بسم الله الرحمن الرحیم

دانشگاه تربت حیدریه

پردازش سیگنال دیجیتال

فصل سوم :

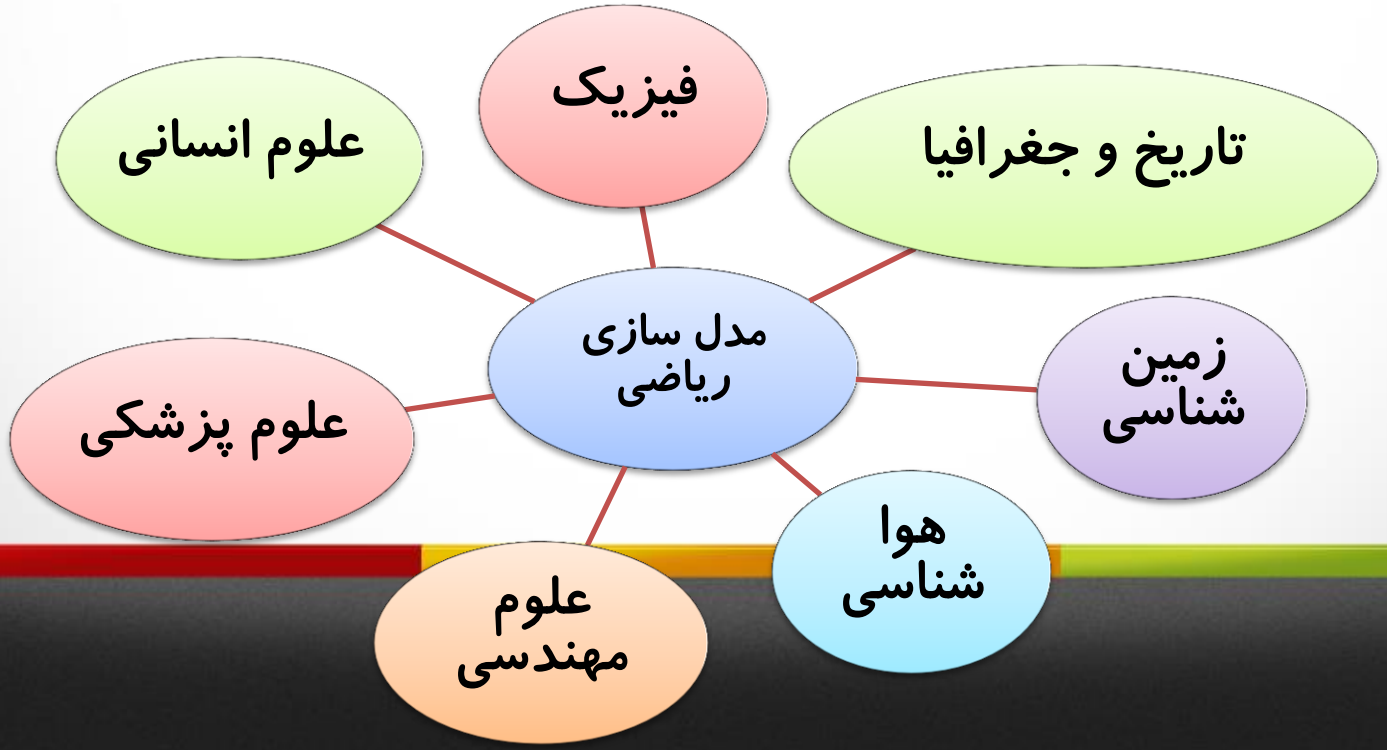
شبکه های عصبی

مجتبی قربانی

مدل :

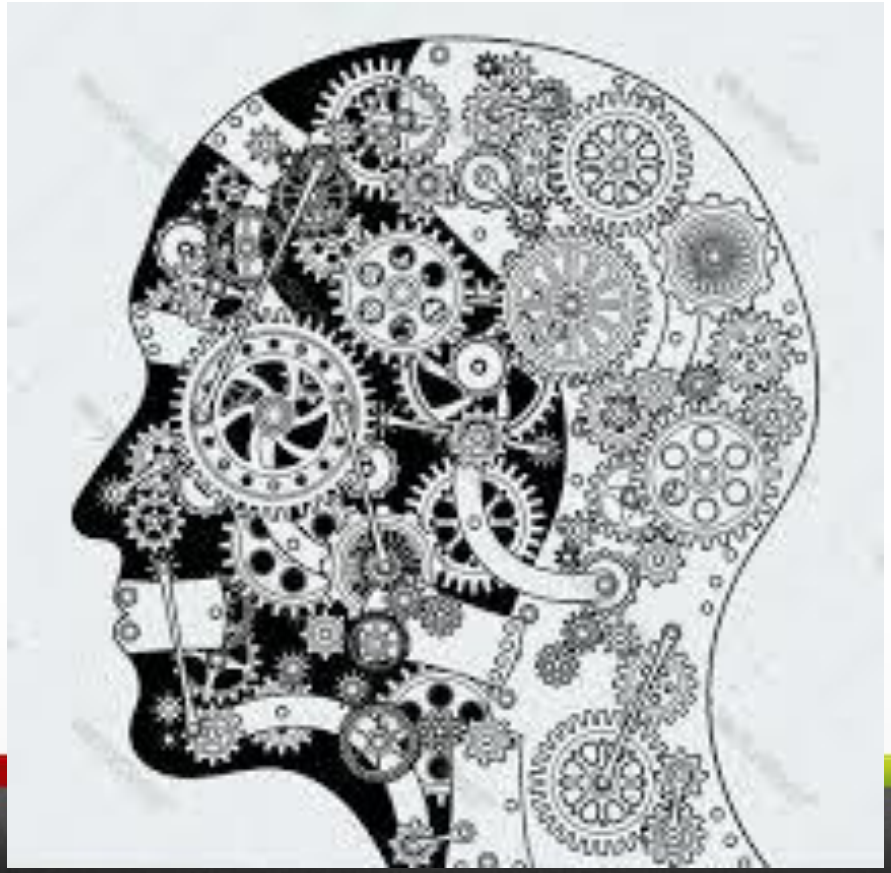
مدل ریاضی عبارت است از توصیف یک سامانه(سیستم) به کمک زبان ریاضی و قضیه‌ها و نمادهایش.

مدل‌سازی یا مدل‌سازی ریاضی عبارت است از تلاش برای توسعه ی یک مدل ریاضی برای یک سامانه مشخص.



مدل سازی در مکانیک :

در مهندسی مکانیک همواره بدست آوردن مدلی دقیق و کامل از یک مسئله، یکی از مهمترین و چالشی ترین مباحث به شمار رفته و می رود.

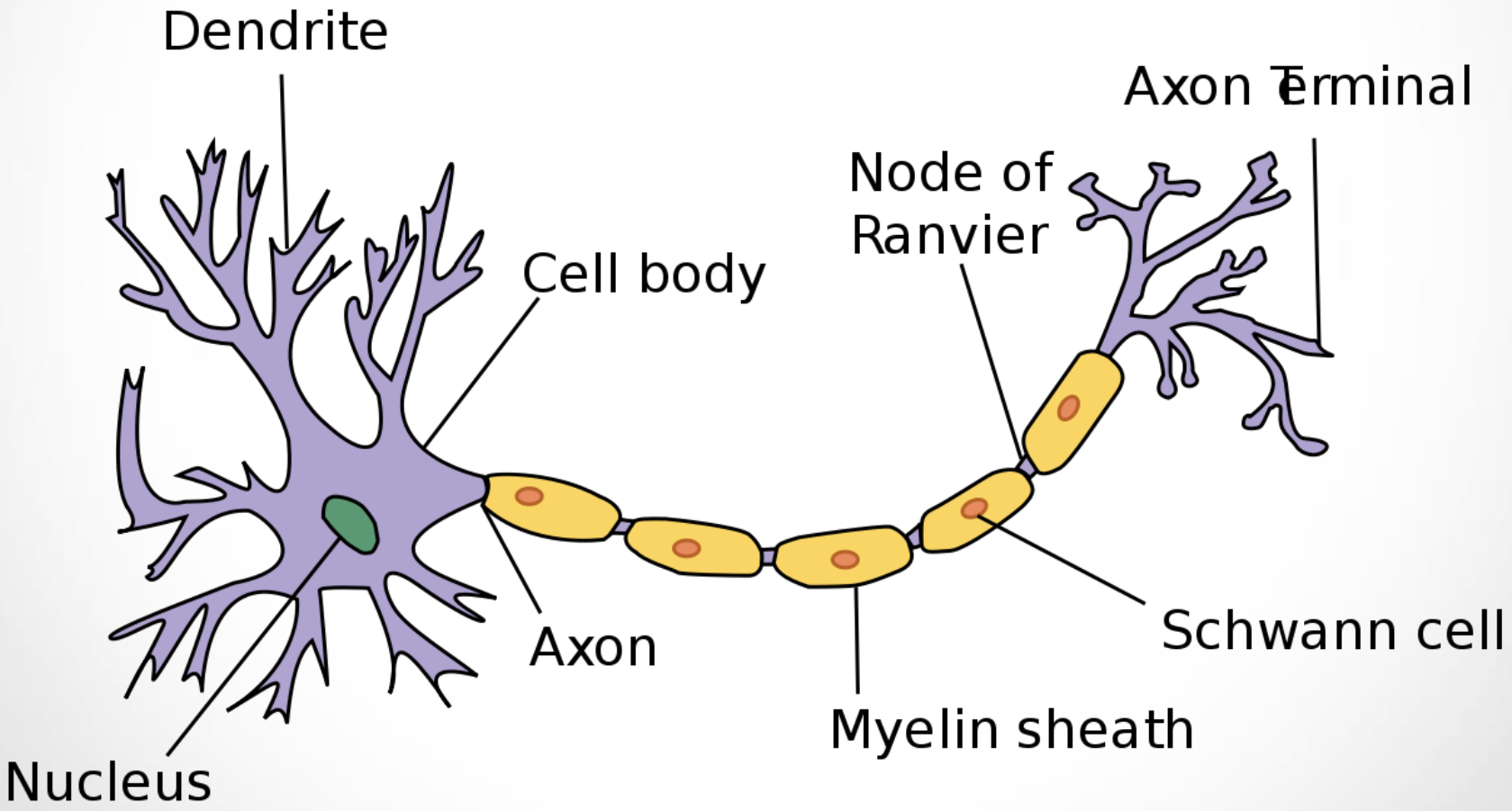


مدل سازی ...

مغز انسان ...

مغز انسان ...

سلول عصبی :


Dendrite
Axon Terminal
Node of Ranvier
Cell body
Schwann cell
Axon
Myelin sheath
Nucleus

دستگاه عصبی انسان :

مغز انسان از که است شده تشکیل نورون $10^{11} = 100,000,000,000$ هریگ از این نورون ها 10^4 اتصال با دیگر نورون ها دارد.

مجموع طول رشته های عصبی نزدیک به ۷۵ هزار کیلومتر می باشد.

سرعت سوئیچنگ نرونها در حدود 10^{-3} ثانیه است، که در مقایسه با کامپیوترها که 10^{-10} ثانیه هستند، بسیار کند به نظر می نماید.

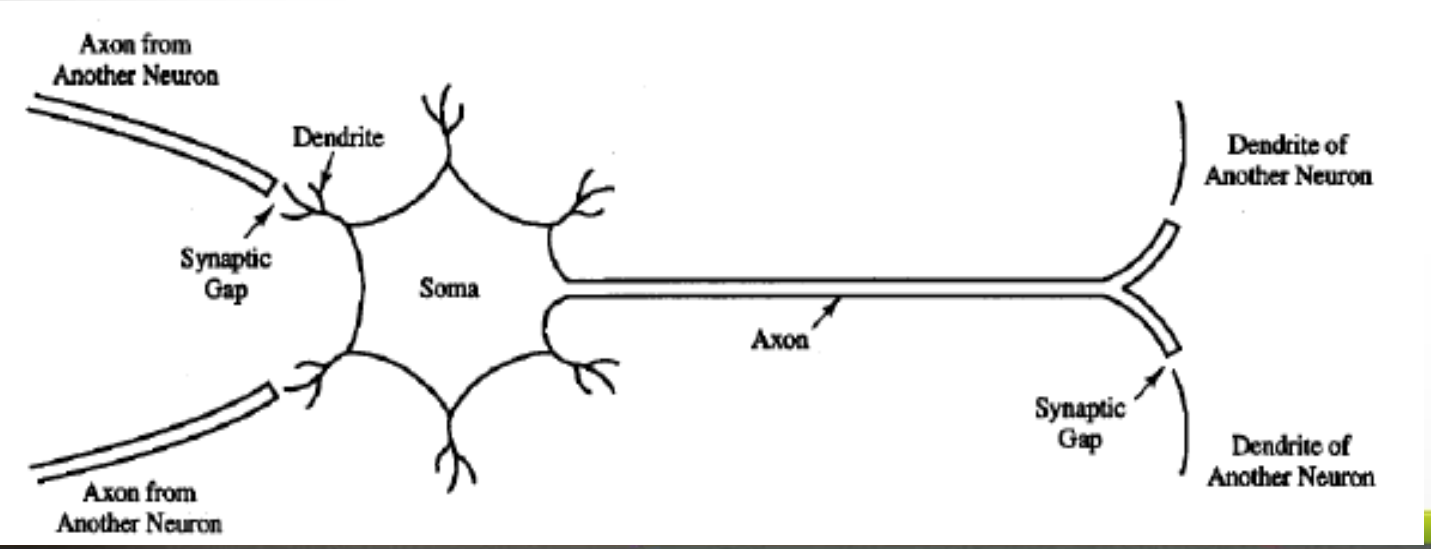
با این وجود چگونه یک انسان در کسری از ثانیه، تصویر یک انسان را بازشناسی نماید و یا مجموعه از خاطرات و تفکرات را انجام دهد ؟

در حالی که همین پرادزشات در سیستم های رایانه ای نیازمند صرف مدت زمانی بسیار بیشتر است؟

سلول عصبی :

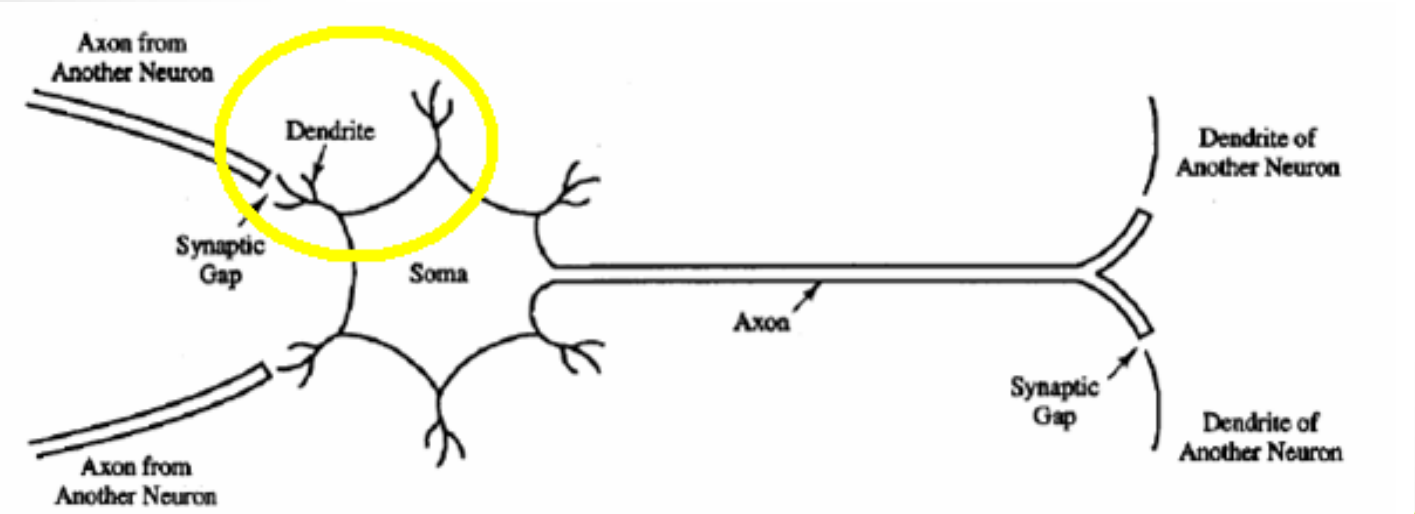
به طور کلی یک نورون از سه بخش اصلی تشکیل شده است.

- **دندریت ها،**
- **بدنه سلولی**
- **آکسون**



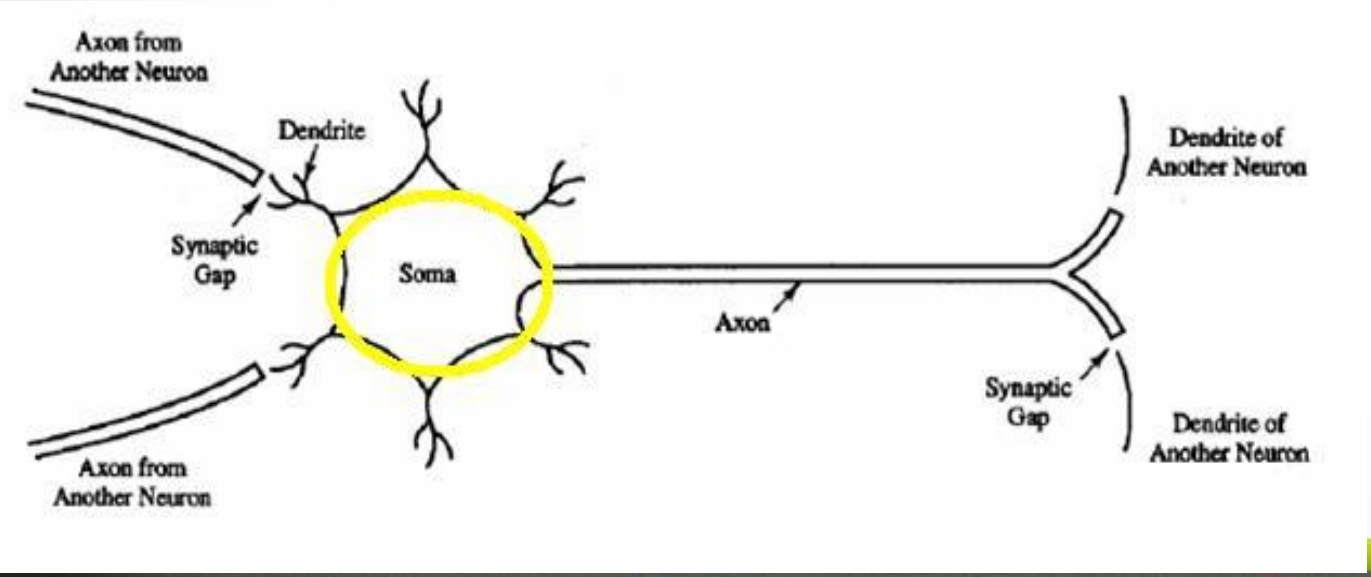
سلول عصبی :

دندریت ها : به عنوان مناطق دریافت سیگنال های الکتریکی هستند که دارای سطح نامنظم و شاخه های انشعابی بی شمار می باشند.



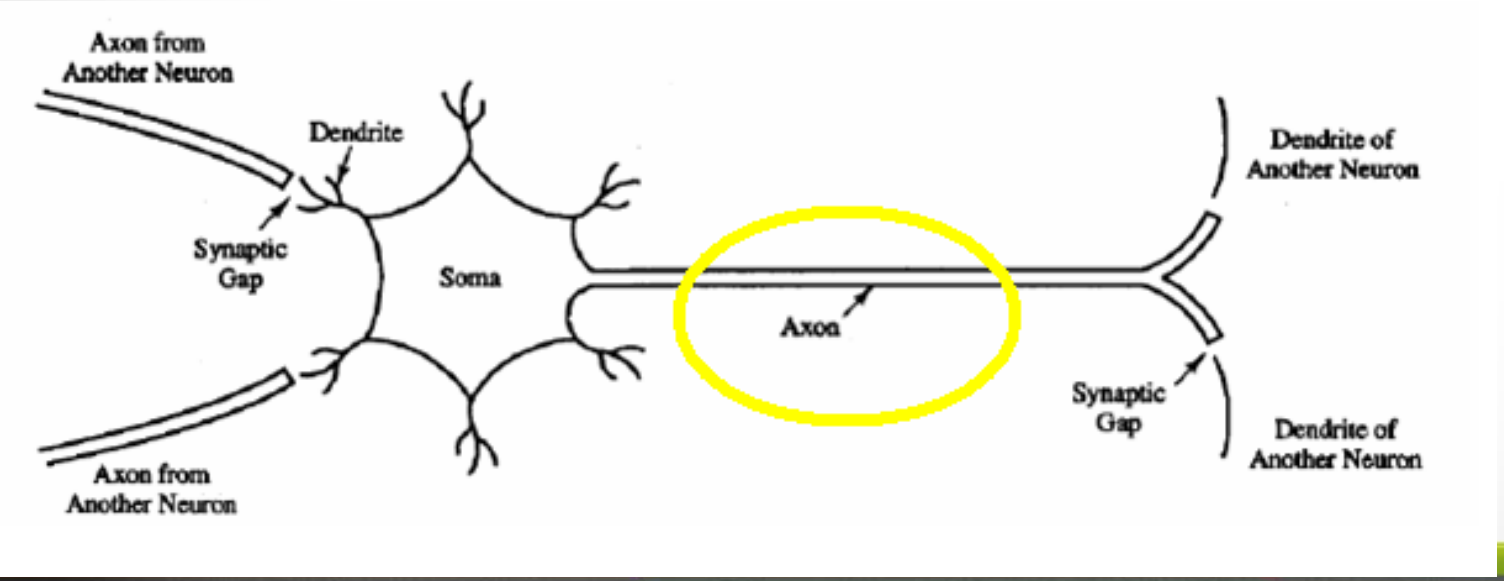
سلول عصبی :

بدنه سلولی: وظیفه تامین انرژی مورد نیاز جهت فعالیت های نرون را به عهده دارد.



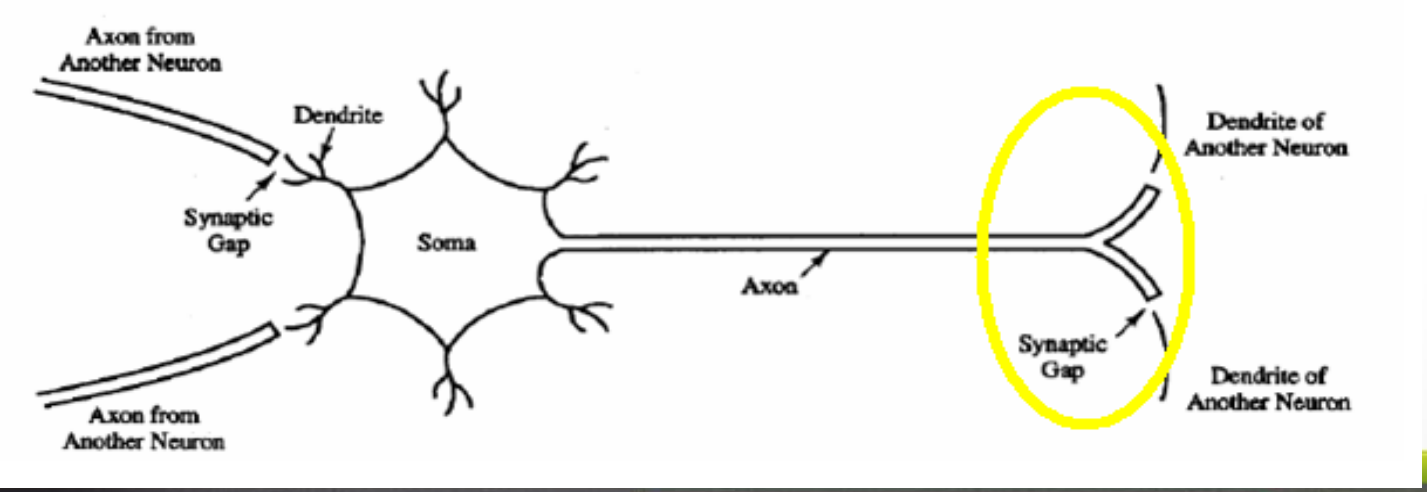
سلول عصبی :

آکسون : اکسون برخلاف دندریت ها از سطحی هموارتر و تعداد شاخه های کمتری برخوردار می باشد. اکسون سیگنال های الکتروشیمیایی دریافتی از هسته سلول را به نرون های دیگر منتقل می کند.



سیناپس : محل تلاقی یک اکسون از یک سلول به دندریت های سلول های دیگر را سیناپس می گویند.

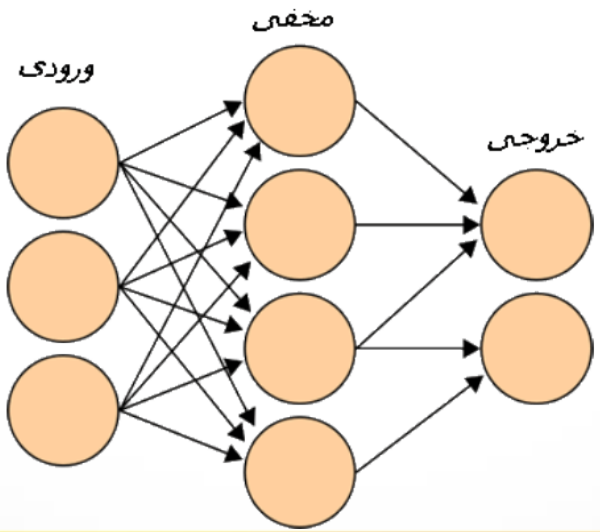
توسط سیناپس ها ارتباطات ما بین نرون ها برقرار می شود. به فضای مابین اکسون و دندریت ها فضای سیناپسی گویند.



شبکه عصبی مصنوعی :

ایده ی اصلی این گونه شبکه‌ها تا حدودی الهام‌گرفته از شیوه ی کارکرد سیستم عصبی زیستی برای پردازش داده‌ها و اطلاعات به منظور یادگیری و ایجاد دانش قرار دارد.

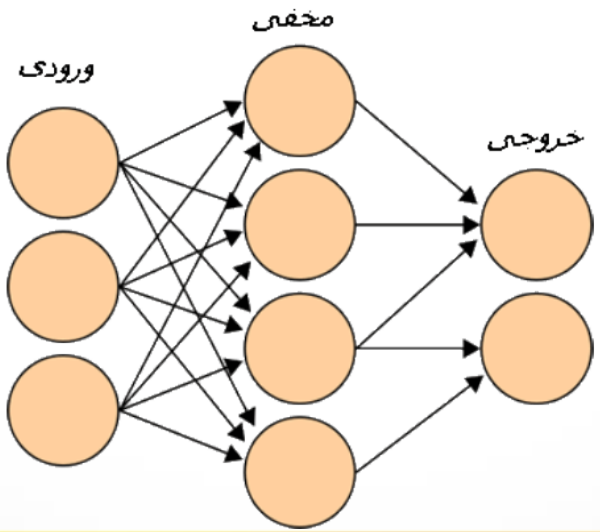
عنصر کلیدی این ایده، ایجاد ساختارهایی جدید برای سامانه ی پردازش اطلاعات است.



شبکه عصبی مصنوعی :

شبکه عصبی روشی برای محاسبه است که بر پایه اتصال به هم پیوسته چندین واحد پردازشی ساخته میشود.

شبکه عصبی از تعداد دلخواهی سلول یا گره یا واحد یا نرون تشکیل می شود که مجموعه ورودی را به خروجی ربط می دهند.

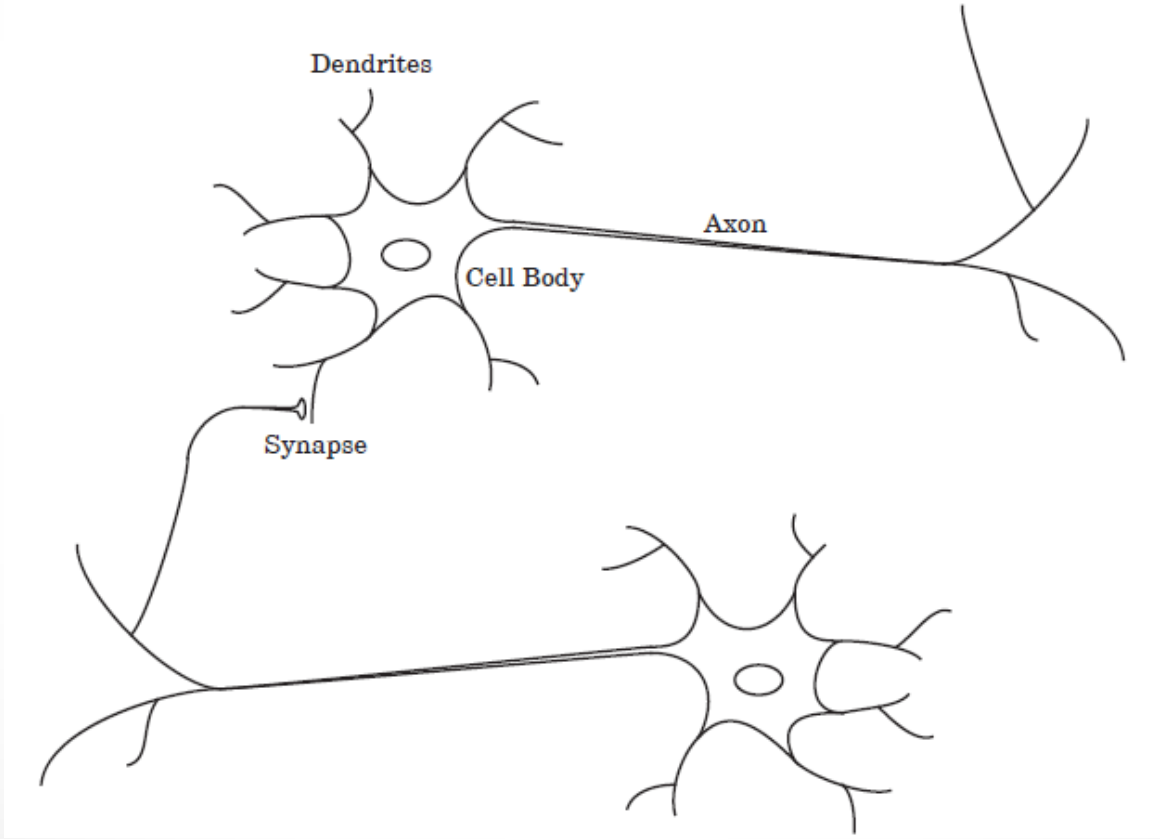


توانایی این روش :

- ✓ محاسبه یک تابع معلوم
- ✓ تقریب یک تابع ناشناخته
- ✓ شناسائی الگو
- ✓ پردازش سیگنال
- ✓ یادگیری
- ✓ و ...

توانایی این روش :

- ✓ اختصار سخن
- ✓ بازبینی امضا
- ✓ ارزیابی سرمایه
- ✓ پیش بینی فروشهای آینده و نیازهای محصول
- ✓ کلاسه بندی نمودارهای مشتری/بازار
- ✓ پیش بینی هوا
- ✓ پیش بینی محصول
- ✓ مدل کردن کنترل فرآیند
- ✓ تشخیص هدف
- ✓ شبیه سازی مسیر


Dendrites
Axon
Cell Body
Synapse

ویژگی شبکه عصبی :

قابلیت یادگیری و تطبیق پذیری

قابلیت یادگیری یعنی توانایی تنظیم پارامترهای شبکه عصبی. برای این منظور نمونه های اولیه را به شبکه اعمال می کنند شبکه، پارامترها را بر اساس این نمونه ها تنظیم می کند. اگر نمونه های جدید به این شبکه که به این طریق آموزش دیده، اعمال شود، خروجی مناسب را با درصد خطای کوچک می توان بدست آورد. با این ترتیب شبکه های عصبی می توانند با تغییر شرایط به صورت هوشمندانه، خود را تطبیق یا اصلاح نماید.

ویژگی شبکه عصبی :

- **قابلیت تعمیم پذیری:**

پس از آنکه نمونه های اولیه به شبکه آموزش داده شد، شبکه می تواند در مقابل ورودیهای آموزش داده نشده (ورودیهای جدید) قرار گیرد و یک خروجی مناسب تولید نماید. این خروجی بر اساس مکانیسم تعمیم، که چیزی جز فرایند درونیابی نیست به دست می آید.

ویژگی شبکه عصبی :

- **پردازش موازی:**

هنگامی که شبکه عصبی در قالب سخت افزار پیاده می شود سلولهایی که در یک تراز قرار می گیرند می توانند به طور همزمان به ورودیهای آن تراز پاسخ دهند. این ویژگی باعث افزایش سرعت پردازش می شود. در واقع در چنین سیستمی، وظیفه کلی پردازش بین پردازنده های کوچکتر مستقل از یکدیگر توزیع می گردد.

ویژگی شبکه عصبی :

مقاوم بودن:

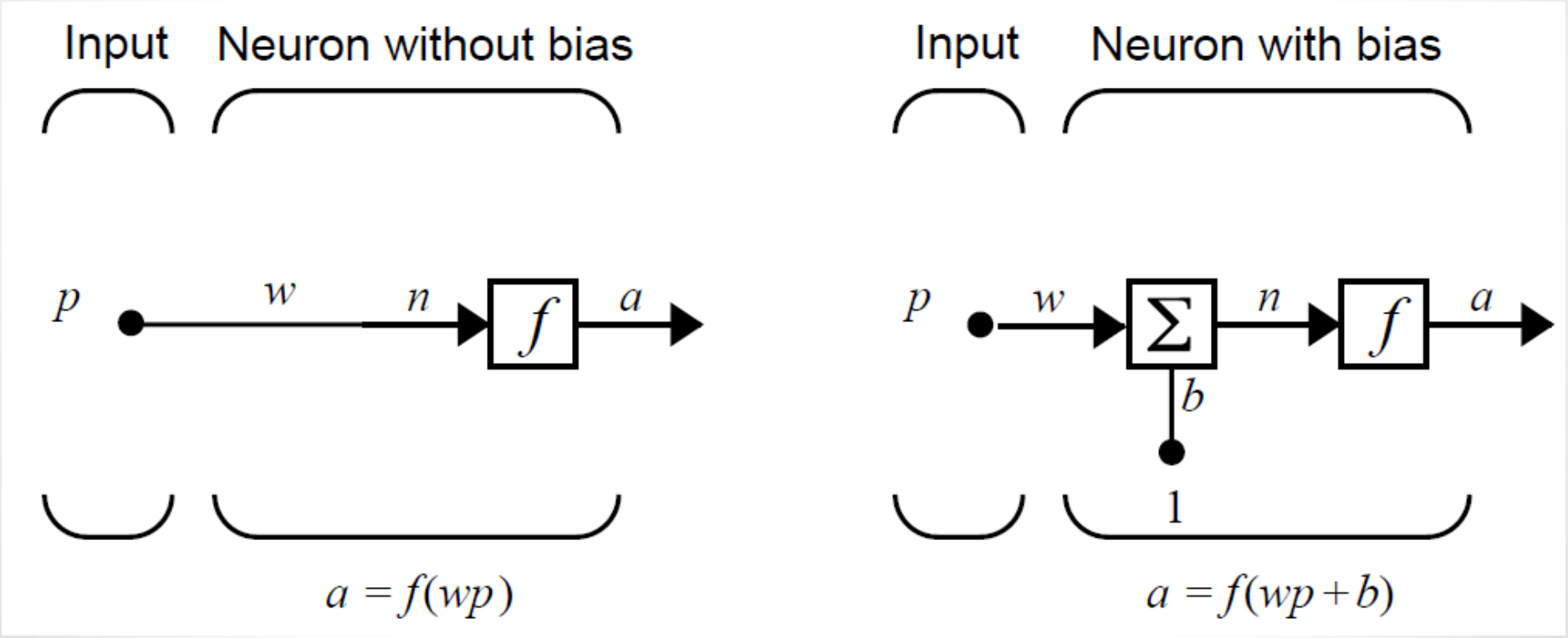
در یک شبکه عصبی رفتار کلی آن مستقل از رفتار هر سلول در شبکه می باشد درواقع رفتار کلی شبکه برآیند رفتارهای محلی تک تک سلولهای شبکه می باشد که این امر باعث می شود تا خطا های محلی سلولها از چشم خروجی نهایی دور بمانند. این خصوصیت باعث افزایش قابلیت مقاوم بودن در شبکه عصبی می گردد.

ویژگی شبکه عصبی :

- **قابلیت تقریب عمومی:**

شبکه های عصبی چند لایه، با یک یا چند لایه مخفی به شرط آن که تعداد نرونهای لایه ها مخفی کافی داشته باشند، می توانند هر تابع غیر خطی پیوسته ای را در فضای ترکیبی تخمین بزنند.

مدل ریاضی نرون :



مدل ریاضی نرون تک ورودی :

Neuron Model

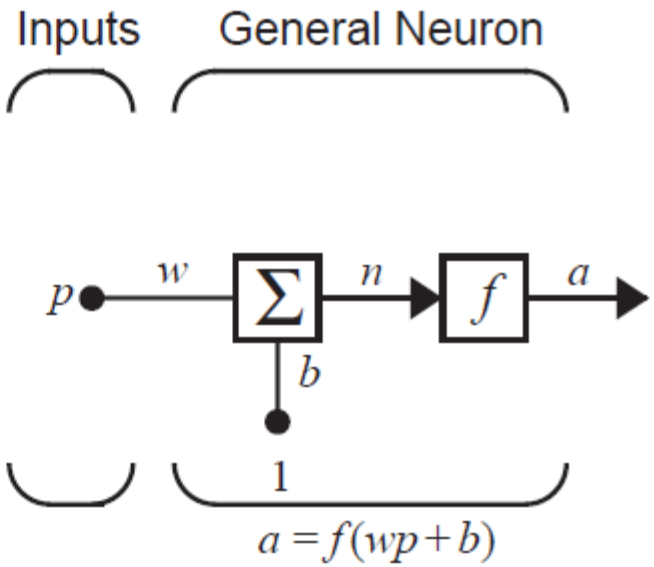


Figure 2.1 Single-Input Neuron

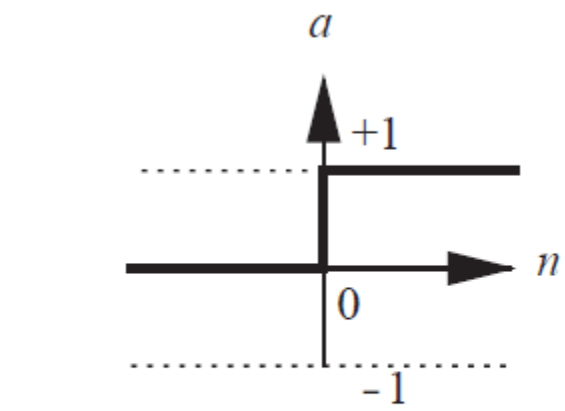
The neuron output is calculated as

$$a = f(wp + b).$$

If, for instance, $w = 3$, $p = 2$ and $b = -1.5$, then

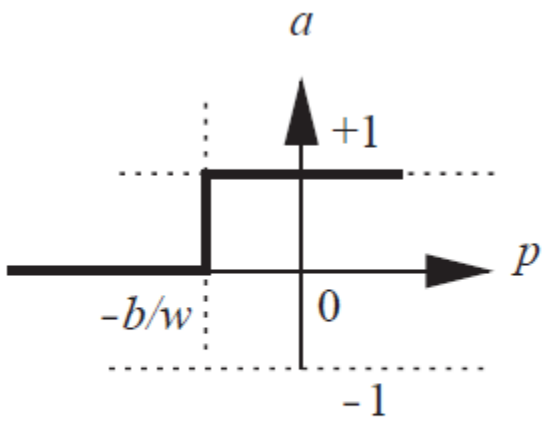
$$a = f(3(2) - 1.5) = f(4.5)$$

تابع انتقال (تابع انگیزش) :



$a = hardlim(n)$

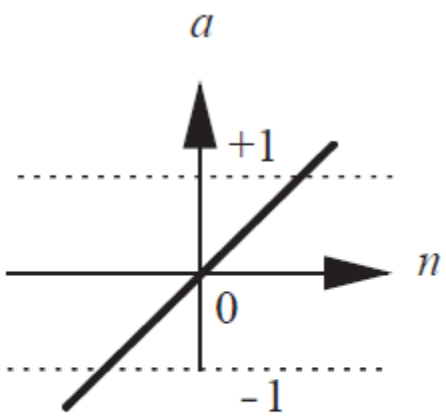
Hard Limit Transfer Function



$a = hardlim(wp + b)$

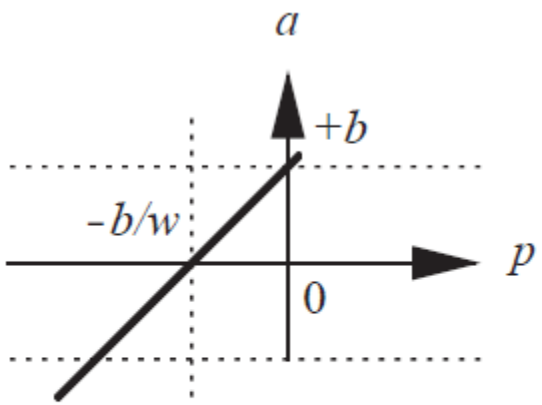
Single-Input *hardlim* Neuron

تابع انتقال (تابع انگیزش) :



$a = purelin(n)$

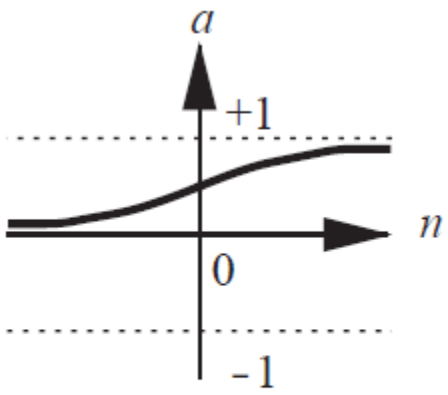
Linear Transfer Function



$a = purelin(wp + b)$

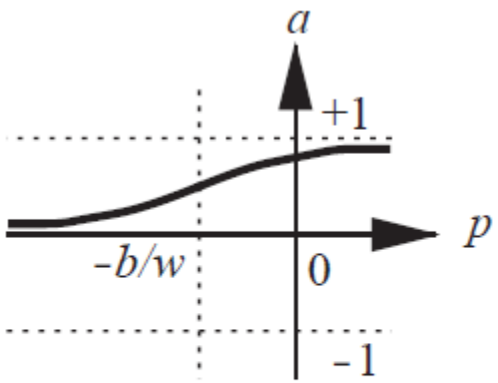
Single-Input *purelin* Neuron

تابع انتقال (تابع انگیزش) :



$a = logsig(n)$

Log-Sigmoid Transfer Function



$a = logsig(wp + b)$

Single-Input *logsig* Neuron

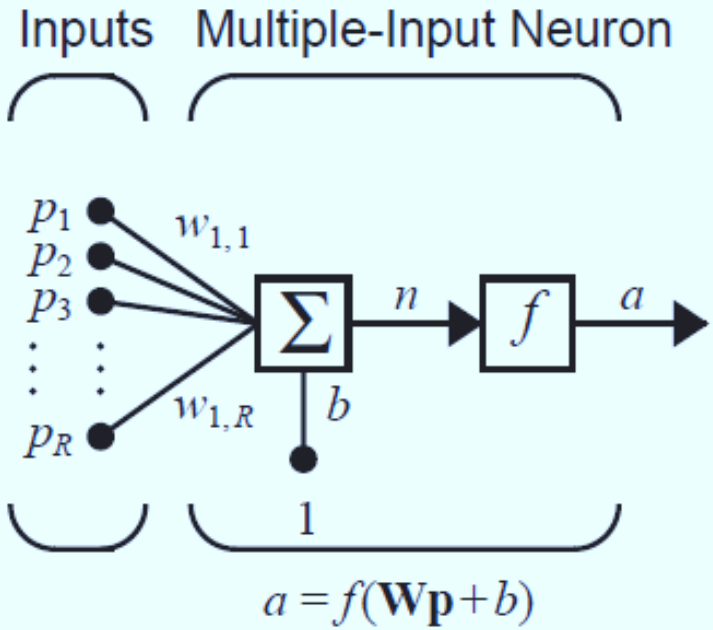
تابع انتقال (تابع انگیزش) :

Name	Input/Output Relation	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
Linear	$a = n$		purelin
Saturating Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n \leq 1$ $a = 1 \quad n > 1$		satlin

تابع انتقال (تابع انگیزش) :

Symmetric Saturating Linear	$a = -1 \quad n < -1$ $a = n \quad -1 \le n \le 1$ $a = 1 \quad n > 1$		satlins
Log-Sigmoid	$a = \frac{1}{1+e^{-n}}$		logsig
Hyperbolic Tangent Sigmoid	$a = \frac{e^{n}-e^{-n}}{e^{n}+e^{-n}}$		tansig
Positive Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \le n$		poslin
Competitive	$a = 1$ neuron with max n $a = 0$ all other neurons		compet

مدل سلول چند ورودی :



- آرایش چند نرون به صورت موازی یک لایه را تشکیل می‌دهد.
- یک لایه شامل ماتریس وزن، جمع کننده‌ها، بردار بایاس، توابع انتقال و بردار خروجی می‌باشد.

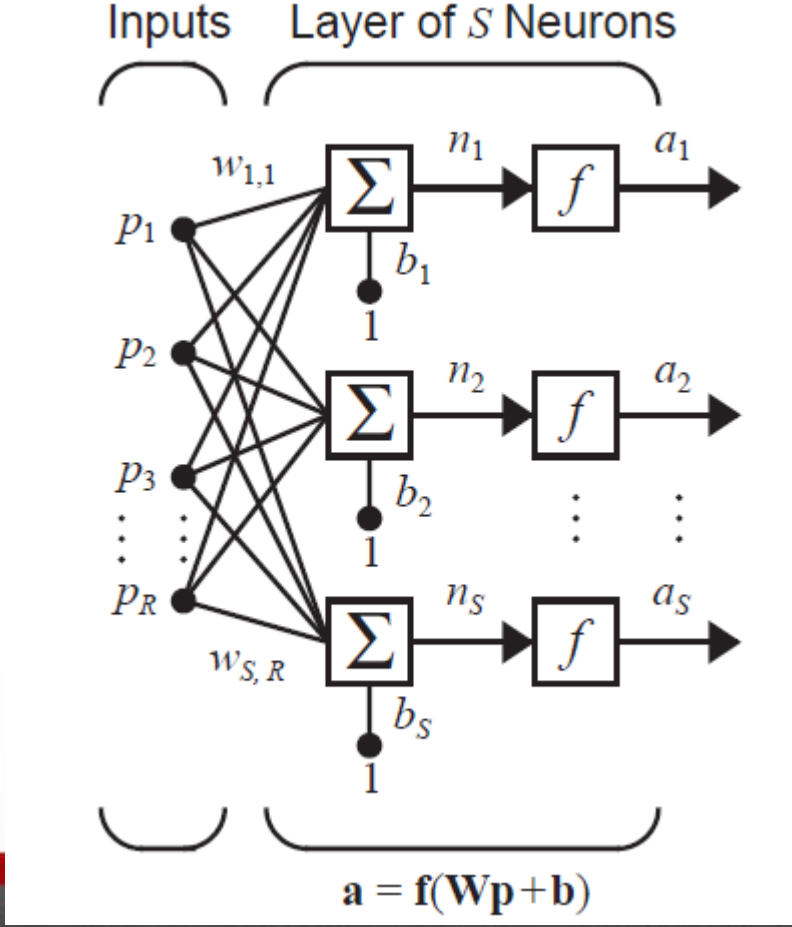
مدل سلول چند ورودی :

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

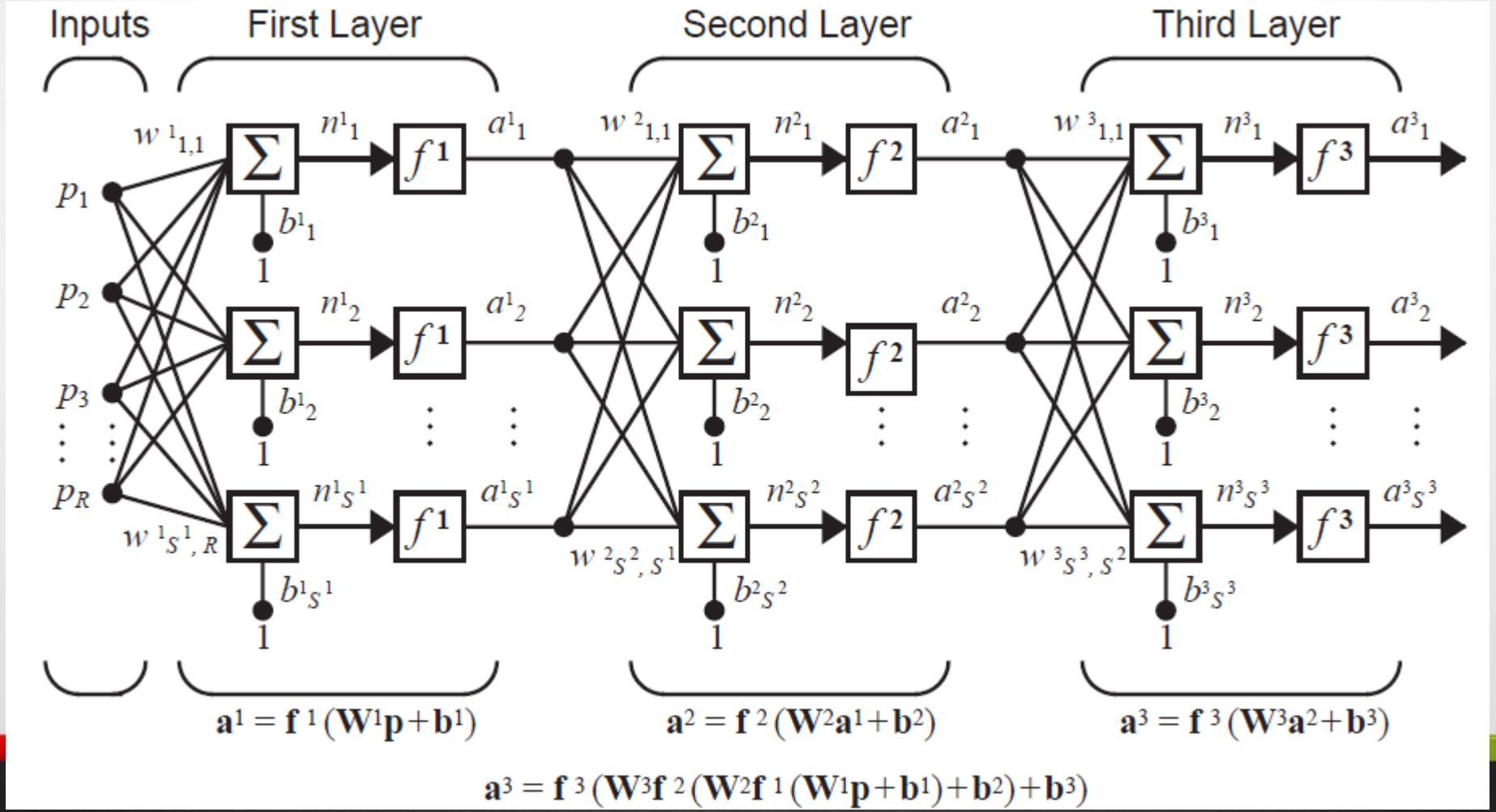
- در ماتریس وزن هر سطر معرف یک نرون و هر ستون معرف یک عضو از ورودی می‌باشد.
- بنابراین اندیس اول هر درایه از ماتریس وزن، شماره نرون و اندیس دوم شماره عضو ورودی است که به هم متصل شده‌اند.

- هر نرون دارای یک بایاس b_i، یک جمع کننده، یک تابع انتقال f و یک خروجی a_i می‌باشد.
- از کنار هم قرار دادن خروجی همه نرونها، بردار خروجی $\boldsymbol{a}$ بدست می‌آید.

مدل سلول چند ورودی چند خروجی :



مدل چند لایه ای :



$$\mathbf{a}^3 = \mathbf{f}^3(\mathbf{W}^3\mathbf{f}^2(\mathbf{W}^2\mathbf{f}^1(\mathbf{W}^1\mathbf{p}+\mathbf{b}^1)+\mathbf{b}^2)+\mathbf{b}^3)$$

مدل چند لایه ای :

- لزومی ندارد که تعداد نرونهای لایه‌های مختلف با هم مساوی باشند.
- خروجی هر لایه، ورودی لایه بعد خواهد بود و لایه‌ای که خروجی آن، همان خروجی شبکه است، لایه خروجی (Output Layer) نامیده می‌شود.
- سایر لایه‌ها را لایه‌های مخفی (Hidden Layers) گویند.
- شبکه‌های چندلایه بسیار قدرتمندتر از شبکه‌های تک لایه هستند.
- به عنوان مثال یک شبکه دو لایه که در لایه اول تابع انتقال سیگموئید و در لایه دوم تابع انتقال خطی داشته باشد، به خوبی قادر به تخمین اغلب توابع می‌باشد در حالی که شبکه‌های تک لایه نمی‌توانند این کار را انجام دهند.

توپولوژی شبکه عصبی :

- به مجموعه‌ای از لایه‌های مختلف و تعداد نرونهای موجود در هر لایه، توپولوژی شبکه گفته می‌شود.
- برای حل مسائل مختلف اولین گام تعیین توپولوژی شبکه عصبی می‌باشد.
- توپولوژی بهینه یک شبکه عصبی با سعی و خطا به دست می‌آید.
- معمولاً در مسائل مختلفی که هدف تعیین یک نگاشت غیرخطی است، شبکه‌های عصبی دارای یک یا دو لایه پنهان که تعداد نرونهای هر لایه درحدود مجموع تعداد درایه‌های بردار ورودی و تعداد نرونهای لایه خروجی است، بهترین جواب را می‌دهند.

یادگیری :

- 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.

یادگیری :

۱-یادگیری با سرپرستی (Supervised Learning)

- در این روش به منظور اصلاح وزنها باید الگوهای ورودی و خروجی مطلوب متناظر آنها به عنوان زوج آموزشی (Training Pair) در اختیار شبکه قرار گیرد.
- سپس به شبکه اجازه داده شود که با اعمال بردار ورودی به لایه اول آن بر روی اطلاعات موجود پردازش لازم را به عمل آورده، خروجی خود را عرضه نماید.
- خروجی شبکه با خروجی مطلوب مقایسه شده و سپس با محاسبه خطای یادگیری، از آن جهت تنظیم پارامترهای شبکه استفاده شود.
- در این نوع یادگیری، مسئله آموزش شبکه عصبی منجر به یک مسئله بهینه‌سازی به منظور کمینه‌سازی معیاری از خطا می گردد.
- برخی از قوانین یادگیری با سرپرستی عبارتند از: قانون آموزش هب (Hebb)، قانون ویدرو-هوف (Widrow-Hoff)، قانون گروسبرگ (Grossberg) و الگوریتم پس انتشار خطا.

یادگیری :

۳-یادگیری تقویتی (Reinforcement Learning)

- این روش یادگیری مبتنی بر این اصل می‌باشد که ضمن وجود یک معلم که رفتار مطلوب شبکه را تعیین می‌نماید، به دلیل وجود یک بیان کیفی از مطلوب یا مطلوب نبودن رفتار شبکه، روش بدون سرپرستی تلقی می‌شود.
- بنابراین حالتی بین دو یادگیری با سرپرستی و بدون سرپرستی را در ذهن مجسم می‌نماید.
- یادگیری تقویتی به صورتی انجام می‌پذیرد که یک شاخص اجرایی موسوم به سیگنال تقویتی ماکزیمم شود.
- این بدین معنی است که اگر شبکه پارامترهایش را به گونه‌ای تغییر داد که منجر به یک حالت مساعد شد، آنگاه تمایل سیستم جهت تولید آن عمل خاص تقویت می‌شود، در غیر این صورت تمایل شبکه جهت تولید آن عمل خاص تضعیف می گردد.
- این نوع یادگیری بیشتر برای سیستم‌های کنترلی کاربرد دارد.

یادگیری :

۲- یادگیری بدون سرپرستی (Unsupervised Learning)

- در یادگیری بدون سرپرستی، پارامترهای شبکه عصبی تنها توسط پاسخ سیستم اصلاح و تنظیم می‌شوند.
- به عبارتی تنها اطلاعات دریافتی از محیط به شبکه را بردارهای ورودی تشکیل می‌دهند و بردار خروجی مطلوب به شبکه اعمال نمی‌شود.
- به عبارت دیگر به شبکه هیچ نمونه‌ای از تابعی که قرار است بیاموزد، داده نمی‌شود.
- همچنین سیگنال تقویتی نیز از عملکرد شبکه حذف شده و تنها به روش تخصیص پاداش یا جزاء اکتفا می گردد.
- در واقع طی یک فرآیند رقابتی شبکه در حال یادگیری به سمتی حرکت می کند که طی آن هر نرون سعی در رقابت با نرونهای دیگر در کسب امتیازات دارد.

مرور اجمالی ...

پنج عنصر وجود دارد:

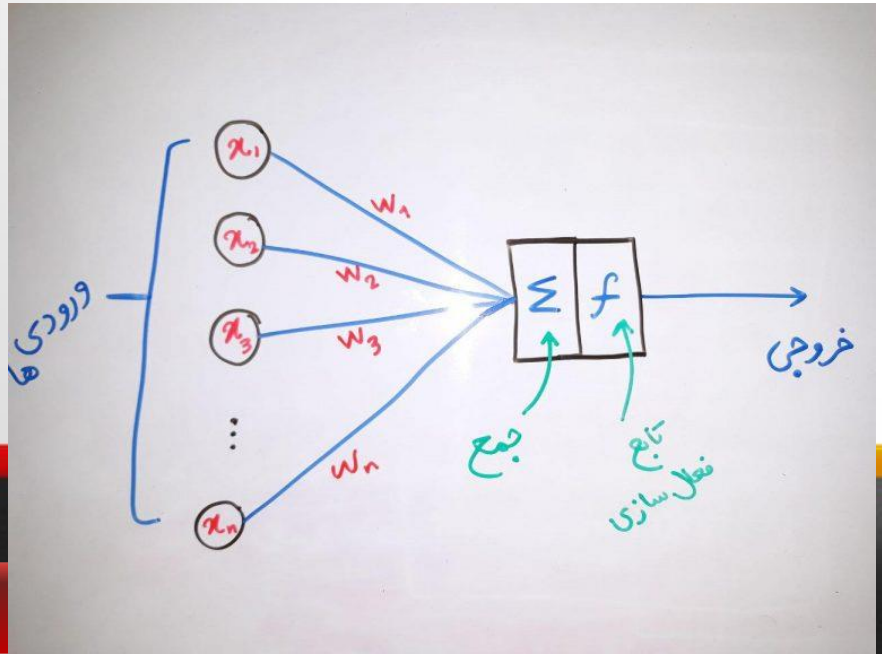
اول Xها هستند. اینها همان ورودی های ما(نرون های ورودی) هستند که از مجموعه داده ها استفاده می کنند. در واقع ورودی الگوریتم همین ها Xها هستند که در این تصویر از X1یا Xnوجود دارند.

عناصر دوم وزن ها هستند. در شبکه های عصبی هر کدام از Xها یک وزن دارد که با Wنمایش میدهیم. همان طور که مشاهده میکنید هر کدام از ورودی های ما به یک وزن متصل شده است. در واقع هر ورودی(مثلا X1یک وزن به اسم W1دارد) که باید در وزن خود ضرب شود.

عنصر سوم در شبکه عصبی تابع جمع(سیگما) است. که حاصل ضرب Xها در Wها را با هم جمع میکند.

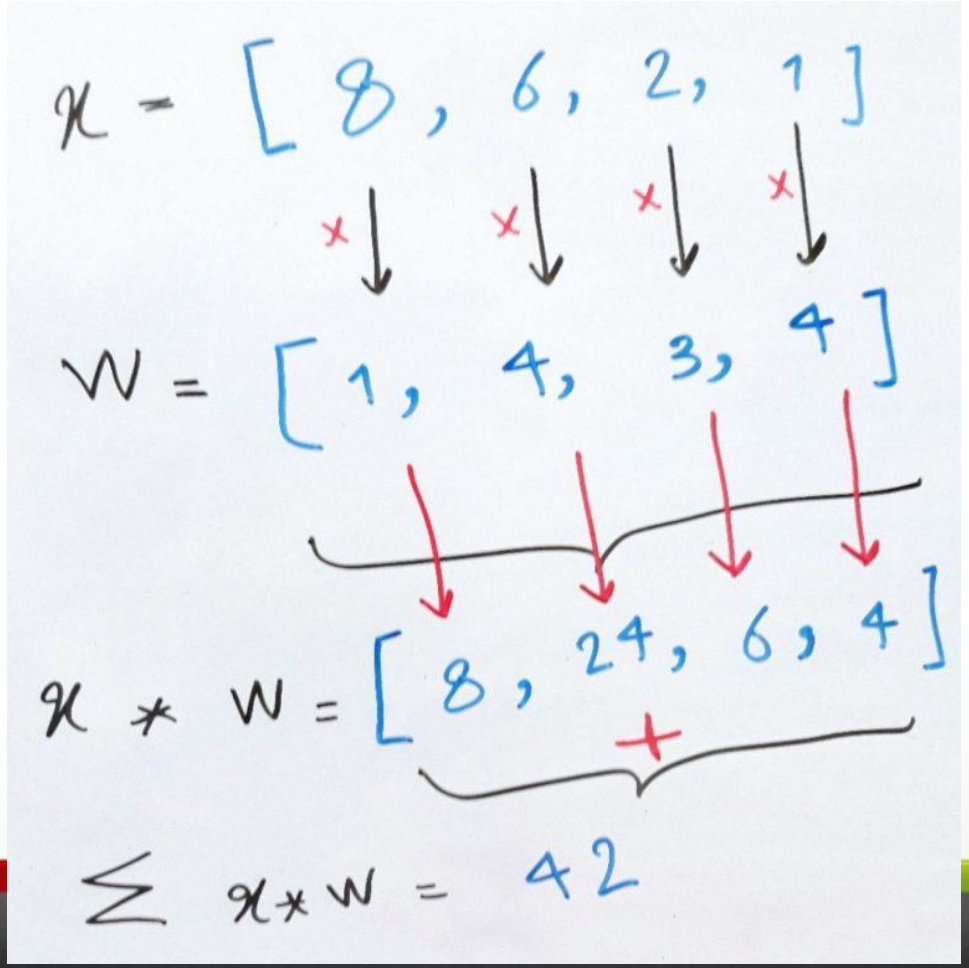
عنصر چهارم یک تابع فعال سازی است .

عنصر پنجم و آخر نیز خروجی شبکه عصبی است که در واقع نتیجه این شبکه را مشخص می کند.



مرور اجمالی ...

$$۴۲ = ۱\times۴ + ۲\times۳ + ۶\times۴ + ۸\times۱$$



مرور اجمالی ...

فرض کنید به یک بچه کوچک میخواهید آموزش بدهید. مثلا میخواهید به اون بگویید که یک اتوبوس چیست. برای این کار چه کاری میکنید؟

یک راه ساده(که خود ما هم خیلی چیزها را به وسیله آن یادگرفته ایم) این است که به بچه کوچک بگوییم که این اتوبوس است و آن اتوبوس نیست. چندین مورد اتوبوس را به بچه نشان میدهیم و چندین موردی که اتوبوس نیست را هم به اون میگوییم. به راحتی یک بچه با چند بار تمرین میتواند یادبگیرد که اتوبوس چیست! یعنی میتواند تمایز اتوبوس را با بقیه ماشین های دیگر درک کند.

این روش پایه روش یادگیری در پرسپترون و بسیاری دیگر از الگوریتم های داده کاوی است.

مرور اجمالی ...

فرض کنید تفاوت بین اتوبوس و پراید را میخواهیم این بار به کامپیوتر یاد بدهیم. در کل نحوه یاد گرفتن کامپیوتر به این صورت است که باید یک سری ویژگی ها را به کامپیوتر داده و بگوییم این اتوبوس است.

یک سری ویژگی های دیگر را هم به کامپیوتر داده و بگوییم که این پراید است. برای یادگیری در شبکه های عصبی(و کلا بقیه الگوریتم های داده کاوی) نمونه ها مختلف یک اتوبوس و نمونه های مختلف یک پراید را همراه با ویژگی های آن ها به الگوریتم شبکه عصبی(یا دیگر الگوریتم های داده کاوی) داده و آن انتظار داریم که یادگیری را انجام دهد.

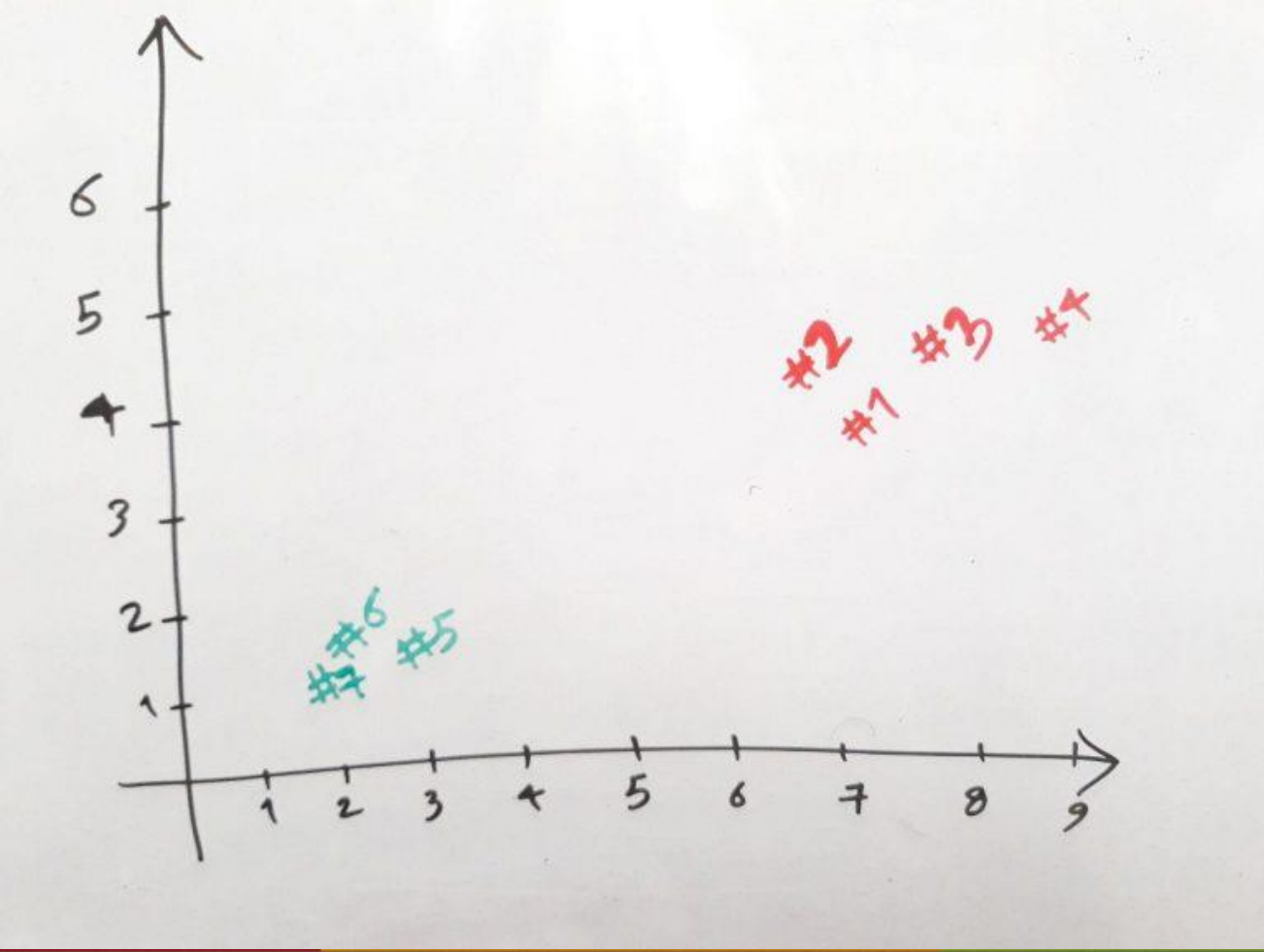
مرور اجمالی ...

#	طول	ارتفاع	نوع
1	7	4	اتوبوس
2	6,5	4,5	اتوبوس
3	7,5	4,5	اتوبوس
4	9	4,5	اتوبوس
5	3	1,5	پراید
6	2,5	1,7	پراید
7	2	1,6	پراید

مرور اجمالی ...

همان طور که میبینید نمونه های مختلف را در یک جدول آوردیم. مثلا نمونه ۱ را نگاه کنید. این **نمونه ۱ اتوبوس** است که **طول ۷متر و ارتفاع ۴متر** دارد. **نمونه ۵** نیز یک **پراید** است که **طول ۳متر و ارتفاع ۱.۵** متر دارد. به همین صورت چندین اتوبوس را برچسب گذاری کرده ایم و ویژگی های مختلف آن(طول و ارتفاع) را اندازه گیری کرده و آماده ساخته ایم. پس نکته مهم این است که در اینجا ما دو ویژگی داریم. یکی **طول** و دیگری **ارتفاع**. این دو ویژگی باعث ایجاد **دو بعد** در ریاضیات مورد استفاده ما می شود. برای واضح تر شدن بحث تصویر را نگاهی بیندازید. در این تصویر ما جدول بالا را در دو بعد که **محور افقی نشان دهنده طول** و **محور عمودی نشان دهنده ارتفاع** است قرار داده و هر کدام از نمونه ها را به صورت یک نقطه در این فضای دو بعدی درج کرده ایم:

مرور اجمالی ...

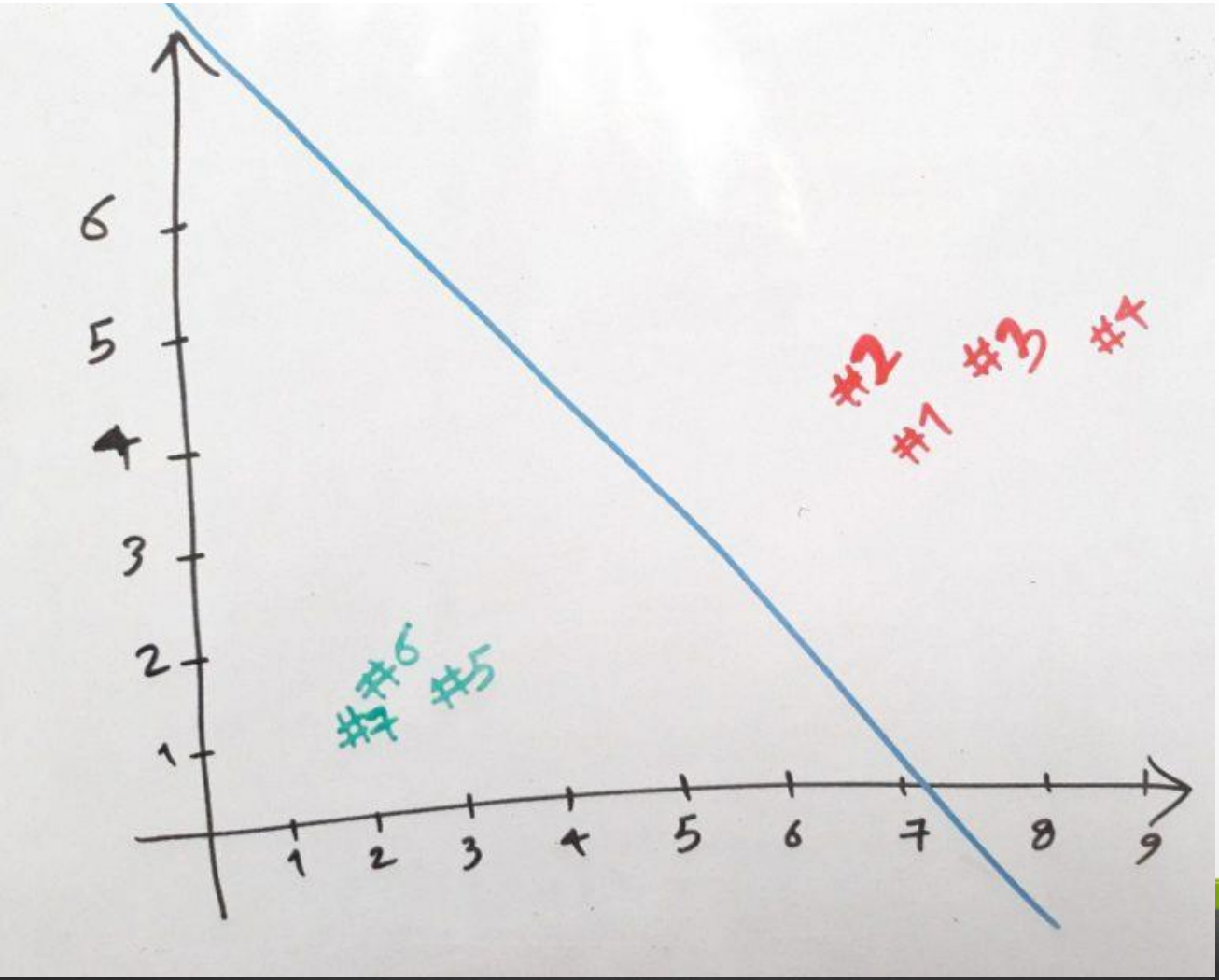


مرور اجمالی ...

تحلیل تصویر بالا ساده است. نگاهی به جدول تصویر بالاتر بیندازید. در اینجا هر کدام از نمونه ها را با توجه به طول(محور افقی) و ارتفاع(محور عمودی) بر روی محور مختصات با شماره درج کرده ایم. رنگ قرمز اتوبوس ها و رنگ سبز پراید ها هستند. مثلا نمونه #۴ نشان دهنده اتوبوس است(که شماره آن در جدول ۴ بود) که طول آن ۹ و ارتفاع آن ۴.۵ متر بود.

حال به معادله خط میرسیم. خط آبی را در تصویر زیر نگاه کنید:

مرور اجمالی ...



مرور اجمالی ...

این خط یک خط جدا کننده بین **اتوبوس و پراید** است. یعنی تمامی نقاطی که بالای این خط قرار میگیرند اتوبوس هستند و تمامی نقاطی که پایین این خط قرار میگیرند پراید هستند. در واقع کار پرسپترون در شبکه عصبی پیدا کردن این خط(یا چیزی شبیه به این خط) است به طوری که بین دو نمونه مختلف تمایز ایجاد کند.

در واقع **پرسپترون** ساده ترین تابع فعال سازی در شبکه عصبی است.

مرور اجمالی ...

میبینید که ما دو بردار داریم. بردار Xو W.

بردار X در شبکه های عصبی همان ورودی های ماست و بردار W وزن های ما(به ازای هر ورودی یک وزن داریم). وقتی از **جمع ضرب** صحبت میکنیم دقیقا مانند کاریست که در تصویر بالا انجام شده است. یعنی تک تک عناصر بردار **X**را در عنصر نظیر آن در بردار **W**ضرب کرده و حاصل آن ها را با یکدیگر جمع میکنیم. به این کار در ریاضیات **ضرب داخلی یا dot product** میگویند. حاصل این ضرب یک عدد است.

اگر این عدد بالاتر از یک مقدار آستانه Thresholdباشد خروجی پرسپترون ۱ و اگر کمتر از این مقدار باشد خروجی پرسپترون ۰ است.

اجازه بدهید این خروجی را (-b) بخوانیم:

مرور اجمالی ...

$$x = [x_1, x_2, x_3]$$

$$w = [w_1, w_2, w_3]$$

$$\sum_{i=1}^{3} x_i w_i = x_1 w_1 + x_2 w_2 + x_3 w_3$$

$x . w$ ← ضرب داخلی (dot product)

خروجی
$$\begin{cases} 0 & \text{if } w . x \leq -b \\ 1 & \text{if } w . x > -b \end{cases}$$

مرور اجمالی ...

همان طور که میبینید اگر ضرب داخلی W در X کمتر از b- شود خروجی پرسپترون ۰ و اگر بیشتر از b- شود خروجی پرسپترون ۱ می شود.

مثال پراید یا **اتوبوس** را به یاد بیاورید.

دو ویژگی داشتیم یکی **طول** و یکی دیگر **ارتفاع**.

برای یادآوری تصویر ویژگی های اتوبوس و پراید را دوباره با یکدیگر مرور کنیم:

#	طول	ارتفاع	نوع
1	7	4	اتوبوس
2	6,5	4,5	اتوبوس
3	7,5	4,5	اتوبوس
4	9	4,5	اتوبوس
5	3	1,5	پراید
6	2,5	1,7	پراید
7	2	1,6	پراید

مرور اجمالی ...

ورودی داریم(طول و ارتفاع) که همان دو ویژگی های ما هستند.

پس **۲ وزن** هم داریم، یکی برای **طول** و یکی برای **ارتفاع**.

وزن طول را ۱ و وزن ارتفاع را ۲ در نظر بگیریم.

حال برای هر نمونه(ماشین) عدد ویژگی طول را در ۱(وزن طول) و عدد ویژگی ارتفاع را در ۲(وزن ارتفاع) ضرب کنیم و حاصل ضرب ها را با هم جمع میکنیم(همان ضرب داخلی).

نتیجه چیزی شبیه به زیر برای هر نمونه تصویر بالا می شود:

مرور اجمالی ...

#1	7	x 1	+	4 x 2	=	15
#2	6,5	x 1	+	4,5 x 2	=	15,5
#3	7,5	x 1	+	4,5 x 2	=	16,5
#4	9	x 1	+	4,5 x 2	=	18
#5	3	x 1	+	1,5 x 2	=	6
#6	2,5	x 1	+	1,7 x 2	=	5,9
#7	2	x 1	+	1,6 x 2	=	5,2

مرور اجمالی ...

میبینید که با انتخاب وزن طول برابر ۱ و وزن ارتفاع برابر ۲ اعداد تولید شده برای **اتوبوس** و **پراید** از یک دیگر تفکیک شدند.

حال اگر عدد b– همان حد آستانه را برابر ۹ در نظر بگیریم، پرسپترون ضرب داخلی های بالای این عدد را **اتوبوس** و پایین این عدد را **پراید** در نظر میگیرد.

به این دست از مسئله ها **تفیک پذیر خطی (Linear Separability)** می گویند.

پروژه :

قوانین یادگیری شبکه های عصبی و خواص آنها

برای مثال :

- **قانون یادگیری هب**
- **قانون یادگیری پرسپترون**
- **قانون یادگیری هاپفیلد**
- **قانون یادگیری دلتا**
- **قانون یادگیری ویدروهاف**
- **قانون یادگیری کوهنن**
- **قانون یادگیری out star**
- **قانون یادگیری بولتززمان**
- **و ...**

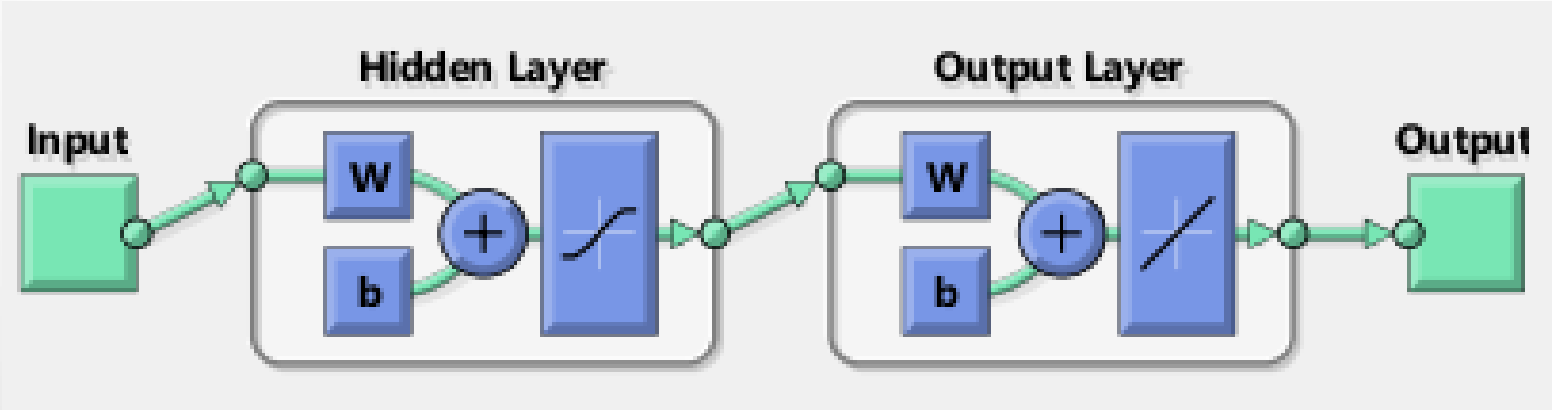
متلب ...

شبکه عصبی مانند سایر الگوریتم های محاسباتی، می توان با کمک متلب مورد استفاده قرار داد.

نرم افزار متلب، توابع و تولباکس های مختلفی برای شبکه عصبی موجود دارد. هر چند در صورت لزوم می توان به صورت مستقیم همه محاسبات لازم در شبکه عصبی را به صورت دلخواه ساخته و از توابع پیشفرض استفاده ای ننمود.

متلب ...

تا کنون به صورت مختصر با مفاهیم بنیادی و کلیدی شبکه عصبی آشنا شدیم.
در ادامه به صورت اجمالی نحوه استفاده از شبکه عصبی و هم چنین تولباکس
این تکنیک را در نرم افزار متلب مرور خواهیم نمود.



قوانین یادگیری :

قبلا هم در این باره توضیح داده شد که برای اینکه وزن سلول ها آپدیت شود لازم است تا قاعده‌ای خاص پیروی نماید. با این قاعده اصطلاحا قانون یادگیری می گویند.

Functions

- learnh - *Hebb weight learning rule*
- learnk - *Kohonen weight learning function*
- learnp - *Perceptron weight and bias learning function*
- learnis - *Instar weight learning function*
- learnos - *Outstar weight learning function*
- learngd - *Gradient descent weight and bias learning function*

قانون یادگیری هب :

learnh

Hebb weight learning rule

```
>> p = [1 2 3]';
>> a = [10 20]';
>> dW = learnh([],p,[],[],a,[],[],[],[],[],lp,[])

dW =

     5     10     15
    10     20     30
```

قانون دلتا :

learnwh

Widrow-Hoff weight/bias learning function

```
>> p = [1 2 3]';
>> e = [10 20]';
>> dW = learnwh([],p,[],[],[],[],e,[],[],[],lp,[])

dW =

     5     10     15
    10     20     30
```

الگوریتم های یادگیری :

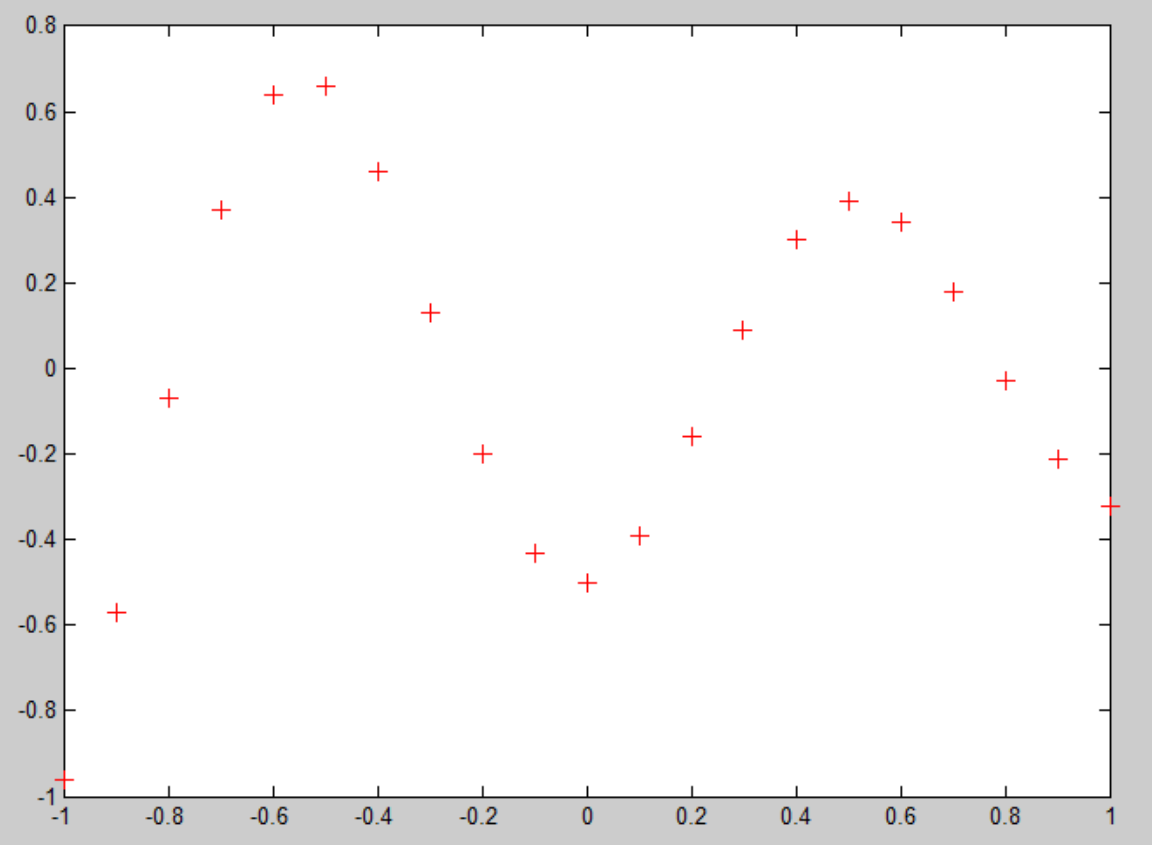
حال به صورت تجمیعی برای اینکه بتوان عملیات یادگیری را انجام داد نیازمند الگوریتم خاص و منسجمی می باشد. برای این امر روش های مختلفی ابداع شده است.

یک الگوریتم می توانند بارها و بارها تکرار شود تا سرانجام به جواب مورد نظر برسد.

یادآوری می گردد می توان با استفاده از دستوارت آماده متلب این عملیات را انجام داد.

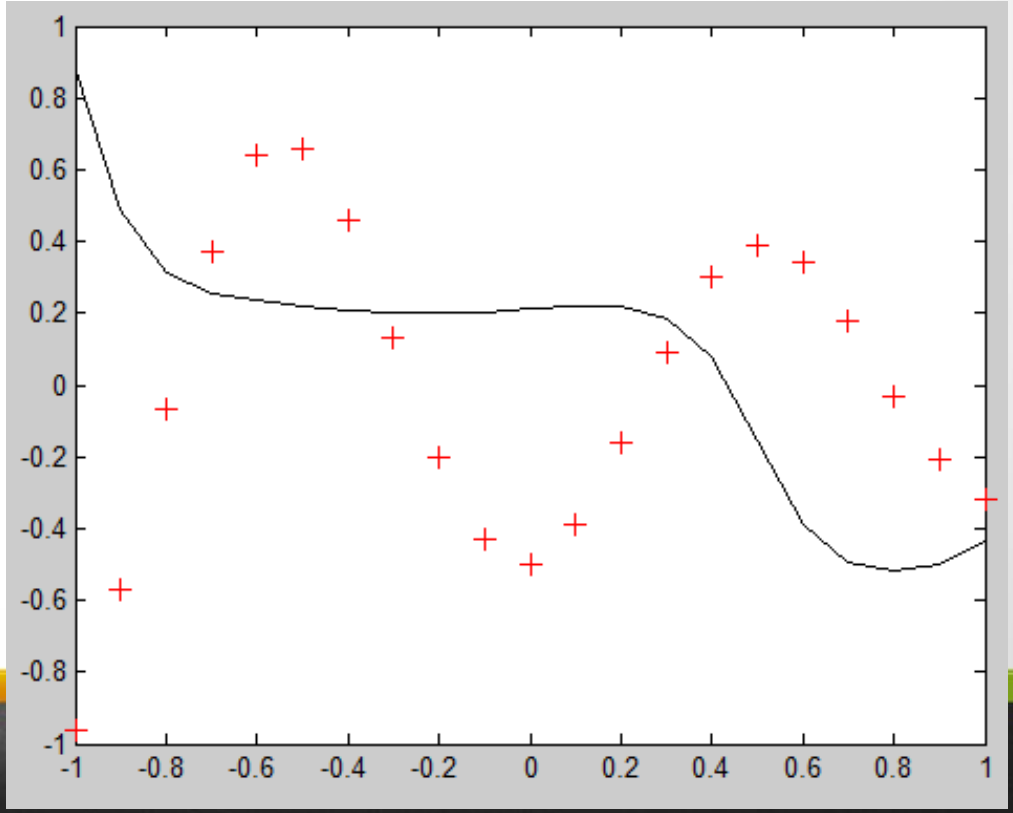
مثال :

می خواهیم یک شبکه دولایه را طوری طراحی کنیم که تابع نشان داده شده در شکل زیر را تقریب بزند.

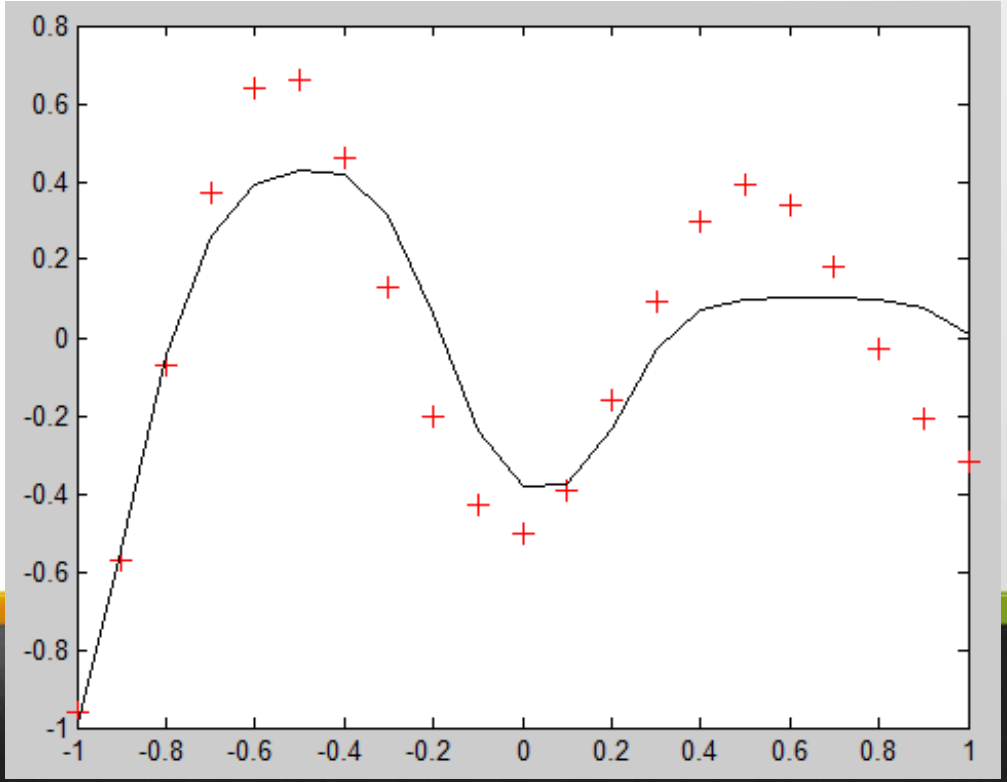


```
clear all;close all;warning off;clc
p=-1:0.1:1;
t=[-0.96 -0.57 -0.07 +0.37 +0.64 0.66 0.46 0.13 ...
   -0.20 -0.43 -0.50 -0.39 -0.16 0.09 0.30 0.39 ...
    0.34  0.18 -0.03 -0.21 -0.32];
net=newff(minmax(p),[5,1],{'tansig','purelin'},'trainlm');
net.trainParam.show=10;
net.trainParam.lr=0.001;
net.trainParam.epoch=500;
net.trainParam.goal=0.0005;
net=train(net,p,t);
y0=sim(net,p);
figure;
plot(p,t,'+r','markersize',8)
hold on
plot(p,y0,'k')
```

```
net.trainParam.show=10;
net.trainParam.lr=0.01;
net.trainParam.epoch=5;
net.trainParam.goal=0.5;
```

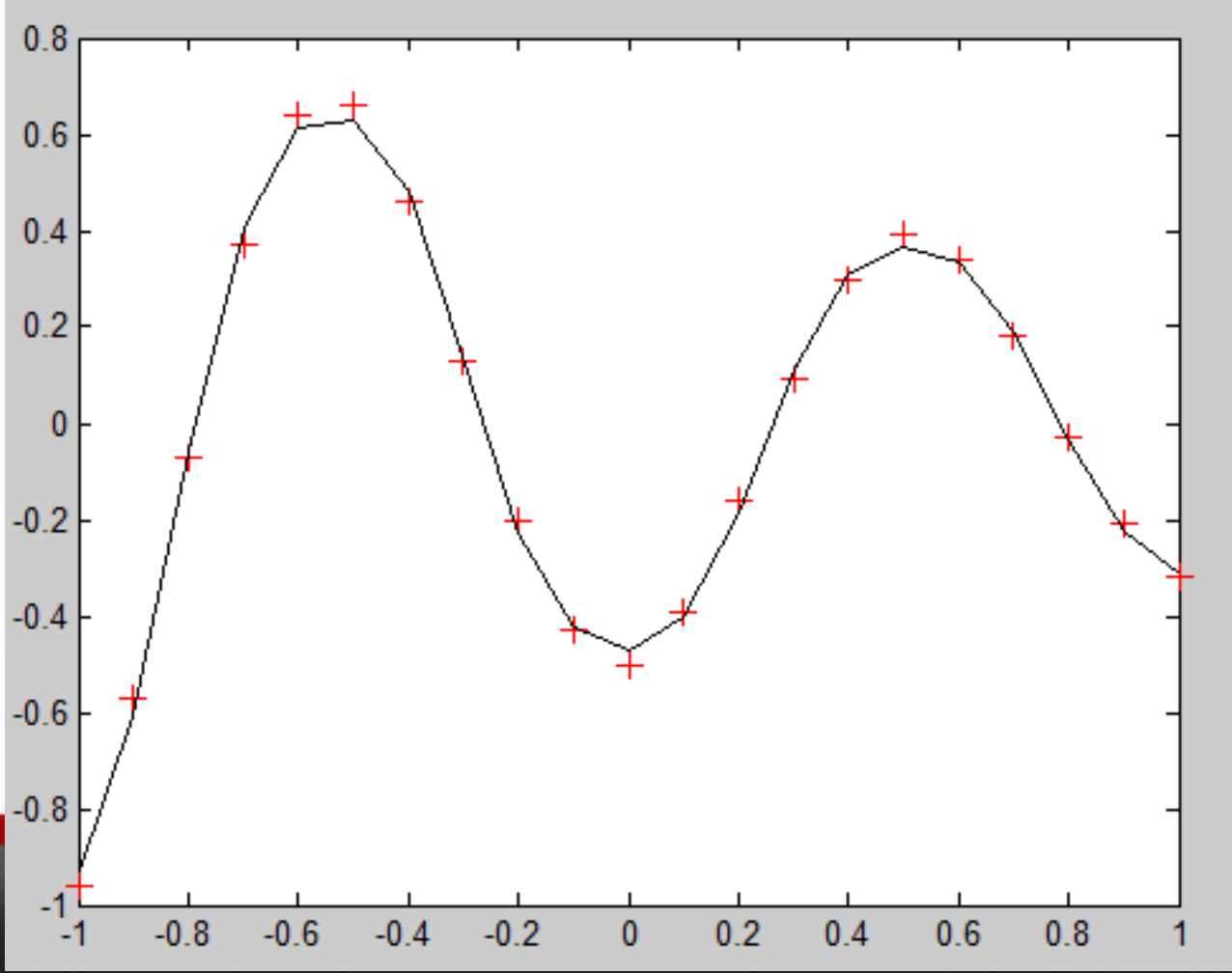


```
net.trainParam.show=10;
net.trainParam.lr=0.001;
net.trainParam.epoch=50;
net.trainParam.goal=0.05;
```



مثال :

```
net.trainParam.epoch=500;
```



مثال ۲ :

$$x_A = (1,2)$$

$$x_B = (2,1)$$

$$x_C = (2,3)$$

$$x_D = (3,1)$$

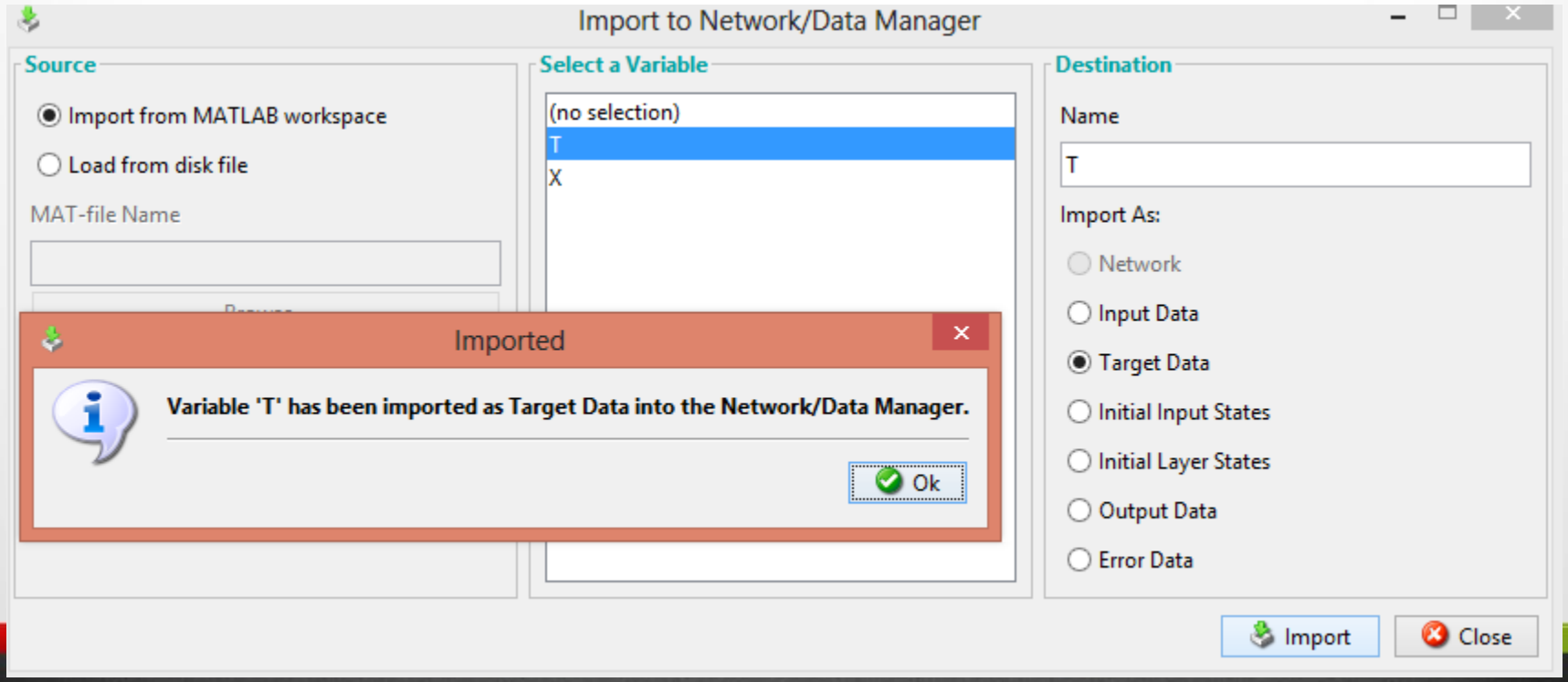
$$T = (4,5,7,7)$$

```
>> X=[1 2 3 3;2 1 3 1]
X =
     1     2     3     3
     2     1     3     1
>> T=[4 5 7 7]
T =
     4     5     7     7
```

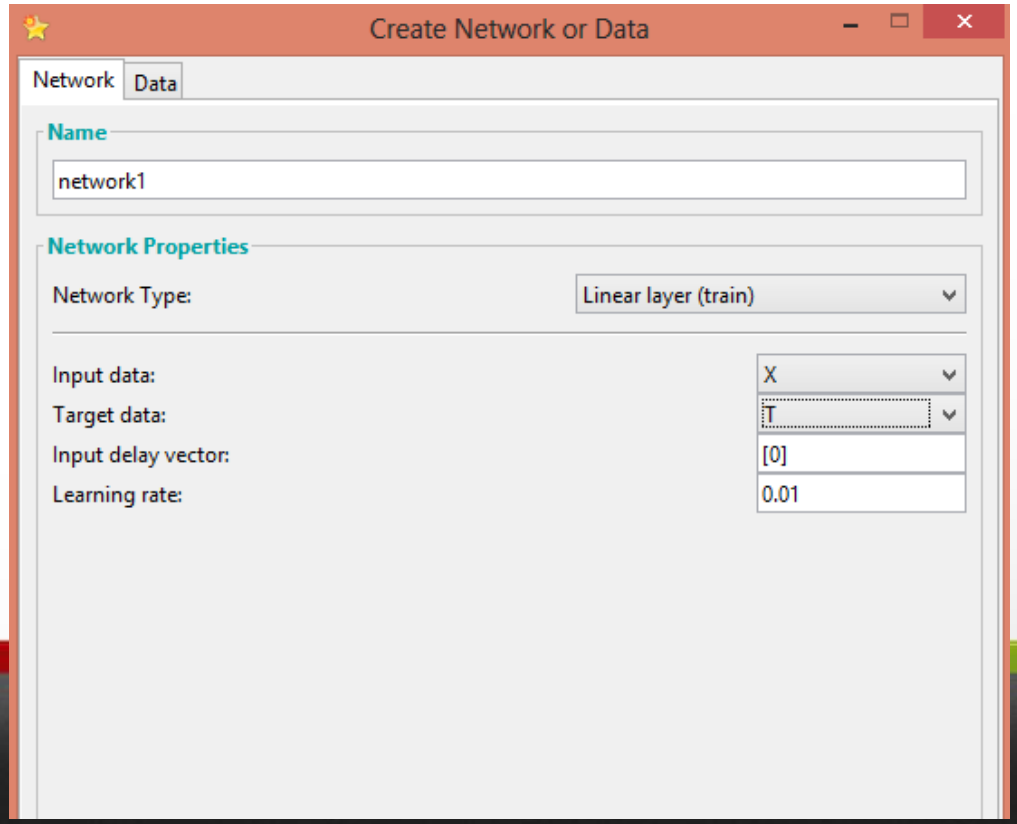
Workspace

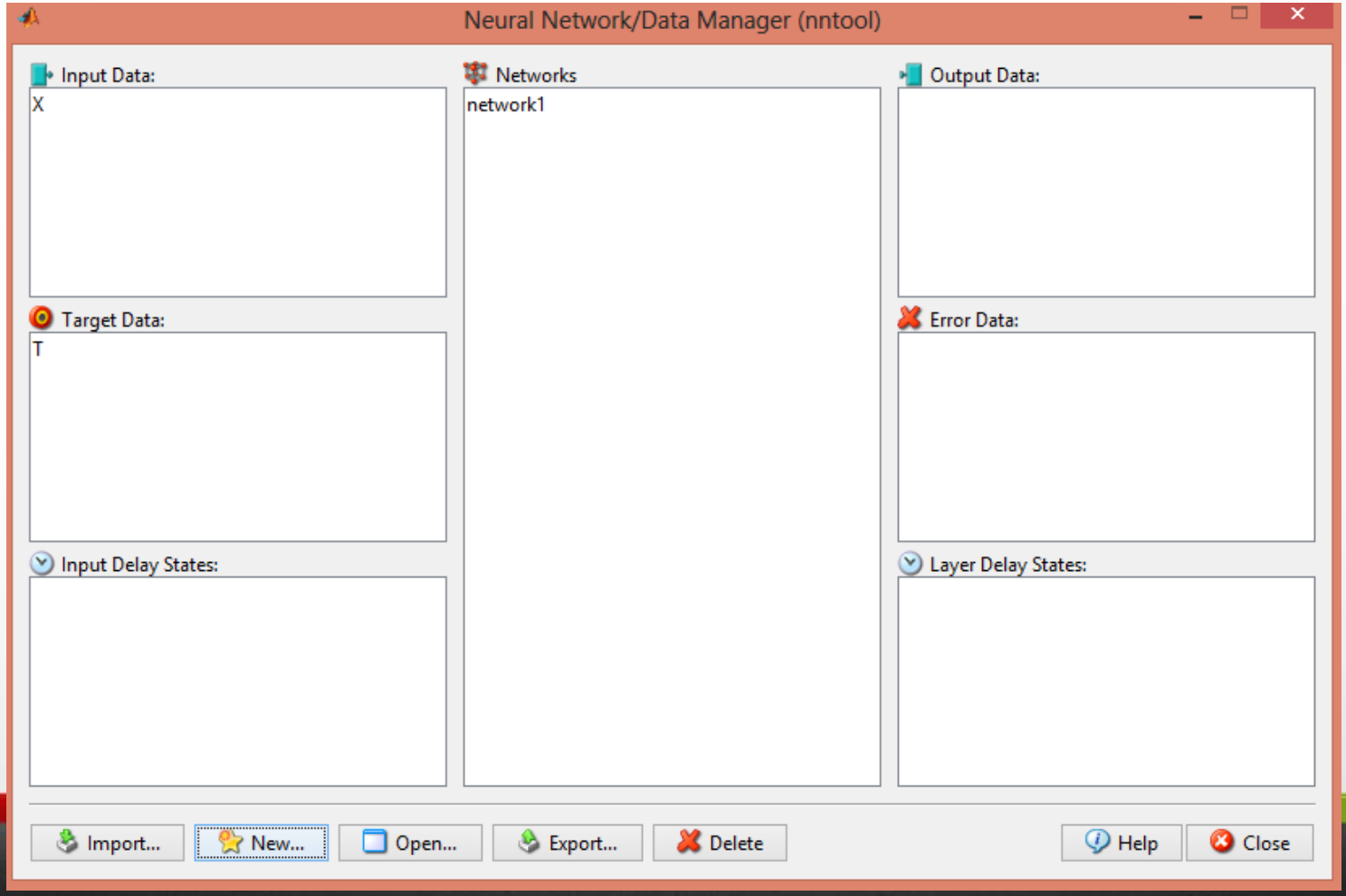
Name	Value	Min	Max
T	[4,5,7,7]	4	7
X	[1,2,3,3;2,1,3,1]	1	3

ابتدا اطلاعات را وارد می کنیم.

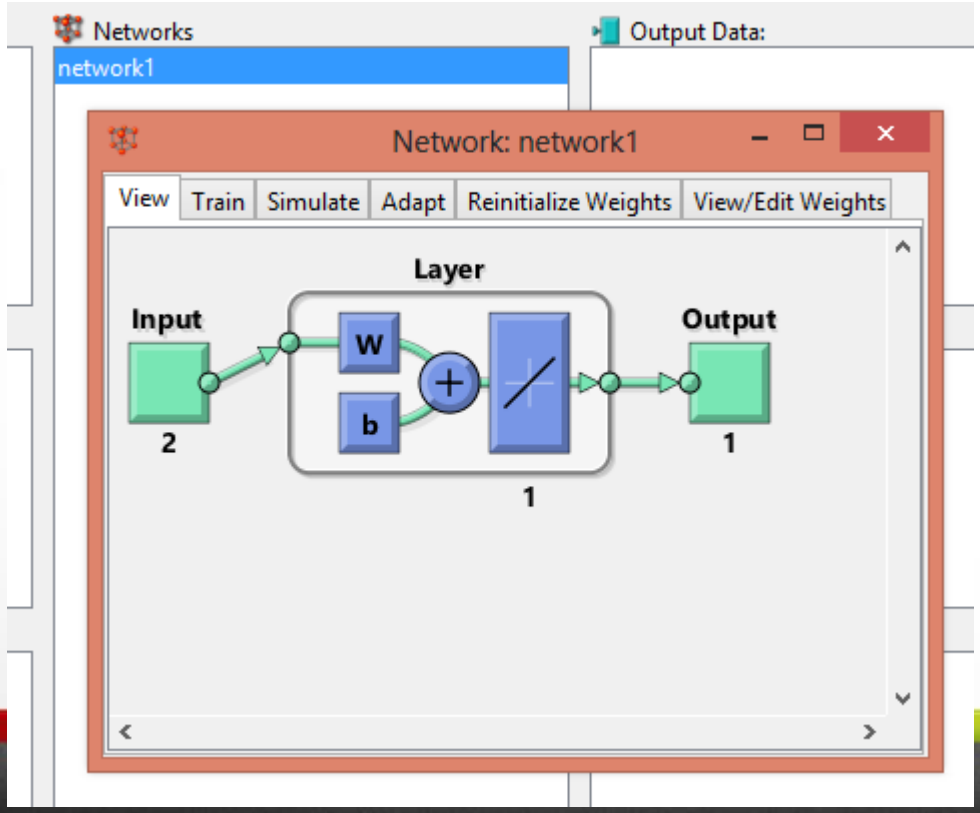


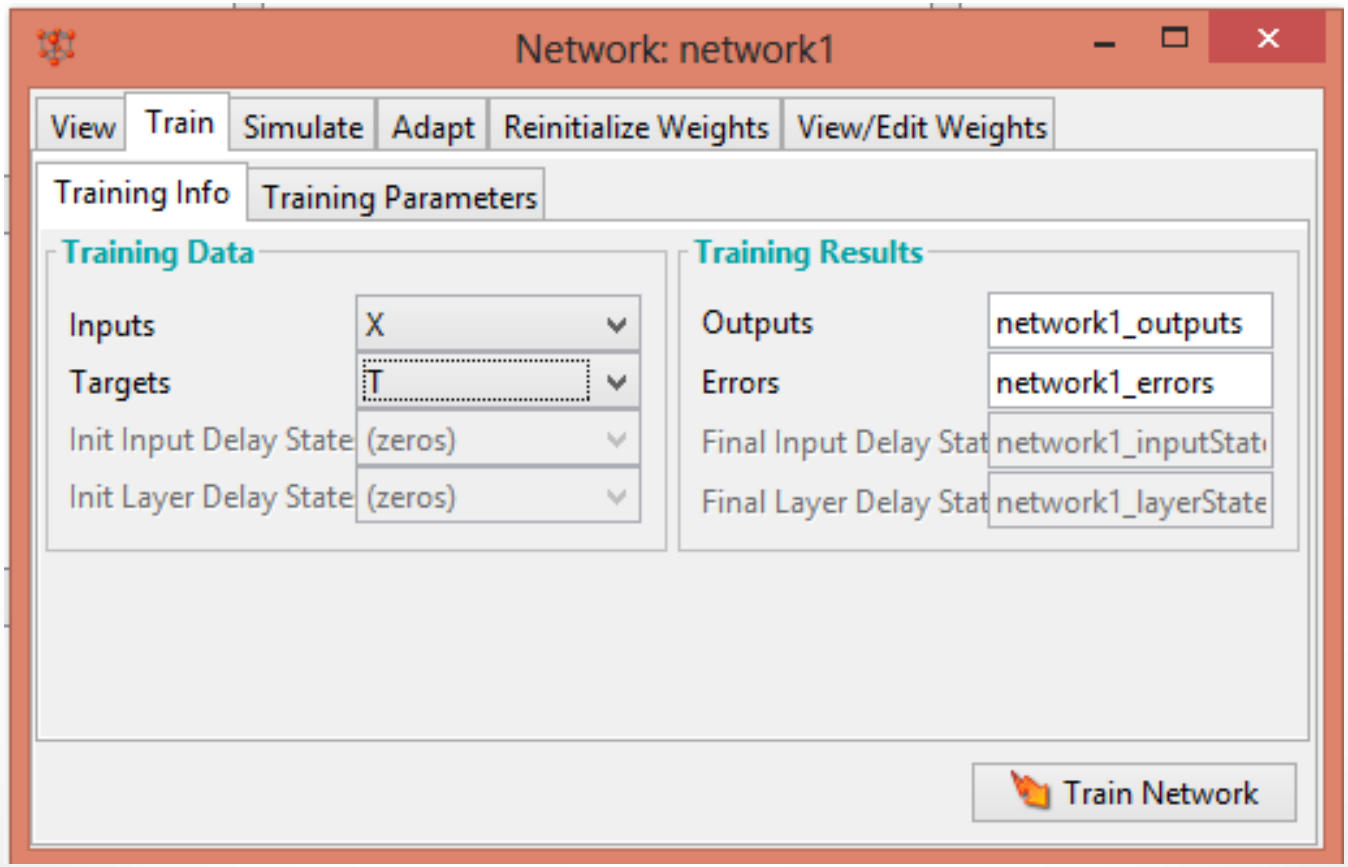
در این مثال می خواهیم یک شبکه خطی یک لایه و پارامتریهای تنظیم شده زیر را بسازیم، برای این کار بر روی دکمه New کلیک می کنیم، و سپس در قسمت Network پارامترها را به صورت زیر تعریف می نماییم و در نهایت Create میکنیم.

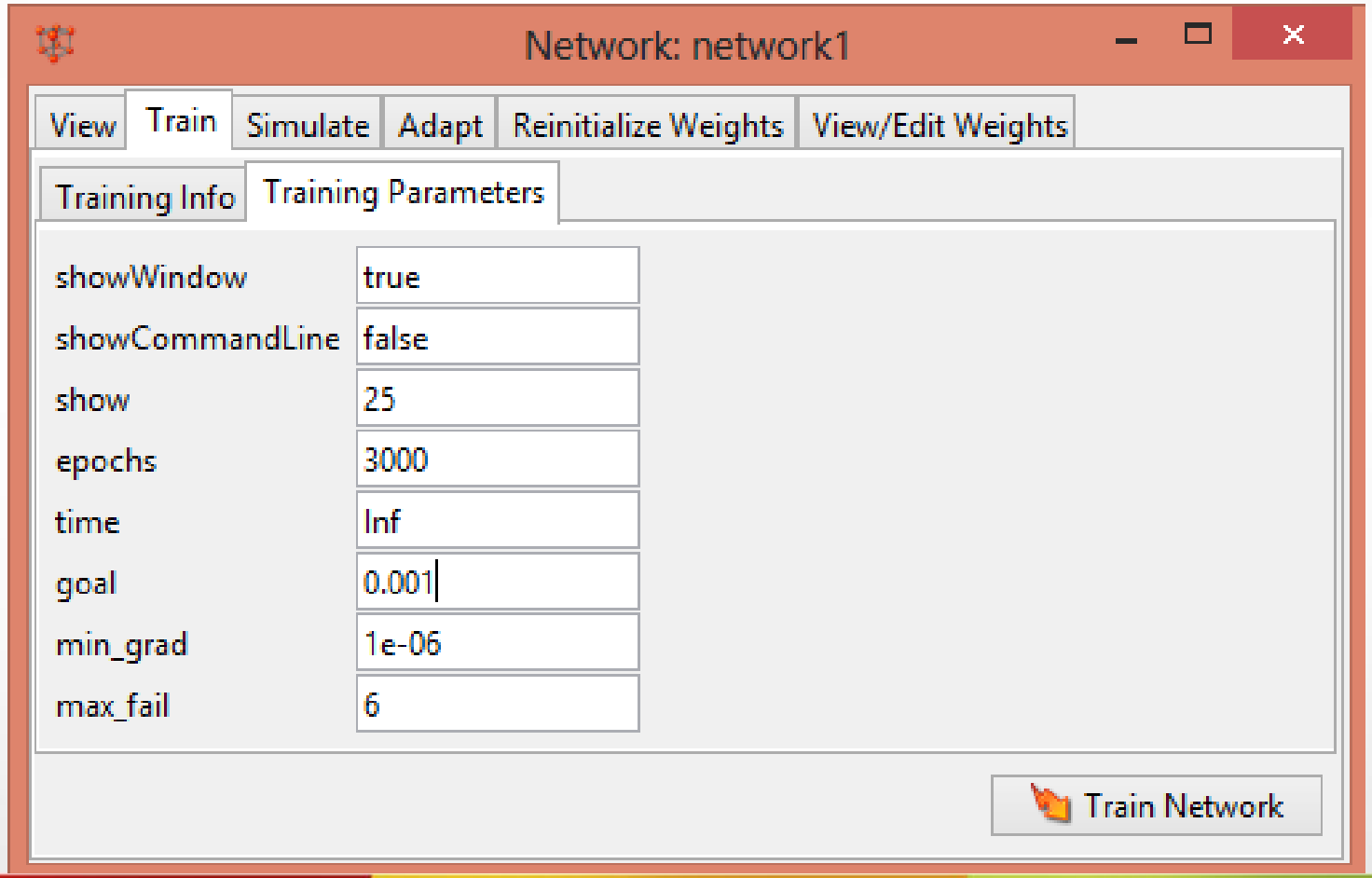


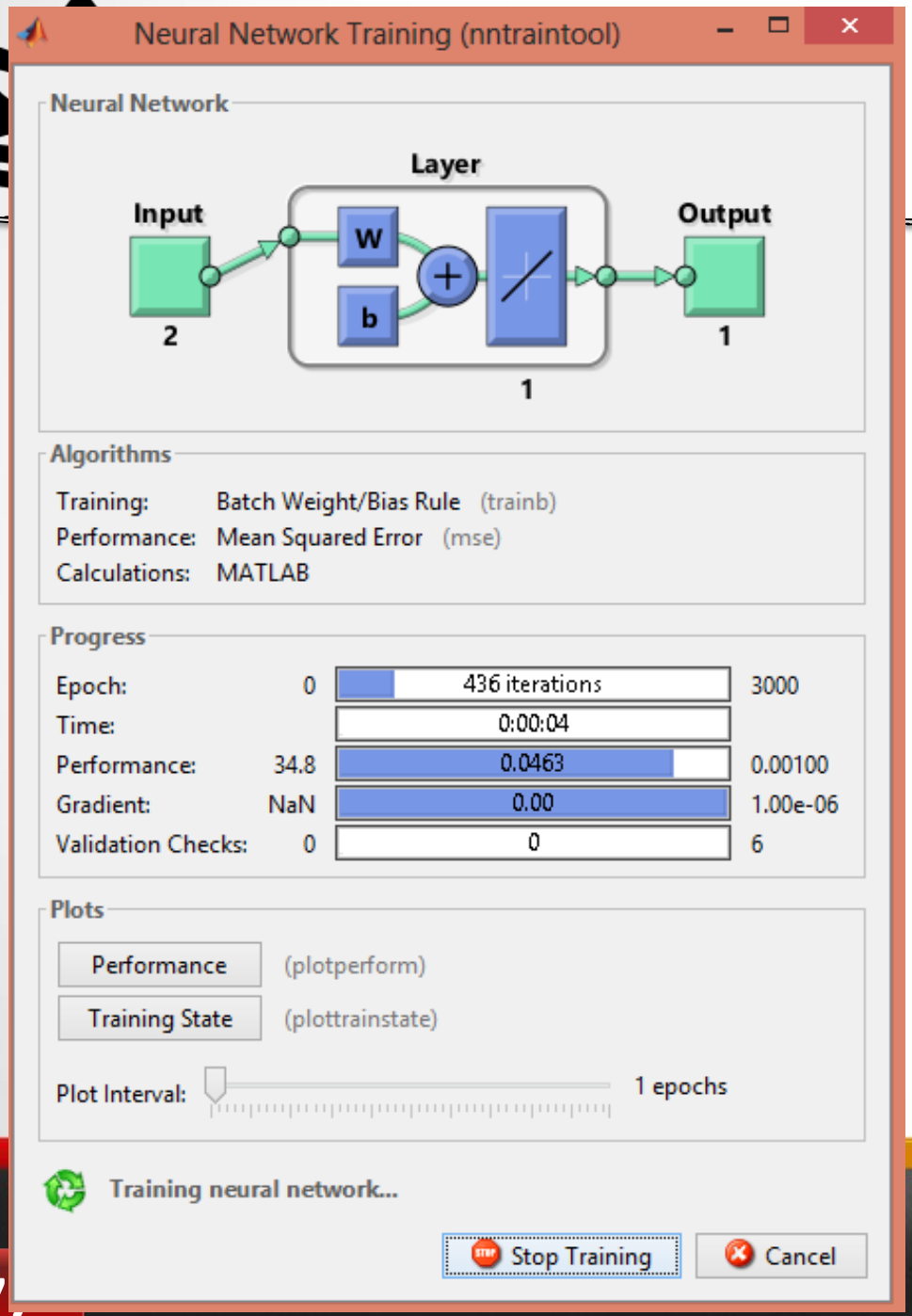

Neural Network/Data Manager (nntool)
Input Data:
X
Networks
network1
Output Data:
Target Data:
T
Error Data:
Input Delay States:
Layer Delay States:
Import...
New...
Open...
Export...
Delete
Help
Close

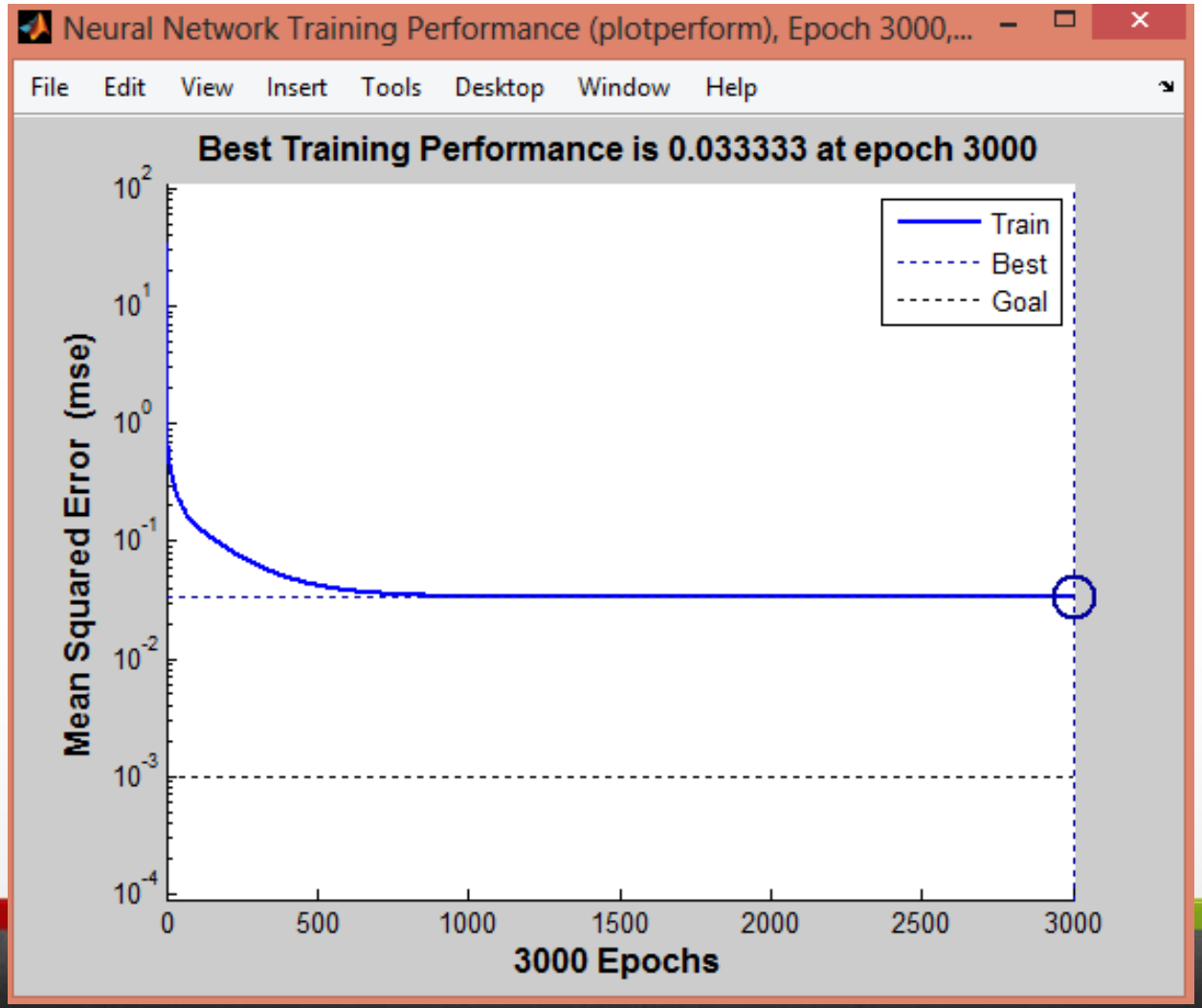
حال با کلیک بر روی Network1 پنجره ای باز می شود که در آن می توانیم تنظیمات بیشتری را انجام دهیم.



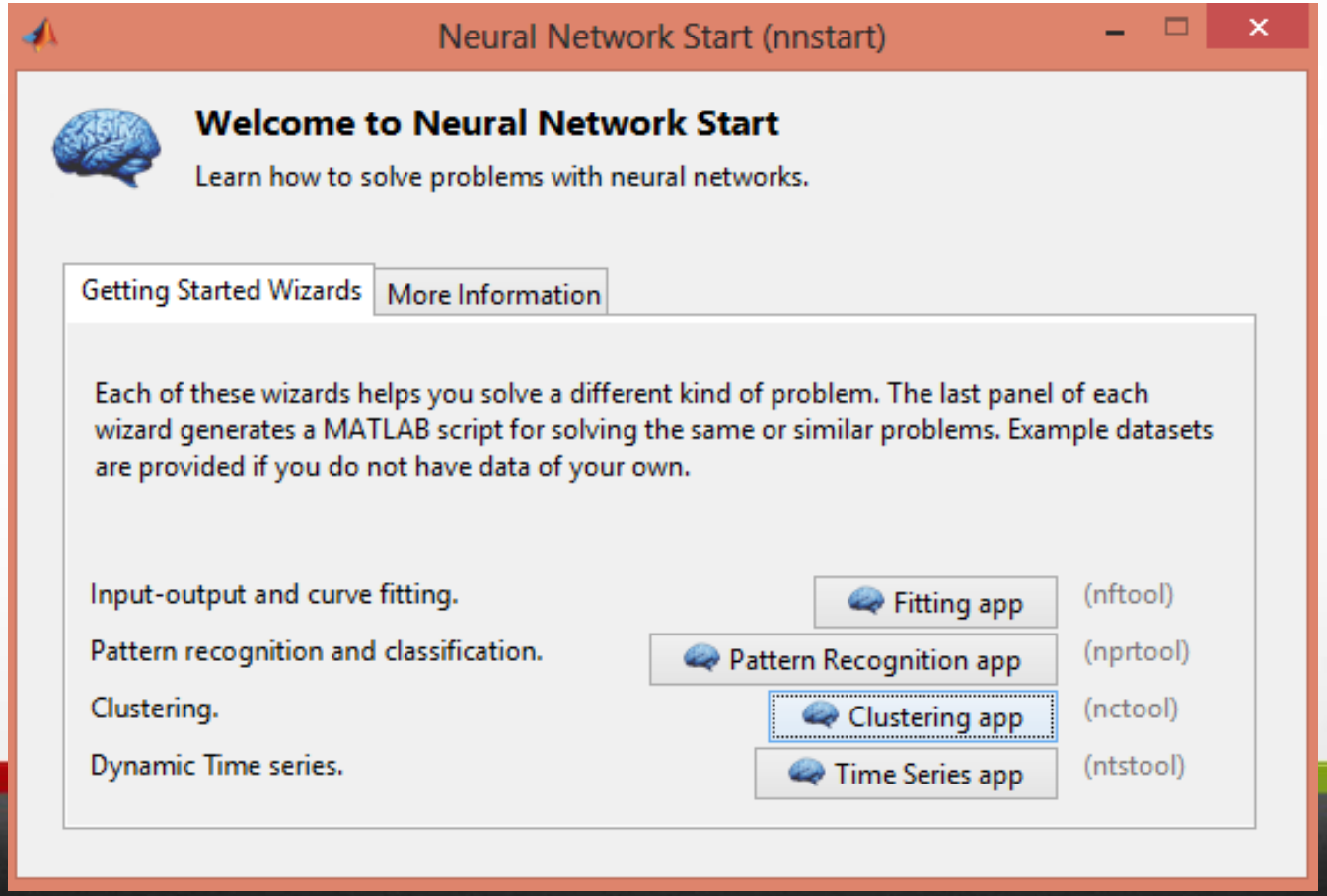

Network: network1
View
Train
Simulate
Adapt
Reinitialize Weights
View/Edit Weights
Training Info
Training Parameters
Training Data
Inputs
X
Targets
T
Init Input Delay State
(zeros)
Init Layer Delay State
(zeros)
Training Results
Outputs
network1_outputs
Errors
network1_errors
Final Input Delay Stat
network1_inputStat
Final Layer Delay Stat
network1_layerState
Train Network


Network: network1
View
Train
Simulate
Adapt
Reinitialize Weights
View/Edit Weights
Training Info
Training Parameters
showWindow
true
showCommandLine
false
show
25
epochs
3000
time
Inf
goal
0.001
min_grad
1e-06
max_fail
6
Train Network


Neural Network Training (nntraintool)
Neural Network
Layer
Input
W
b
Output
2
1
1
Algorithms
Training: Batch Weight/Bias Rule (trainb)
Performance: Mean Squared Error (mse)
Calculations: MATLAB
Progress
Epoch: 0 436 iterations 3000
Time: 0:00:04
Performance: 34.8 0.0463 0.00100
Gradient: NaN 0.00 1.00e-06
Validation Checks: 0 0 6
Plots
Performance (plotperform)
Training State (plottrainstate)
Plot Interval: 1 epochs
Training neural network...
Stop Training
Cancel


Neural Network Training Performance (plotperform), Epoch 3000,...
File Edit View Insert Tools Desktop Window Help
Best Training Performance is 0.033333 at epoch 3000
Train
Best
Goal
Mean Squared Error (mse)
10^{2}
10^{1}
10^{0}
10^{-1}
10^{-2}
10^{-3}
10^{-4}
0
500
1000
1500
2000
2500
3000
3000 Epochs

```
>> nnstart
```


Neural Network Start (nnstart)
Welcome to Neural Network Start
Learn how to solve problems with neural networks.
Getting Started Wizards
More Information
Each of these wizards helps you solve a different kind of problem. The last panel of each wizard generates a MATLAB script for solving the same or similar problems. Example datasets are provided if you do not have data of your own.
Input-output and curve fitting.
Fitting app
(nftool)
Pattern recognition and classification.
Pattern Recognition app
(nprtool)
Clustering.
Clustering app
(nctool)
Dynamic Time series.
Time Series app
(ntstool)

پایان ...