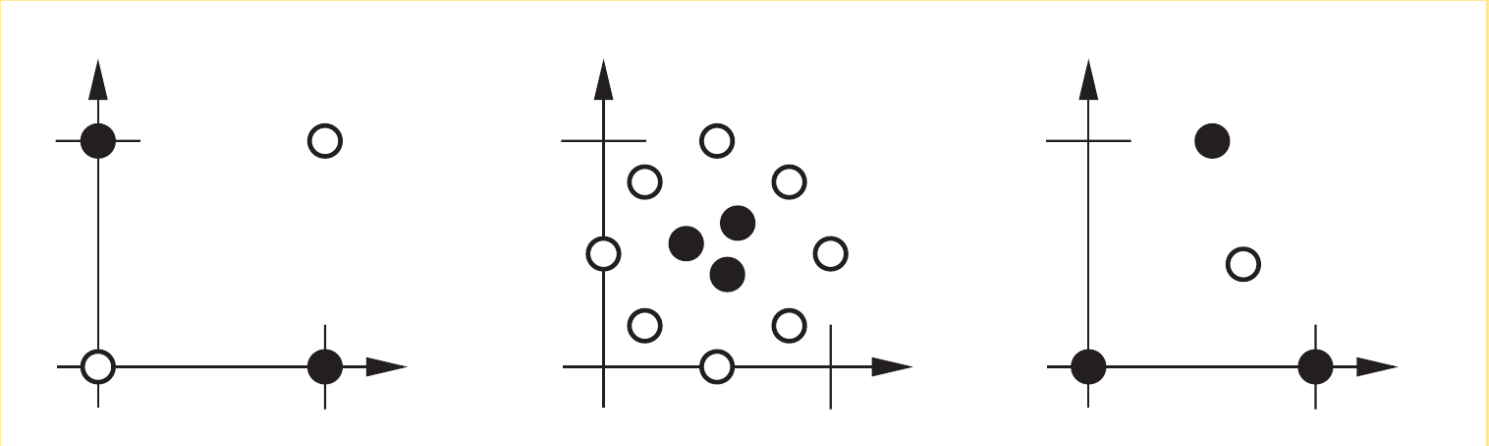
$$\mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{e}\mathbf{p}^T$$

$$\mathbf{b}^{new} = \mathbf{b}^{old} + \mathbf{e}$$

توانائی یادگیری با قاعده پرسپترون

قاعده یادگیری پرسپترون همواره به وزن‌هایی همگرایی می‌شود که می‌توانند به درستی دسته‌بندی مطلوب را انجام دهند، تنها اگر، چنین بردار وزنی وجود داشته باشد.

برای مسائل بسیاری، مرز تصمیم‌گیری خطی وجود ندارد.
اصطلاحاً نمونه‌ها «خطی جدائی‌پذیر» نیستند.



الگوریتم یادگیری پرسپترون

- Given a sequence of examples and labels, $\{\boldsymbol{p}_i, t_i\}$
- Initialize: $\boldsymbol{w} = 0$

for i=1,2,...

e = t_i – $hardlim(\boldsymbol{w}^{\mathrm{T}} \boldsymbol{p}_i)$

if e ≠ 0 /* mistake */

$\boldsymbol{w} = \boldsymbol{w} + e\, \boldsymbol{p}_i$

الگوریتم یادگیری پرسپترون

- Given a sequence of examples and labels, $\{\boldsymbol{p}_i, t_i\}$
- Initialize: $\boldsymbol{w} = 0$

هر نمونه‌ی یادگیری ممکن است
بارها در چرخه یادگیری استفاده شود

for i=1,2,...

$$e = t_i - hardlim(\boldsymbol{w}^T \boldsymbol{p}_i)$$

if $e \neq 0$ /* mistake */

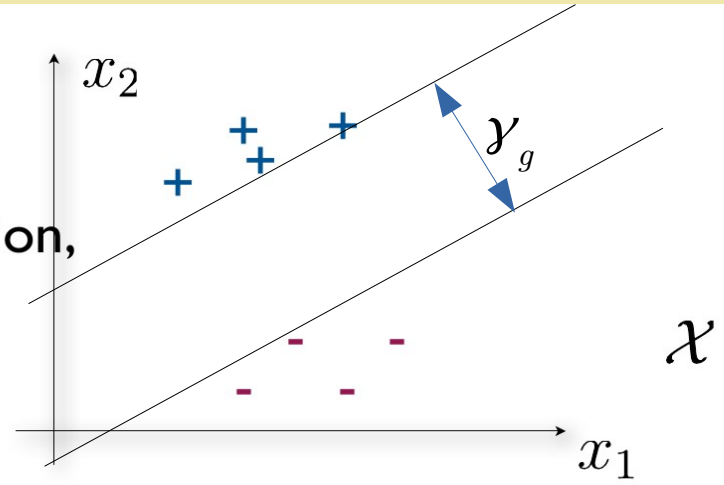
$$\boldsymbol{w} = \boldsymbol{w} + e\,\boldsymbol{p}_i$$

در اینجا، شرایط همگرائی
می‌بایست بررسی
و حلقه break شود

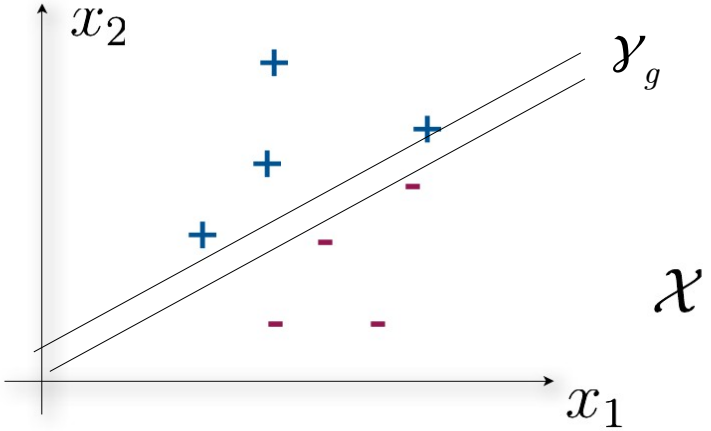
Geometric Margin

حاشیه هندسی

Good separation,
few mistakes



Poor separation,
many mistakes



همگرائی الگوریتم یادگیری پرسپترون

الگوریتم یادگیری پرسپترون، همیشه، پس از تعداد متناهی بهنگام سازی وزن همگرا می‌شود مگر آنکه نمونه‌های دو کلاس «خطی جدائی‌پذیر» نباشند.

$$\text{تعداد بار بهنگام سازی وزن} \leqslant \frac{R^2}{\gamma_g^2}$$

$$\|\boldsymbol{p}_i\| \leqslant R\ , \quad for\ i=1,\ldots,N$$

$$\gamma_g : geometric\ margin\ of\ samples$$