بسم الله الرحمن الرحیم

شبکه‌های عصبی مصنوعی

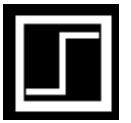
درس ۴

قاعده‌ی یادگیری پرسپترون

Perceptron Learning Rule

کاظم فولادی قلعه
دانشکده مهندسی، پردیس فارابی
دانشگاه تهران

http://courses.fouladi.ir/nn



Perceptron Learning Rule

پرسپترون

قاعده‌ی یادگیری

چگونه می‌توانیم ماتریس وزن و بردار بایاس را
برای شبکه‌های پرسپترون با چند ورودی (d بعد) تعیین کنیم؟

الگوریتم آموزش شبکه‌های پرسپترون: برای یادگیری حل مسئله‌ی طبقه‌بندی

✍ معرفی پرسپترون (توسط فرانک روزنبلات، اواخر دهه‌ی 1950)
- ❑ معرفی قاعده‌ی یادگیری برای آموزش پرسپترون
- ❑ اثبات همگرایی به وزن‌های صحیح برای حل مسئله (در صورت وجود این وزن‌ها)

✍ کشف محدودیت‌های ذاتی پرسپترون (توسط مینسکی و پاپرت، 1969)
- ❑ عدم توانایی در پیاده‌سازی برخی توابع ابتدائی (مانند xor)

✍ معرفی پرسپترون چندلایه و الگوریتم یادگیری آن (در اواخر دهه‌ی 1980)
- ❑ رفع محدودیت‌ها

اهمیت پرسپترون
- سریع بودن
- قابل اعتماد (برای مسائلی که می‌تواند آنها را حل کند)
- دارای مبانی خوب برای درک شبکه‌های پیچیده‌تر

شبکه‌های عصبی مصنوعی

قاعده‌ی یادگیری پرسپترون

۱

قواعد یادگیری

قاعده‌های یادگیری

LEARNING RULES

قاعده‌ی یادگیری *Learning Rule*	روالی برای تغییر وزن‌ها و بایاس‌های شبکه	الگوریتم آموزش *Training Algorithm*

هدف: آموزش شبکه برای انجام یک کار خاص

Learning Rules

- Supervised Learning

  Network is provided with a set of examples of proper network behavior (inputs/targets)

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

- Reinforcement Learning

  Network is only provided with a grade, or score, which indicates network performance

- Unsupervised Learning

  Only network inputs are available to the learning algorithm. Network learns to categorize (cluster) the inputs.

قاعده‌های یادگیری

انواع

LEARNING RULES

قاعده‌ی یادگیری *Learning Rule*	روالی برای تغییر وزن‌ها و بایاس‌های شبکه	الگوریتم آموزش *Training Algorithm*

با نظارت *Supervised*	بدون نظارت *Unsupervised*	تقویتی *Reinforcement*

قاعده‌های یادگیری

یادگیری با نظارت

SUPERVISED LEARNING

الگوریتم آموزش *Training Algorithm*	روالی برای تغییر وزن‌ها و بایاس‌های شبکه	قاعده‌ی یادگیری *Learning Rule*

با نظارت *Supervised*	بدون نظارت *Unsupervised*	تقویتی *Reinforcement*

قاعده‌ی یادگیری از مجموعه‌ای از مثال‌ها (مجموعه‌ی آموزشی) استفاده می‌کند که رفتار مناسب شبکه را نشان می‌دهد.

$\mathbf{p}_q$: ورودی شبکه (percept)
$\mathbf{t}_q$: خروجی درست شبکه (target)

مجموعه‌ی آموزشی
Training Set

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

- ورودی به شبکه داده می‌شود.
- خروجی شبکه با target مقایسه می‌شود.
- قاعده‌ی یادگیری، وزن‌ها را به‌گونه‌ای تغییر می‌دهد که خروجی‌های شبکه به با targetها نزدیک‌تر شود.

قاعده‌ی یادگیری پرسپترون، نمونه‌ای از قواعد یادگیری بانظارت است.

قاعده‌های یادگیری

یادگیری بدون نظارت

UNSUPERVISED LEARNING

الگوریتم آموزش *Training Algorithm*	روالی برای تغییر وزن‌ها و بایاس‌های شبکه	قاعده‌ی یادگیری *Learning Rule*

تقویتی *Reinforcement*	**بدون نظارت** ***Unsupervised***	با نظارت *Supervised*

وزن‌ها و بایاس‌ها تنها در پاسخ به ورودی‌های شبکه تغییر پیدا می‌کنند.
خروجی target موجود نیست.

بیشتر الگوریتم‌های بدون نظارت، نوعی خوشه‌بندی انجام می‌دهند:
یادگیری دسته‌بندی ورودی‌ها در تعدادی خوشه‌ی از قبل نامعلوم

قاعده‌های یادگیری

یادگیری تقویتی

REINFORCEMENT LEARNING

قاعده‌ی یادگیری *Learning Rule*	روالی برای تغییر وزن‌ها و بایاس‌های شبکه	الگوریتم آموزش *Training Algorithm*

با نظارت *Supervised*	بدون نظارت *Unsupervised*	تقویتی *Reinforcement*

شبیه روش با نظارت، اما به‌جای جواب درست هر ورودی، صرفاً یک امتیاز (نمره) به آن داده می‌شود.
این امتیاز، معیاری از کارآیی شبکه بر روی یک دنباله از ورودی‌است.

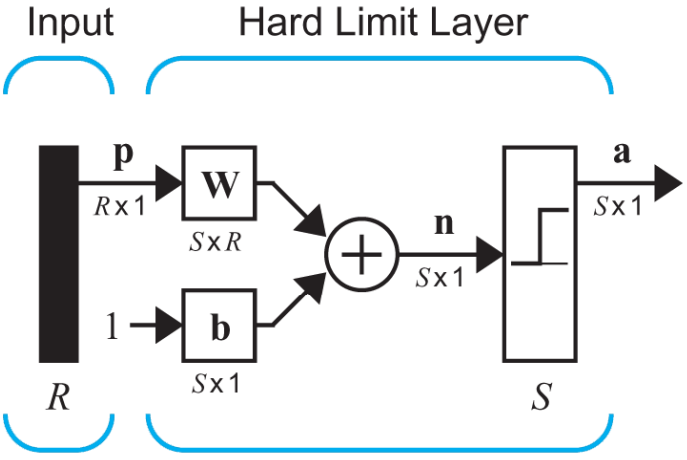
شبکه‌های عصبی مصنوعی

قاعده‌ی یادگیری پرسپترون

۲

معماری پرسپترون

Perceptron Architecture



$$\mathbf{a} = \mathbf{hardlim}(\mathbf{Wp}+\mathbf{b})$$

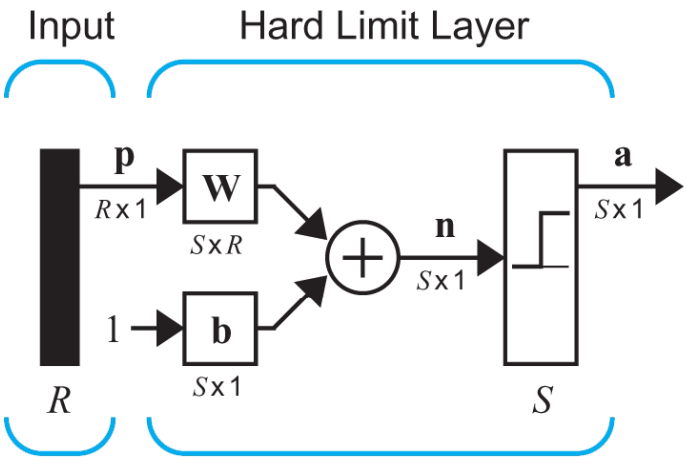
$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

$${}_i\mathbf{w} = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix} \qquad \mathbf{W} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \\ \vdots \\ {}_S\mathbf{w}^T \end{bmatrix}$$

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

معماری پرسپترون

PERCEPTRON ARCHITECTURE



$$\mathbf{a} = \mathbf{hardlim}(\mathbf{Wp}+\mathbf{b})$$

یا

hardlims

$$\mathbf{W} = \begin{bmatrix} w_{1,1} & w_{1,2} & \cdots & w_{1,R} \\ w_{2,1} & w_{2,2} & \cdots & w_{2,R} \\ \vdots & \vdots & & \vdots \\ w_{S,1} & w_{S,2} & \cdots & w_{S,R} \end{bmatrix}$$

ماتریس وزن‌ها

$${}_i\mathbf{w} = \begin{bmatrix} w_{i,1} \\ w_{i,2} \\ \vdots \\ w_{i,R} \end{bmatrix}$$

بردار ستونی متشکل از عناصر سطر iام ماتریس $\mathbf{W}$

$$\mathbf{W} = \begin{bmatrix} {}_1\mathbf{w}^T \\ {}_2\mathbf{w}^T \\ \vdots \\ {}_S\mathbf{w}^T \end{bmatrix}$$

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

iامین عنصر خروجی شبکه

ضرب داخلی سطر iام ماتریس وزن و بردار ورودی

معماری پرسپترون

تابع انتقال حد سخت

HARD-LIMIT TRANSFER FUNCTION

Name	Input / Output Relation $a = f(n)$	Icon	MATLAB Function
Hard Limit	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$		hardlim
Symmetrical Hard Limit	$a = -1 \quad n < 0$ $a = +1 \quad n \geq 0$		hardlims

نتیجه:

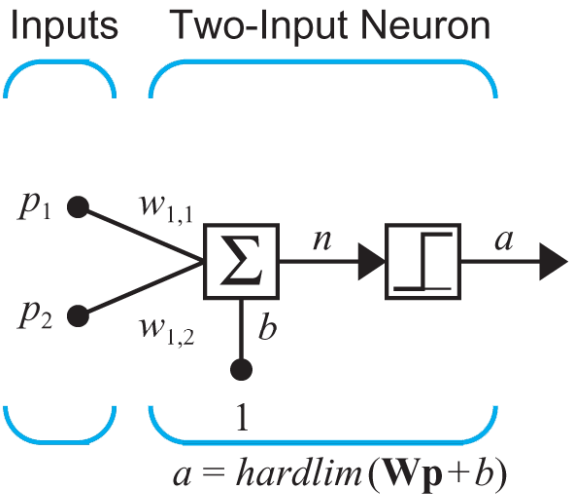
خروجی تابع انتقال پرسپترون **دو حالتی** است

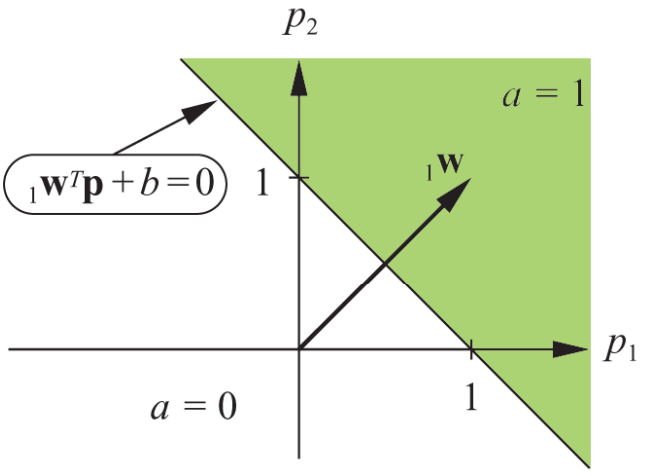
⇓

هر نرون در شبکه، فضای ورودی را به **دو** ناحیه تقسیم می‌کند.

Single-Neuron Perceptron

$w_{1,1} = 1 \qquad w_{1,2} = 1 \qquad b = -1$



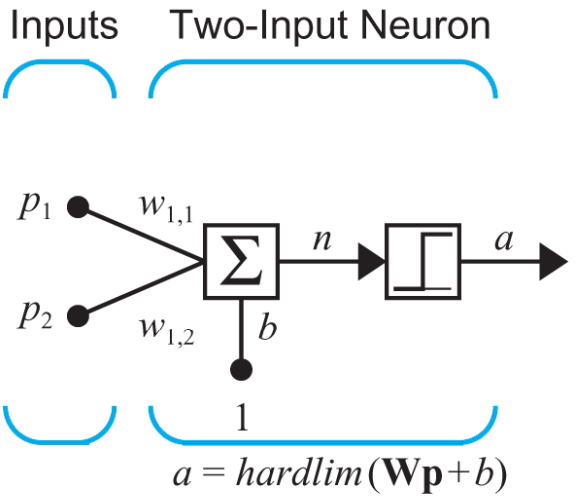


$$a = hardlim(_1\mathbf{w}^T\mathbf{p} + b) = hardlim(w_{1,1}p_1 + w_{1,2}p_2 + b)$$

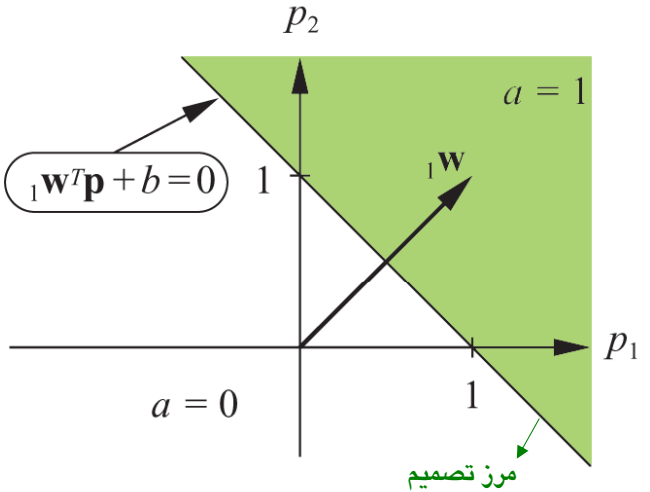
پرسپترون تک-نرونی

SINGLE-NEURON PERCEPTRON

پرسپترون دو-ورودی، تک-خروجی با یک نرون



$$w_{1,1} = 1 \qquad w_{1,2} = 1 \qquad b = -1$$



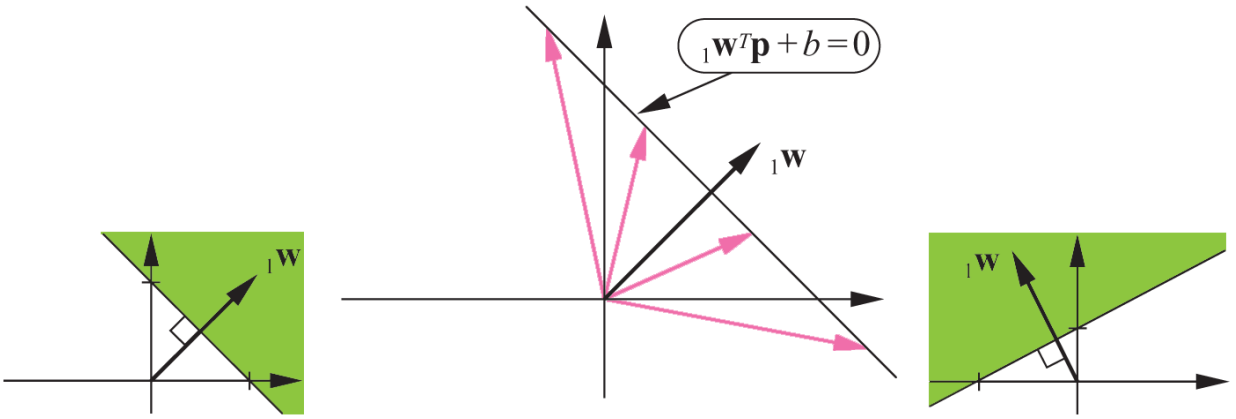
متشکل از بردارهای ورودی که برای آنها ورودی خالص نرون صفر باشد: $n=0$.

$$a = hardlim(_1\mathbf{w}^T\mathbf{p} + b) = hardlim(w_{1,1}p_1 + w_{1,2}p_2 + b)$$

Decision Boundary

$${}_1\mathbf{w}^T\mathbf{p} + b = 0 \qquad {}_1\mathbf{w}^T\mathbf{p} = -b$$

- All points on the decision boundary have the same inner product with the weight vector.
- Therefore they have the same projection onto the weight vector, and they must lie on a line orthogonal to the weight vector



پرسپترون تک-ورونی

مرز تصمیم

DECISOON BOUNDARY

مرز تصمیم: متشکل از بردارهای ورودی که برای آنها ورودی خالص نرون صفر باشد: $n=0$.
مرز تصمیم یک خط در فضای ورودی است:

$$ {}_1\mathbf{w}^{\mathrm{T}}\mathbf{p} + b = 0 \qquad {}_1\mathbf{w}^{\mathrm{T}}\mathbf{p} = -b $$

$$ n = {}_1\mathbf{w}^{T}\mathbf{p} + b = w_{1,1}p_1 + w_{1,2}p_2 + b = 0 $$

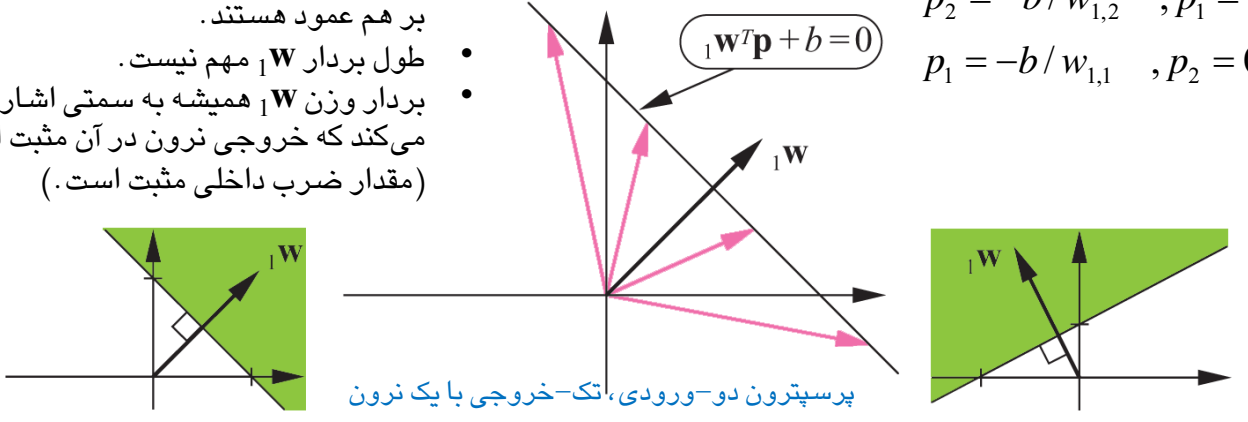
خروجی در یک سمت خط صفر، روی خط و سمت دیگر آن یک است.

در محل تقاطع با محورها داریم:

$$ p_2 = -b / w_{1,2} \quad , p_1 = 0 $$

$$ p_1 = -b / w_{1,1} \quad , p_2 = 0 $$

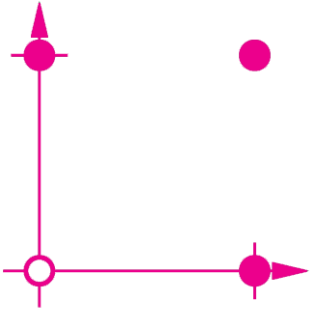
- بردار وزن ${}_1\mathbf{w}$ و مرز تصمیم‌گیری همیشه بر هم عمود هستند.
- طول بردار ${}_1\mathbf{w}$ مهم نیست.
- بردار وزن ${}_1\mathbf{w}$ همیشه به سمتی اشاره می‌کند که خروجی نرون در آن مثبت است. (مقدار ضرب داخلی مثبت است.)



پرسپترون دو-ورودی، تک-خروجی با یک نرون

Example - OR

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

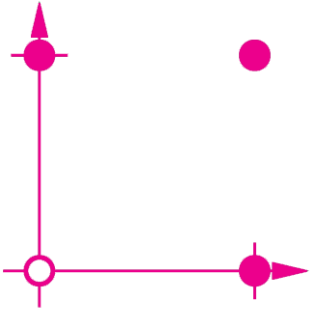


پرسپترون تک-نرونی

مثال: OR (۱ از ۳)

DECISOON BOUNDARY

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

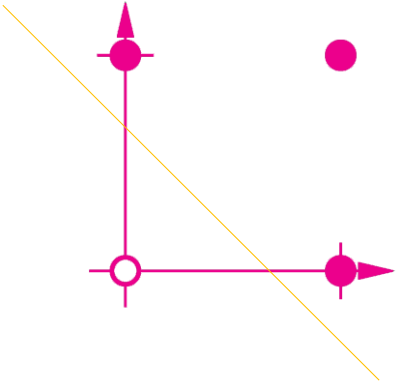


پرسپترون تک-نرونی

مثال: OR (۲ از ۳)

DECISOON BOUNDARY

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



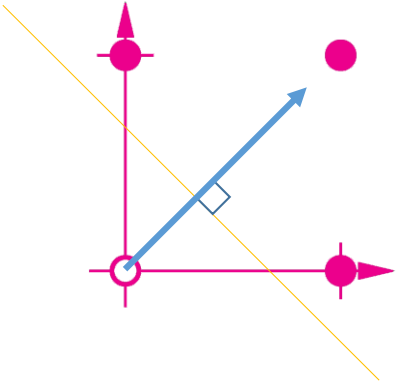
تعیین یک خط دلخواه جدا کننده‌ی دو کلاس

پرسپترون تک-نرونی

مثال: OR (۳ از ۳)

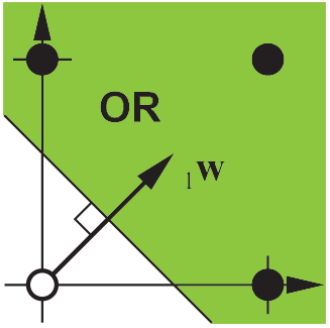
DECISOON BOUNDARY

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



تعیین بردار نرمال خط جدا کننده‌ی دو کلاس

OR Solution



Weight vector should be orthogonal to the decision boundary.

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

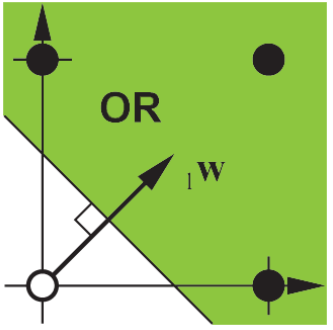
Pick a point on the decision boundary to find the bias.

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

پرسپترون تک-نرونی

مثال: OR: راه حل

DECISOON BOUNDARY

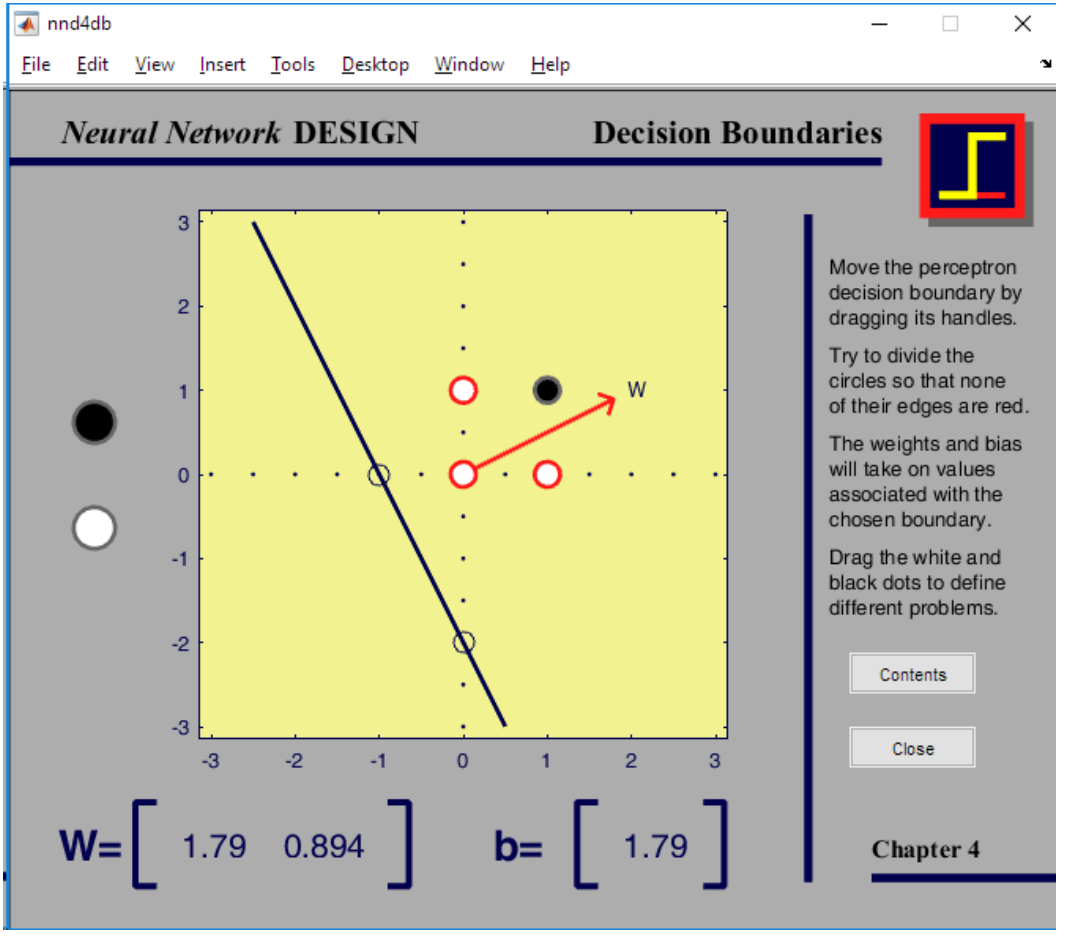


بردار وزن: باید عمود بر مرز تصمیم باشد:

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

بایاس: برای یافتن بایاس یک نقطه روی مرز تصمیم را انتخاب می‌کنیم:

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



MATLAB

```
>> nnd4db
```

Multiple-Neuron Perceptron

Each neuron will have its own decision boundary.

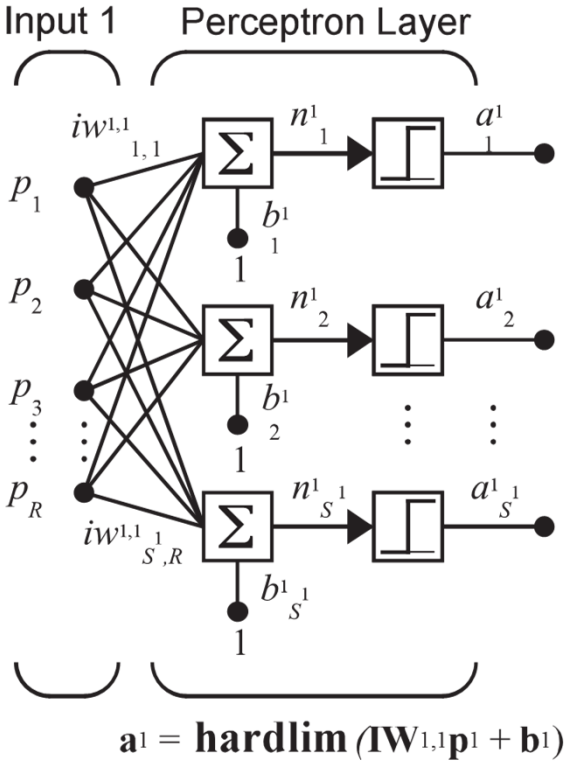
$$ {}_i\mathbf{w}^T\mathbf{p} + b_i = 0 $$

A single neuron can classify input vectors into two categories.

A multi-neuron perceptron can classify input vectors into 2^S categories.

پرسپترون چند–نرونی

MULTIPLE-NEURON PERCEPTRON



هر نرون، مرز تصمیم خودش را دارد:

مرز تصمیم برای نرون i عبارت است از:

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

یک نرون واحد، فضای ورودی را به دو دسته تقسیم می‌کند.

در پرسپترون چند–نرونی، S نرون داریم

⇓

فضای ورودی به 2^S دسته تقسیم می‌شود.

قاعده‌ی یادگیری پرسپترون

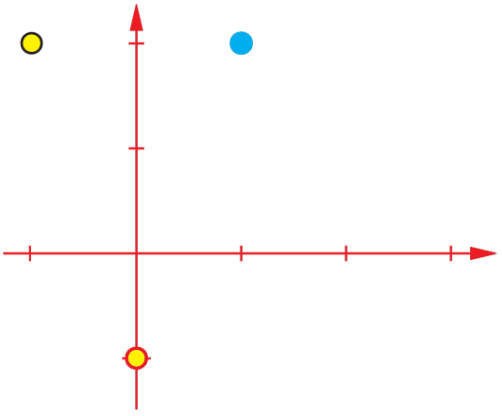
۳

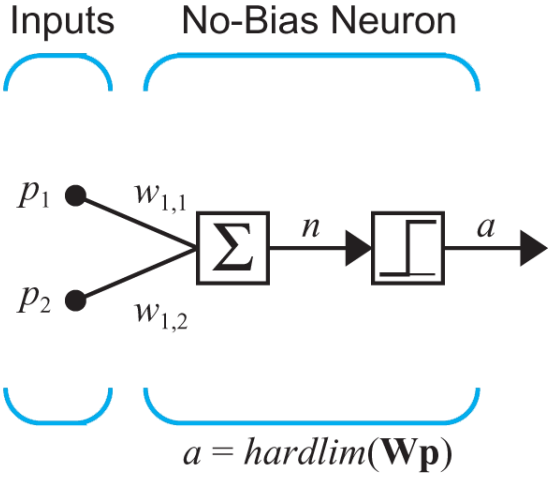
قاعده‌ی یادگیری پرسپترون

Learning Rule Test Problem

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

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





قاعده‌ی یادگیری پرسپترون

مسئله‌ی آزمایشی

PERCEPTRON LEARNING RULE TEST PROBLEM

روش یادگیری با نظارت

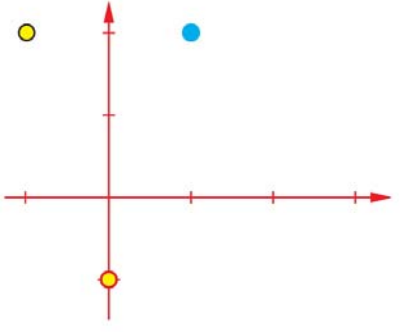
$\mathbf{p}_q$: ورودی شبکه (percept)
$\mathbf{t}_q$: خروجی درست شبکه (target)

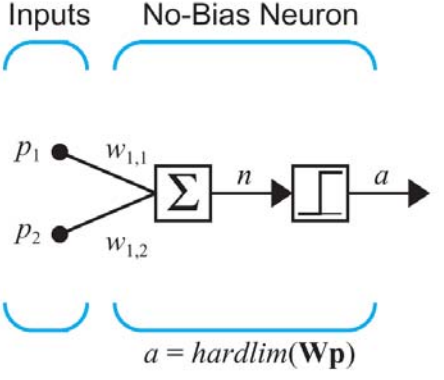
$$\{\mathbf{p}_1, \mathbf{t}_1\}, \{\mathbf{p}_2, \mathbf{t}_2\}, \ldots, \{\mathbf{p}_Q, \mathbf{t}_Q\}$$

مجموعه‌ی مثال‌ها:

مجموعه‌ی آموزشی
Training Set

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

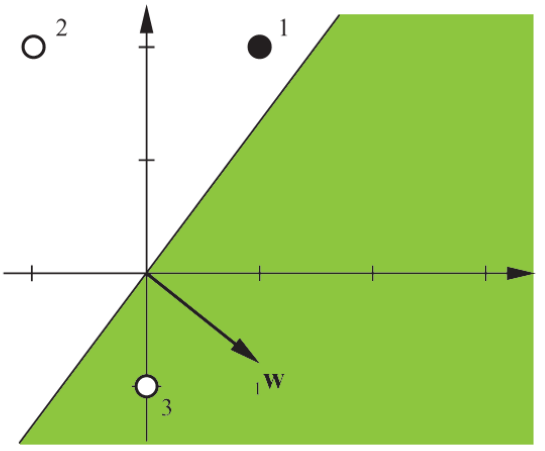




Starting Point

Random initial weight:

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



Present $\mathbf{p}_1$ to the network:

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

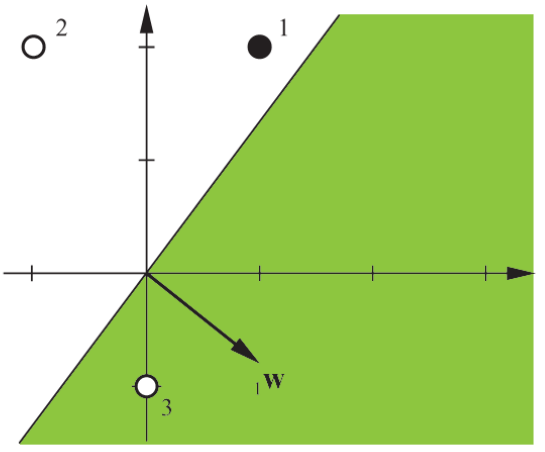
$$ a = hardlim(-0.6) = 0 $$

Incorrect Classification.

قاعده‌ی یادگیری پرسپترون

نقطه‌ی شروع

STARTING POINT



با مقادیر اولیه‌ی تصادفی وزن‌ها شروع می‌کنیم:

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

بردار ورودی $\mathbf{p}_1$ را به شبکه نشان می‌دهیم:

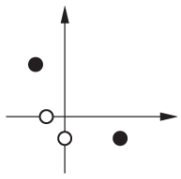
$$ a = hardlim({}_1\mathbf{w}^T \mathbf{p}_1) = hardlim\left(\begin{bmatrix} 1.0 & -0.8 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \end{bmatrix} \right) $$

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

Incorrect Classification. طبقه‌بندی نادرست

Tentative Learning Rule

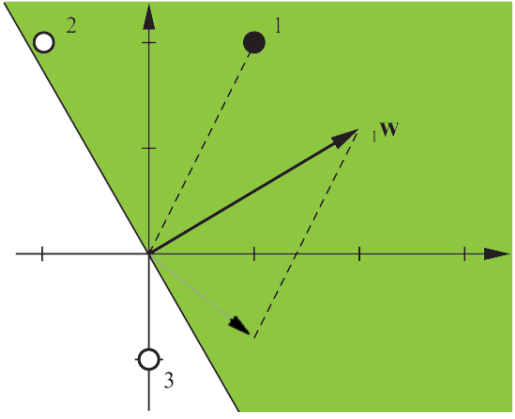
- Set ${}_1\mathbf{w}$ to $\mathbf{p}_1$
  - Not stable ×



- Add $\mathbf{p}_1$ to ${}_1\mathbf{w}$ ✓

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

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}_1 = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix}$$



قاعده‌ی یادگیری پرسپترون

قاعده‌ی تکراری یادگیری

TENTATIVE LEARNING RULE

وقتی خروجی نرون برای ورودی $\mathbf{p}_1$ درست نیست، می‌توانیم:

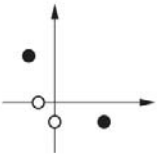
۱) قرار دهیم ${}_1\mathbf{w} \leftarrow \mathbf{p}_1$

- Set ${}_1\mathbf{w}$ to $\mathbf{p}_1$
  - Not stable ×

مشکل این است که

با دیدن نمونه‌های بعدی نادرست هم ${}_1\mathbf{w} \leftarrow \mathbf{p}$ می‌شود

⇐ عدم همگرایی وزن‌ها ⇐ عدم پایداری شبکه



۲) قرار دهیم ${}_1\mathbf{w} \leftarrow {}_1\mathbf{w} + \mathbf{p}_1$

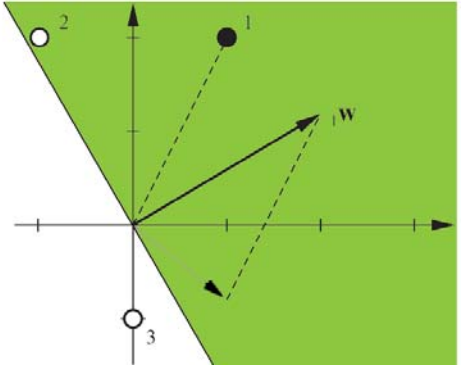
- Add $\mathbf{p}_1$ to ${}_1\mathbf{w}$ ✓

⇐ جهت دهی بیشتر ${}_1\mathbf{w}$ به سمت $\mathbf{p}_1$

حرکت مجانبی ${}_1\mathbf{w}$ به جهت $\mathbf{p}_1$ با دیدن نمونه‌های بیشتر

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

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}_1 = \begin{bmatrix} 1.0 \\ -0.8 \end{bmatrix} + \begin{bmatrix} 1 \\ 2 \end{bmatrix} = \begin{bmatrix} 2.0 \\ 1.2 \end{bmatrix}$$



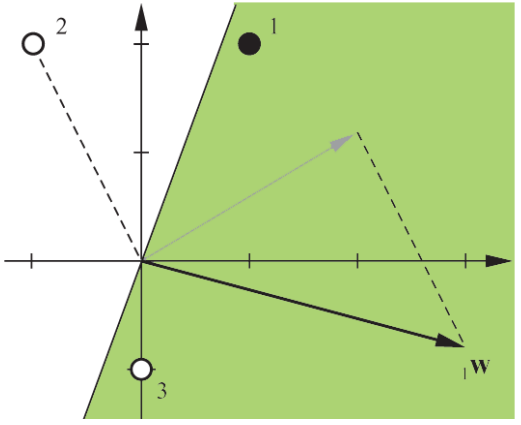
Second Input Vector

$$a = hardlim({}_1\mathbf{w}^T\mathbf{p}_2) = hardlim\left(\begin{bmatrix}2.0 & 1.2\end{bmatrix}\begin{bmatrix}-1\\2\end{bmatrix}\right)$$

$$a = hardlim(0.4) = 1 \quad \text{(Incorrect Classification)}$$

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

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_2 = \begin{bmatrix}2.0\\1.2\end{bmatrix} - \begin{bmatrix}-1\\2\end{bmatrix} = \begin{bmatrix}3.0\\-0.8\end{bmatrix}$$



قاعده‌ی یادگیری پرسپترون

بردار ورودی دوم

SECOND INPUT VECTOR

$$a = hardlim({}_1\mathbf{w}^T\mathbf{p}_2) = hardlim\left(\begin{bmatrix}2.0 & 1.2\end{bmatrix}\begin{bmatrix}-1\\2\end{bmatrix}\right)$$

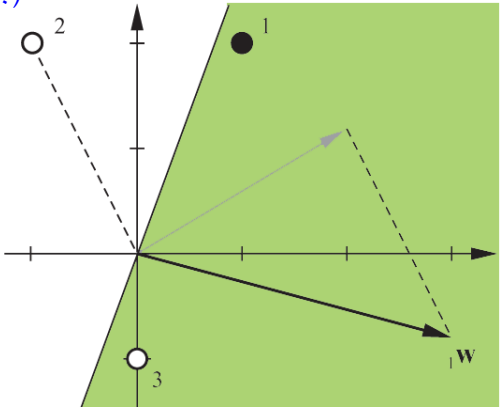
$$a = hardlim(0.4) = 1 \quad \text{(Incorrect Classification)}$$

طبقه‌بندی نادرست

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

برای الگوها با target صفر
(برای دور کردن از ورودی)

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_2 = \begin{bmatrix}2.0\\1.2\end{bmatrix} - \begin{bmatrix}-1\\2\end{bmatrix} = \begin{bmatrix}3.0\\-0.8\end{bmatrix}$$

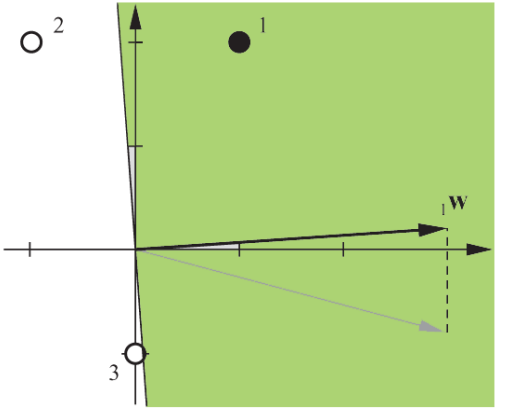


Third Input Vector

$$a = hardlim({}_1\mathbf{w}^T \mathbf{p}_3) = hardlim\left(\begin{bmatrix}3.0 & -0.8\end{bmatrix}\begin{bmatrix}0\\-1\end{bmatrix}\right)$$

$$a = hardlim(0.8) = 1 \qquad \text{(Incorrect Classification)}$$

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_3 = \begin{bmatrix}3.0\\-0.8\end{bmatrix} - \begin{bmatrix}0\\-1\end{bmatrix} = \begin{bmatrix}3.0\\0.2\end{bmatrix}$$



Patterns are now correctly classified.

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

قاعده‌ی یادگیری پرسپترون

بردار ورودی سوم

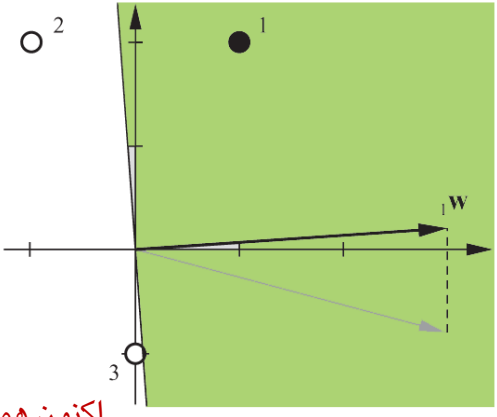
THIRD INPUT VECTOR

$$a = hardlim({}_1\mathbf{w}^T \mathbf{p}_3) = hardlim\left(\begin{bmatrix}3.0 & -0.8\end{bmatrix}\begin{bmatrix}0\\-1\end{bmatrix}\right)$$

$$a = hardlim(0.8) = 1 \quad \text{(Incorrect Classification)}$$

طبقه‌بندی نادرست

$${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}_3 = \begin{bmatrix}3.0\\-0.8\end{bmatrix} - \begin{bmatrix}0\\-1\end{bmatrix} = \begin{bmatrix}3.0\\0.2\end{bmatrix}$$



اکنون همه‌ی الگوها درست طبقه‌بندی شده‌اند.

Patterns are now correctly classified.

در حالتی که target با خروجی مساوی باشد.

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

Unified Learning Rule

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

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

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

$$e = t - a$$

If $e = 1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$

If $e = -1$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$

If $e = 0$, then ${}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$

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

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

A bias is a weight with an input of 1.

قاعده‌ی یادگیری پرسپترون

قاعده‌ی یادگیری یک‌دست

UNIFIED LEARNING RULE

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

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

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

$$e = t - a$$

$$\text{If } e = 1, \text{ then } {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} + \mathbf{p}$$

$$\text{If } e = -1, \text{ then } {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old} - \mathbf{p}$$

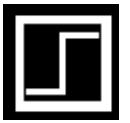
$$\text{If } e = 0, \text{ then } {}_1\mathbf{w}^{new} = {}_1\mathbf{w}^{old}$$

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

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

چون بایاس وزنی است
که ورودی آن 1 است.

Multiple-Neuron Perceptrons



To update the *i*th row of the weight matrix:

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

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

Matrix form:

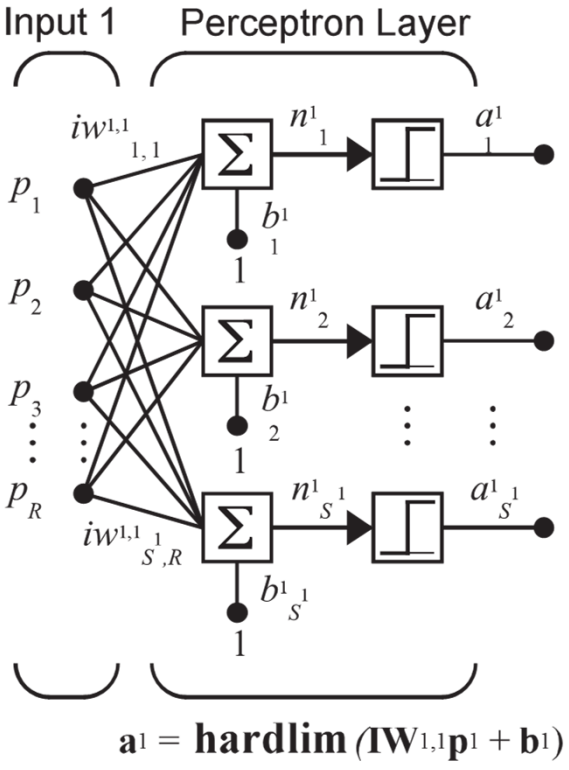
$$ \mathbf{W}^{new} = \mathbf{W}^{old} + \mathbf{e}\mathbf{p}^T $$

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

قاعده‌ی یادگیری پرسپترون

پرسپترون چند نرونی

MULTIPLE-NEURON PERCEPTRONS



برای به‌هنگام‌سازی سطر iام ماتریس وزن:

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

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

فرم ماتریسی:

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

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

Apple/Banana Example

Training Set

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

Initial Weights

$$\mathbf{W} = \begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} \qquad b = 0.5$$

First Iteration

$$a = hardlim(\mathbf{W}\mathbf{p}_1 + b) = hardlim\left(\begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix}\begin{bmatrix} -1 \\ 1 \\ -1 \end{bmatrix} + 0.5\right)$$

$$a = hardlim(-0.5) = 0 \qquad e = t_1 - a = 1 - 0 = 1$$

$$\mathbf{W}^{new} = \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix} 0.5 & -1 & -0.5 \end{bmatrix} + (1)\begin{bmatrix} -1 & 1 & -1 \end{bmatrix} = \begin{bmatrix} -0.5 & 0 & -1.5 \end{bmatrix}$$

$$b^{new} = b^{old} + e = 0.5 + (1) = 1.5$$

قاعده‌ی یادگیری پرسپترون

مثال موز / سیب

APPLE/BANANA EXAMPLE

Training Set : مجموعه‌ی آموزشی

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

Initial Weights : وزن‌های آغازین

$$\mathbf{W} = \begin{bmatrix}0.5 & -1 & -0.5\end{bmatrix} \qquad b = 0.5$$

First Iteration : تکرار اول

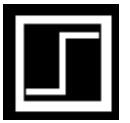
$$a = hardlim(\mathbf{W}\mathbf{p}_1 + b) = hardlim\left(\begin{bmatrix}0.5 & -1 & -0.5\end{bmatrix}\begin{bmatrix}-1\\1\\-1\end{bmatrix} + 0.5\right)$$

$$a = hardlim(-0.5) = 0 \qquad e = t_1 - a = 1 - 0 = 1$$

$$\mathbf{W}^{new} = \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix}0.5 & -1 & -0.5\end{bmatrix} + (1)\begin{bmatrix}-1 & 1 & -1\end{bmatrix} = \begin{bmatrix}-0.5 & 0 & -1.5\end{bmatrix}$$

$$b^{new} = b^{old} + e = 0.5 + (1) = 1.5$$

Second Iteration



$$a = hardlim\left(\mathbf{W}\mathbf{p}_2 + b\right) = hardlim\left(\begin{bmatrix}-0.5 & 0 & -1.5\end{bmatrix}\begin{bmatrix}1\\1\\-1\end{bmatrix} + (1.5)\right)$$

$$a = hardlim\,(2.5) = 1$$

$$e = t_2 - a = 0 - 1 = -1$$

$$\mathbf{W}^{new} = \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix}-0.5 & 0 & -1.5\end{bmatrix} + (-1)\begin{bmatrix}1 & 1 & -1\end{bmatrix} = \begin{bmatrix}-1.5 & -1 & -0.5\end{bmatrix}$$

$$b^{new} = b^{old} + e = 1.5 + (-1) = 0.5$$

قاعده‌ی یادگیری پرسپترون

مثال موز / سیب: تکرار دوم

APPLE/BANANA EXAMPLE: SECOND ITERATION

$$a = hardlim(\mathbf{W}\mathbf{p}_2 + b) = hardlim\left(\begin{bmatrix}-0.5 & 0 & -1.5\end{bmatrix}\begin{bmatrix}1\\1\\-1\end{bmatrix} + (1.5)\right)$$

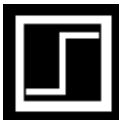
$$a = hardlim(2.5) = 1$$

$$e = t_2 - a = 0 - 1 = -1$$

$$\mathbf{W}^{new} = \mathbf{W}^{old} + e\mathbf{p}^T = \begin{bmatrix}-0.5 & 0 & -1.5\end{bmatrix} + (-1)\begin{bmatrix}1 & 1 & -1\end{bmatrix} = \begin{bmatrix}-1.5 & -1 & -0.5\end{bmatrix}$$

$$b^{new} = b^{old} + e = 1.5 + (-1) = 0.5$$

Check



$$a = hardlim\,(\mathbf{W}\mathbf{p}_1 + b) = hardlim\left(\begin{bmatrix}-1.5 & -1 & -0.5\end{bmatrix}\begin{bmatrix}-1\\1\\-1\end{bmatrix} + 0.5\right)$$

$$a = hardlim\,(1.5) = 1 = t_1$$

$$a = hardlim\,(\mathbf{W}\mathbf{p}_2 + b) = hardlim\left(\begin{bmatrix}-1.5 & -1 & -0.5\end{bmatrix}\begin{bmatrix}1\\1\\-1\end{bmatrix} + 0.5\right)$$

$$a = hardlim\,(-1.5) = 0 = t_2$$

قاعده‌ی یادگیری پرسپترون

مثال موز / سیب: بررسی

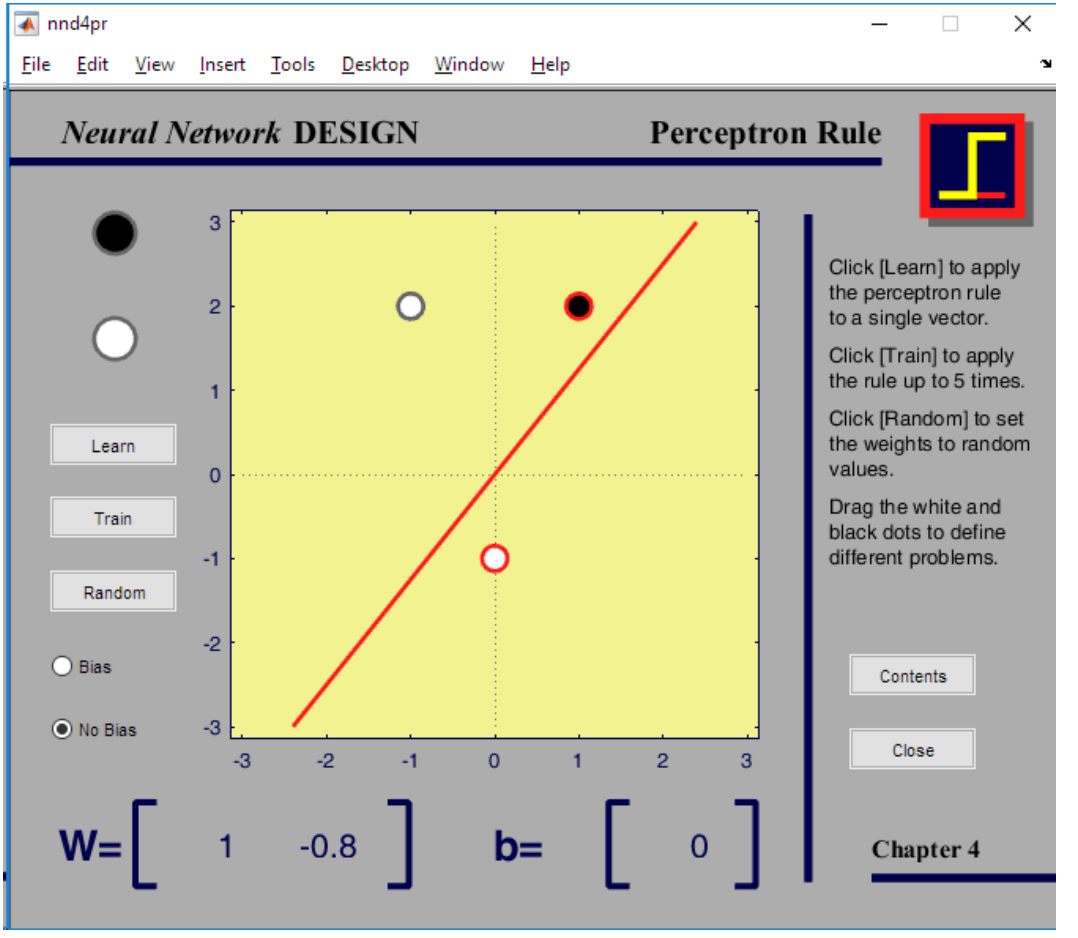
APPLE/BANANA EXAMPLE: CHECK

$$a = hardlim\,(\mathbf{W}\mathbf{p}_1 + b) = hardlim\,(\begin{bmatrix}-1.5 & -1 & -0.5\end{bmatrix}\begin{bmatrix}-1\\1\\-1\end{bmatrix} + 0.5)$$

$$a = hardlim\,(1.5) = 1 = t_1$$ ✓ درست

$$a = hardlim\,(\mathbf{W}\mathbf{p}_2 + b) = hardlim\,(\begin{bmatrix}-1.5 & -1 & -0.5\end{bmatrix}\begin{bmatrix}1\\1\\-1\end{bmatrix} + 0.5)$$

$$a = hardlim\,(-1.5) = 0 = t_2$$ ✓ درست



MATLAB

```
>> nnd4pr
```

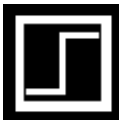
شبکه‌های عصبی مصنوعی

قاعده‌ی یادگیری پرسپترون

۴

اثبات همگرایی

Perceptron Rule Capability



The perceptron rule will always converge to weights which accomplish the desired classification, assuming that such weights exist.

قاعده‌ی یادگیری پرسپترون

اثبات همگرایی

PROOF OF CONVERGENCE

ثابت می‌شود که

قاعده‌ی یادگیری پرسپترون

همیشه به وزن‌هایی همگرا می‌شود که طبقه‌بندی مطلوب را انجام می‌دهد.

(در صورت وجود این وزن‌ها)

وزن‌ها در تعدادی متناهی مرتبه تغییر می‌کنند

$\Downarrow$

قاعده‌ی یادگیری پرسپترون در تعدادی متناهی تکرار همگرا می‌شود.

* حداکثر تعداد تکرارها (تغییر وزن‌ها) رابطه‌ی معکوس با مربع δ دارد.

(δ = معیار میزان نزدیکی مرز تصمیم به الگوهای ورودی)

$\Leftarrow$ اگر کلاس‌های ورودی به‌سختی جداپذیر خطی باشند (یعنی به مرز تصمیم نزدیک باشند)،

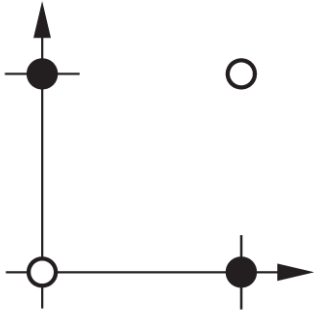
آن‌گاه تعداد تکرار برای همگرایی زیاد خواهد بود.

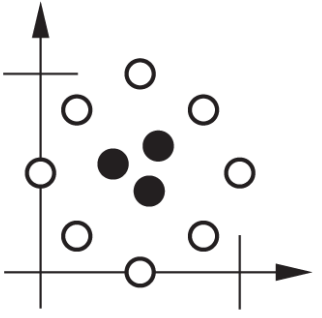
Perceptron Limitations

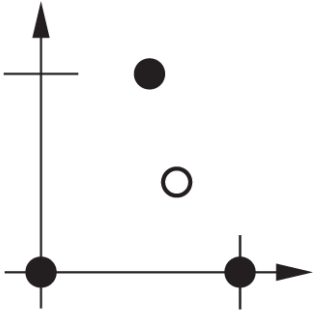
Linear Decision Boundary

$$_1\mathbf{w}^T\mathbf{p} + b = 0$$

Linearly Inseparable Problems







قاعده‌ی یادگیری پرسپترون

محدودیت‌های پرسپترون

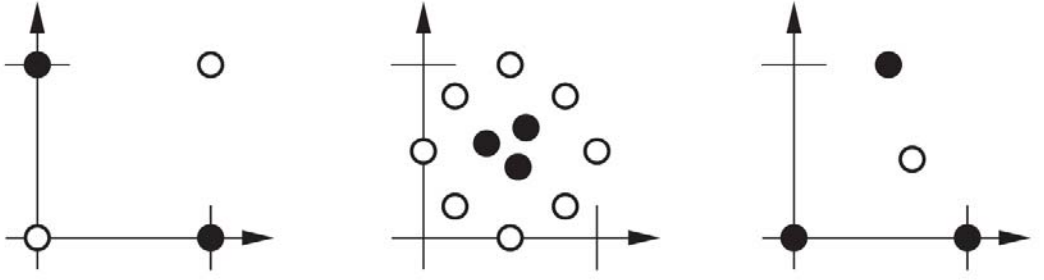
PERCEPTRON LIMITATIONS

مرز تصمیم خطی
Linear Decision Boundary

$$_1\mathbf{w}^T\mathbf{p} + b = 0$$

ابرصفحه

مسائل جدا ناپذیر به‌طور خطی
Linearly Inseparable Problems



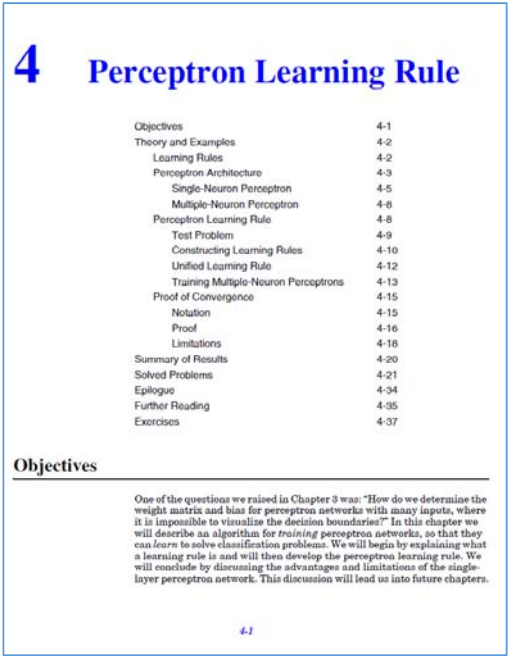
پرسپترون فقط می‌تواند الگوهایی را جداسازی کند که با یک مرز خطی قابل تفکیک باشند.

قاعده‌ی یادگیری پرسپترون

۵

منابع

منبع اصلی



4 Perceptron Learning Rule

Objectives

One of the questions we raised in Chapter 3 was: "How do we determine the weight matrix and bias for perceptron networks with many inputs, where it is impossible to visualize the decision boundaries?" In this chapter we will describe an algorithm for *training* perceptron networks, so that they can *learn* to solve classification problems. We will begin by explaining what a learning rule is and will then develop the perceptron learning rule. We will conclude by discussing the advantages and limitations of the single-layer perceptron network. This discussion will lead us into future chapters.

4-1

Martin T. Hagan, Howard B. Demuth, Mark H. Beale, Orlando De Jesus,
Neural Network Design,
2nd Edition, Martin Hagan, 2014.
Chapter 4

Online version can be downloaded from: http://hagan.okstate.edu/nnd.html