این محتوای آموزشی توسط وبسایت پاورسیم
www.PowerSim.ir
تهیه شده است.

شبکه های عصبی مصنوعی

شبکه های عصبی زیستی

- قدرت زیاد مغز انسان در تفکر، یادگیری، به یادآوری، حل مسائل و ... سبب شد، تا دانشمندان به مدلسازی آن بپردازند.

- می‌توان ایده اصلی شبکه‌های عصبی مصنوعی را در الهام از ساختار مغز و سیستم اعصاب طبیعی دانست.

- مغز انسان مجموعه ای بسیار عظیم از پردازشگرهایی موازی به نام نورون اند که به صورت هماهنگ برای حل مسئله عمل می کنند و توسط سیناپس ها(ارتباط های الکترومغناطیسی) اطلاعات را منتقل می کنند.در این شبکه ها اگر یک سلول آسیب ببیند بقیه ی سلولها می توانند نبود آنرا جبران کرده و نیز در بازسازی آن سهیم باشند.

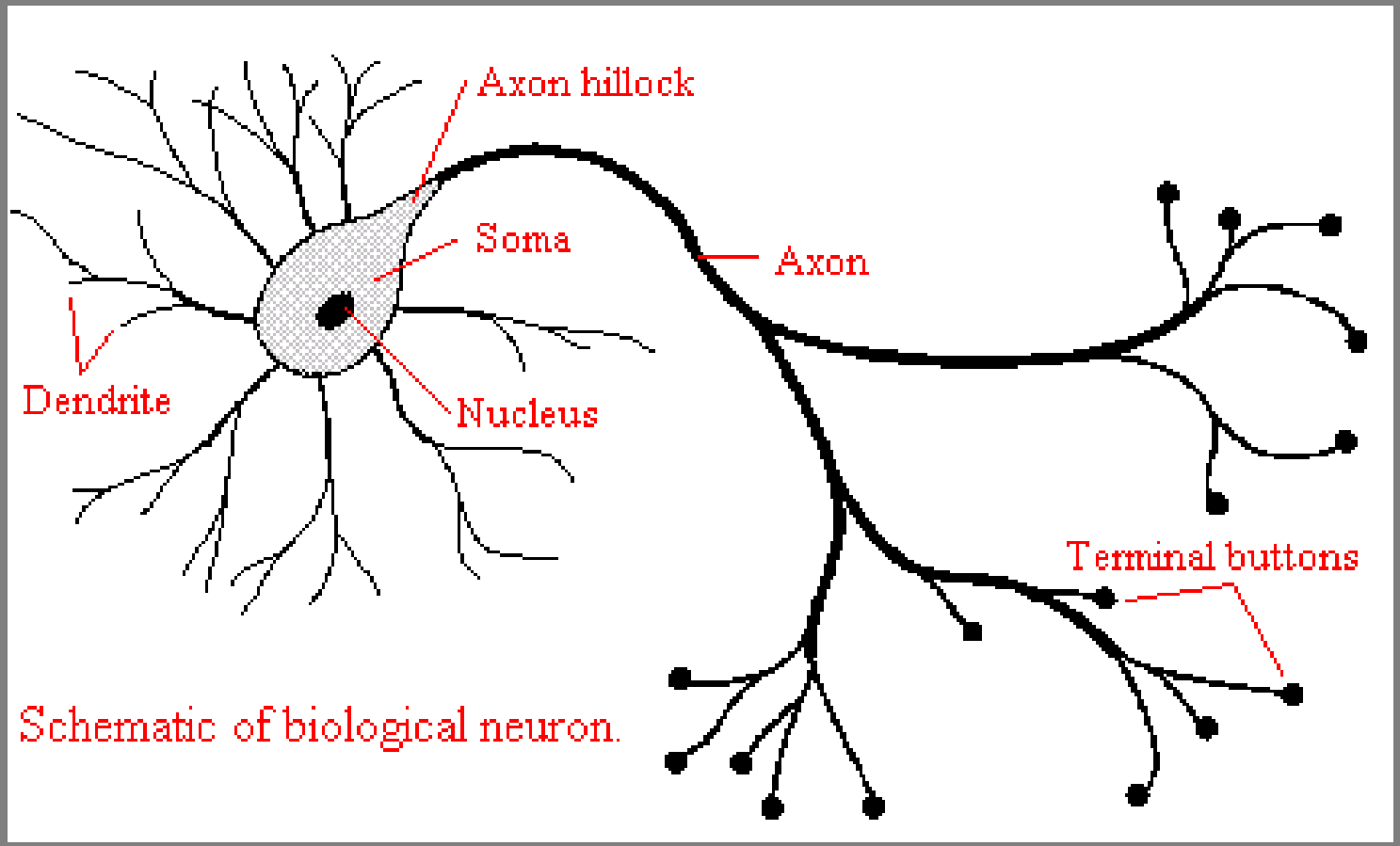
شبکه های عصبی زیستی

- این شبکه ها قادر به یادگیری اند. مثلا با اعمال سوزش به سلولهای عصبی لامسه، سلولها یاد می گیرند که به طرف جسم داغ نروند و با این الگوریتم سیستم می آموزد که خطای خود را اصلاح کند.

یادگیری در این سیستم ها به صورت تطبیقی صورت می گیرد، یعنی با استفاده ازمثال ها وزن سیناپس ها به گونه ای تغییر می کند که در صورت دادن ورودی های جدید سیستم پاسخ درستی تولید کند.

شمای یک نرون بیولوژیکی

اجزاء یک نرون بیولوژیکی



اجزاء یک نرون بیولوژیکی

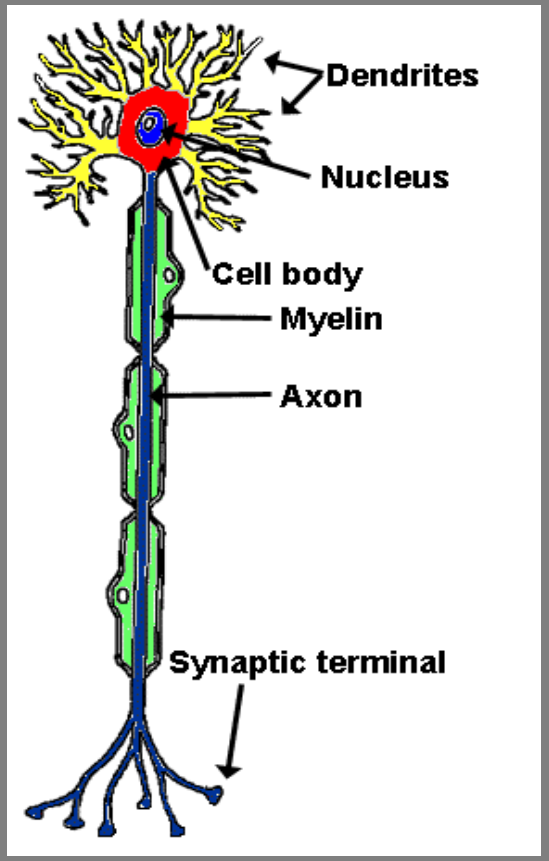
- هر نرون دارای یک بدنه اصلی به‌نام soma می‌باشد.
- سیگنال‌های تولیدشده در soma، از طریق axon به نرون‌های بعدی منتقل می شوند.
- هر نرون دارای یک ساختار درختی در اطراف soma است که dendrites نامیده می‌شوند. وظیفه آنها دریافت اطلاعات تولید شده توسط نرون‌های دیگر است. بدین‌وسیله یک نرون به هزاران نرون مجاور مرتبط می‌شود.

شبکه‌های عصبی مصنوعی

- شبکه‌های عصبی مصنوعی از تعداد زیادی وسیله محاسباتی ساده به نام نرون تشکیل شده‌اند که ارتباطات متعدد و بسیار زیاد بین آنها سبب شده است تا ضمن شرکت در نوع خاصی از پردازش موازی و افزایش سرعت در صدور پاسخ به تحریکات، امکان مدلسازی هر نوع رابطه غیرخطی را فراهم آورند.

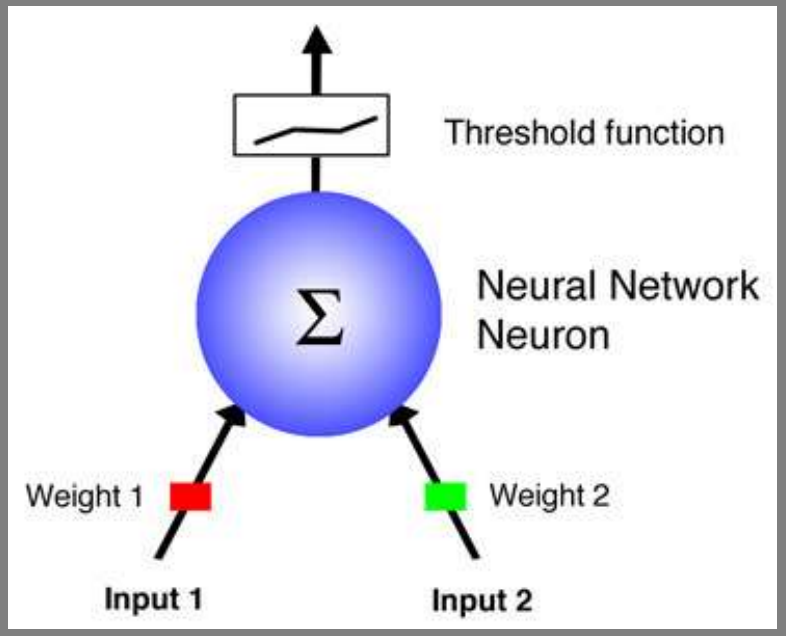
شبکه عصبی مصنوعی

- Biological Neurons



http://faculty.washington.edu/chudler/color/pic1an.gif

- Artificial Neurons

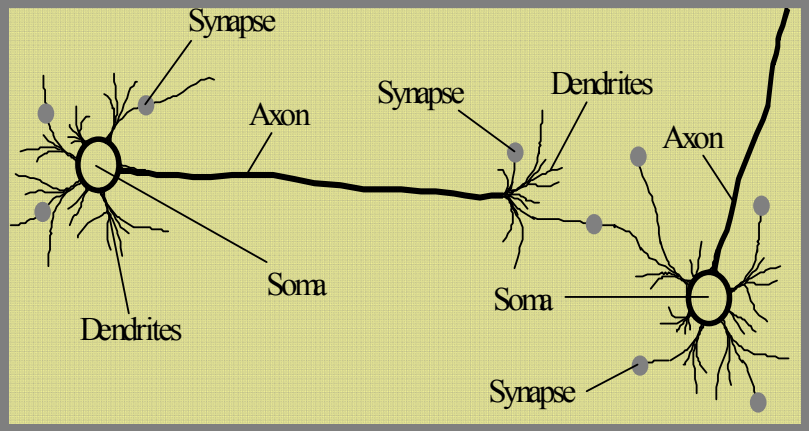


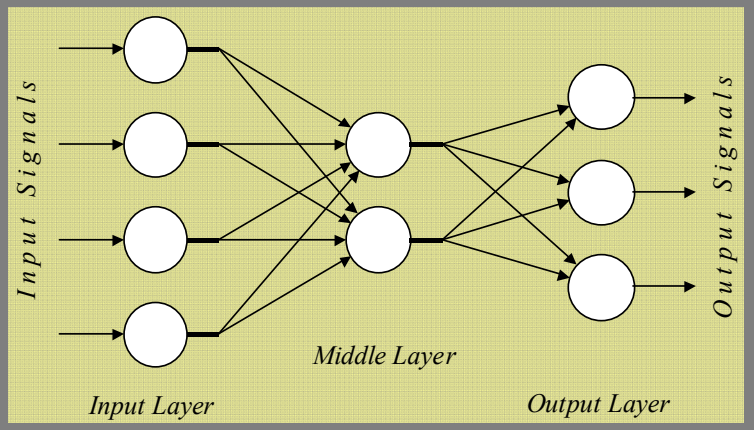
http://research.yale.edu/ysm/images/78.2/articles-neural-neuron.jpg

شبکه عصبی مصنوعی

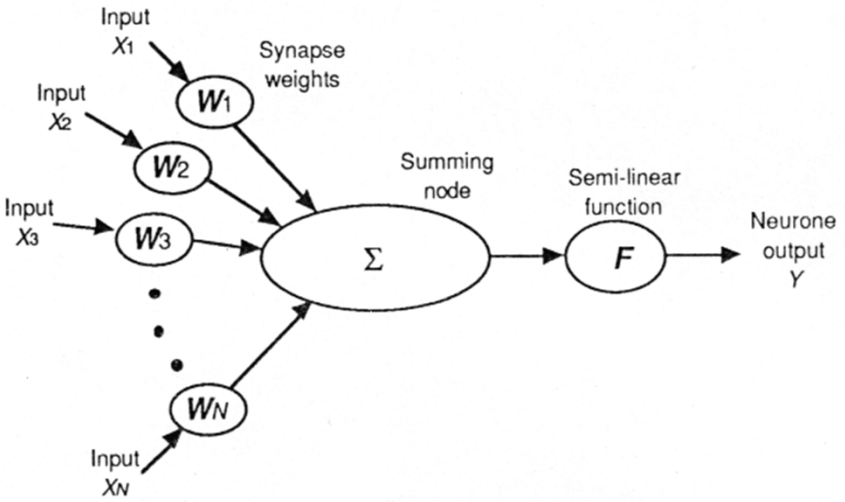
- تناظر بین شبکه عصبی و شبکه مصنوعی

Biological Neural Network	*Artificial Neural Network*
Soma	Neuron
Dendrite	Input
Axon	Output
Synapse	Weight





ساختار یک نمونه عصب مصنوعی



$$Y = F\left(\sum_{i=1}^{N} W_i x_i + b\right)$$

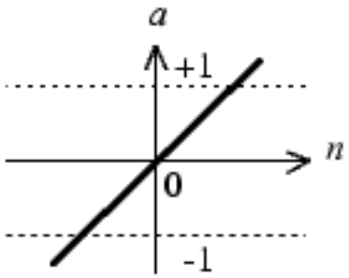
- عصب مصنوعی چندین سیگنال ورودی را دریافت می‌کند و یک خروجی را تولید می‌نماید.
- هرکدام از سیگنالهای ورودی در وزن عصبی مربوط به خود ضرب می‌شوند و در قسمت گره جمع‌کننده با یکدیگر و با یک مقدار بایاس جمع می‌شوند.
- حاصل این عمل وارد یک تابع عملکرد می‌شود و خروجی را نتیجه می‌دهد.

مشخصات توابع عملکرد

- Range تابع محدود باشد. (معمولاً بین ۱- و ۱ یا ۰ و ۱)
- این توابع می‌بایست یکنوا باشند.
- این توابع باید مشتق‌پذیر باشند (این ویژگی در بعضی از الگوریتم‌های آموزش لازم می‌باشد).

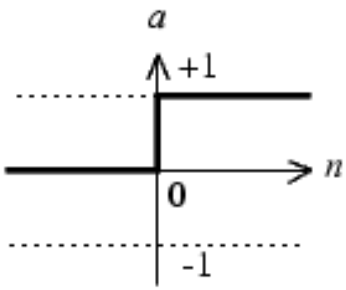
چند نوع تابع عملکرد

Linear transfer function



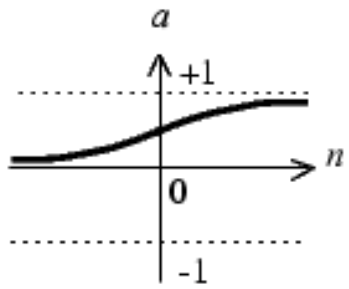
$a = purelin(n)$

Hard limit transfer function



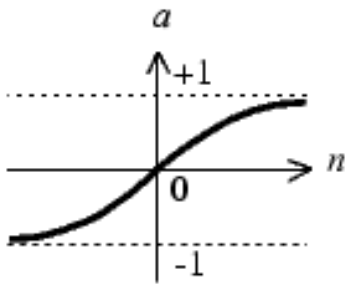
$a = hardlim(n)$

Log sigmoid transfer function



$a = logsig(n)$

Tan sigmoid transfer function

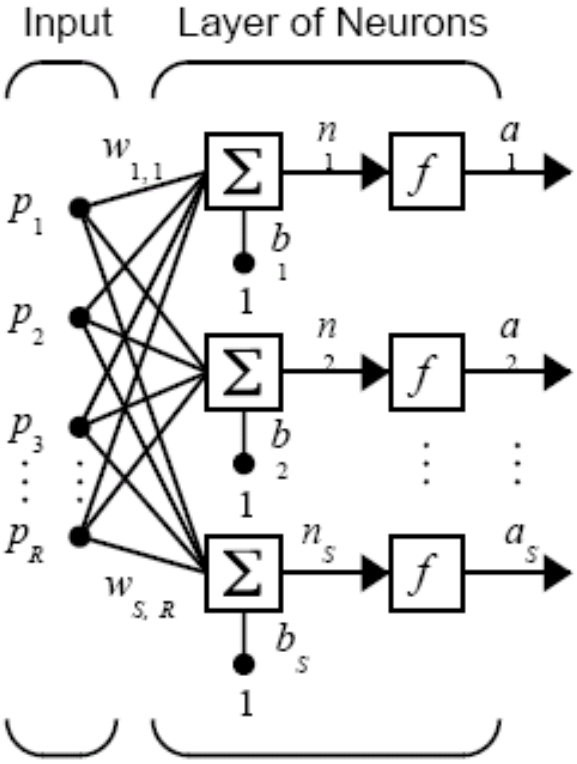


$a = tansig(n)$

اجزاء شبکه عصبی مصنوعی

- یک شبکهٔ عصبی مصنوعی از اتصال چندین عصب مصنوعی بدست می‌آید.
- تکنیک‌های متفاوتی برای اتصال عصب‌های مصنوعی به یکدیگر در جهت بدست آوردن یک شبکهٔ عصبی مصنوعی وجود دارد.
- در شبکه‌های تغذیه مستقیم خروجی در هر لحظه فقط وابسته به سیگنالهای ورودی همان لحظه و وزنهای عصبی می‌باشد.
- در بقیهٔ تکنیک‌های اتصال، شبکه عصبی دارای سیستم فیدبک می‌باشد و خروجی در هر لحظه، علاوه بر سیگنالهای ورودی در همان لحظه و وزنهای عصبی، به خروجی (یا برخی از خروجیهای) لحظات قبل نیز وابسته می‌باشد.

شمای شبکه عصبی تک لایه



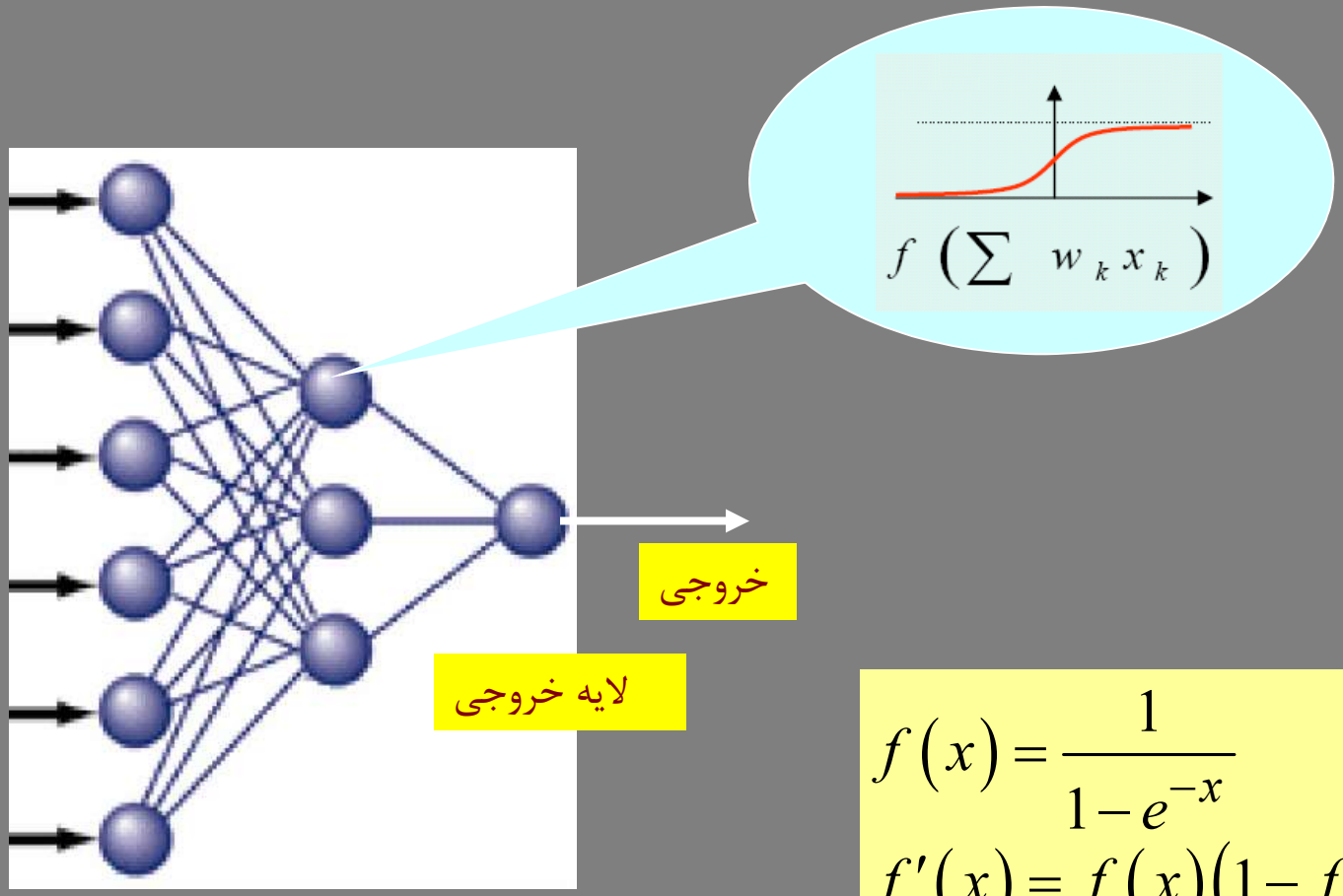
Where...

R = number of elements in input vector

S = number of neurons in layer

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

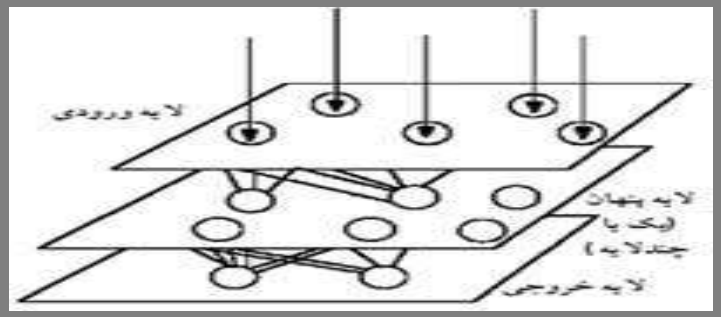
مدل شبکه عصبی مصنوعی



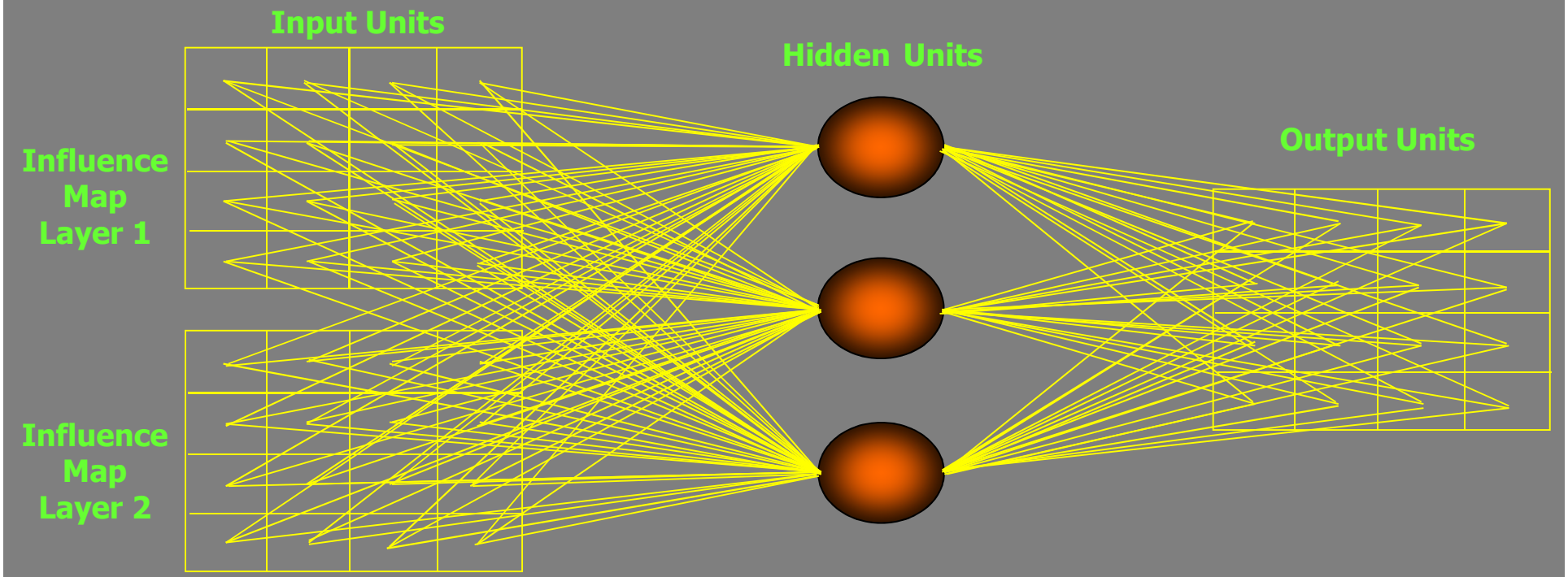
$$f(x) = \frac{1}{1 - e^{-x}}$$

$$f'(x) = f(x)\left(1 - f(x)\right)$$

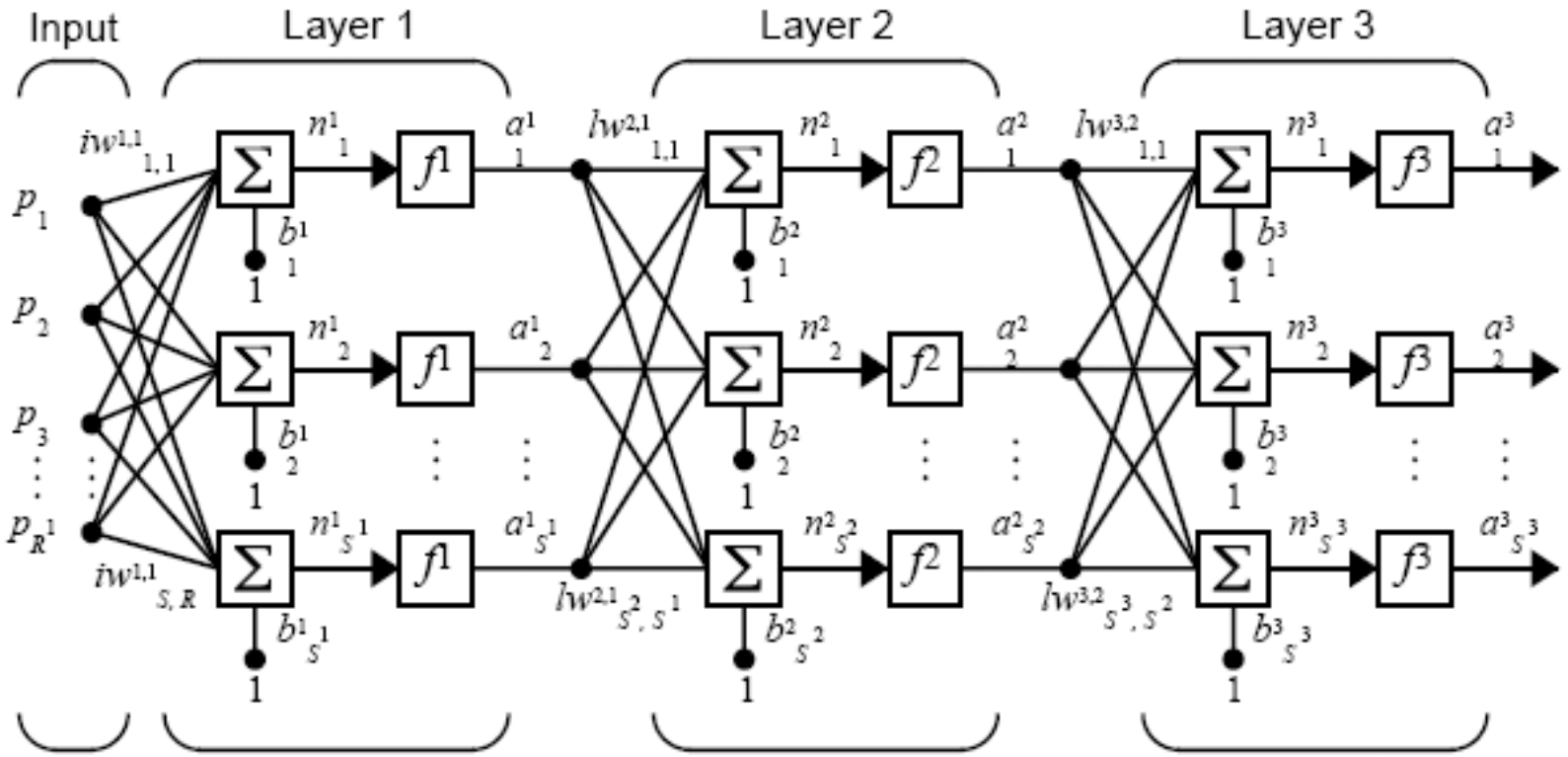
شبکه عصبی مصنوعی



- ساختار شبکه عصبی مصنوعی



شمای شبکه عصبی سه لایه



$$\mathbf{a}^1 = \mathbf{f}^1(\mathbf{IW}^{1,1}\mathbf{p} + \mathbf{b}^1) \qquad \mathbf{a}^2 = \mathbf{f}^2(\mathbf{LW}^{2,1}\mathbf{a}^1 + \mathbf{b}^2) \qquad \mathbf{a}^3 = \mathbf{f}^3(\mathbf{LW}^{3,2}\mathbf{a}^2 + \mathbf{b}^3)$$

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

کاربردهای شبکه‌های عصبی مصنوعی

- طبقه بندی، شناسایی و تشخیص الگو، خوشه بندی، طبقه بندی
  - نسبت دادن الگوی ورودی به یکی از کلاس‌های موجود در بانک اطلاعاتی- شناسایی دستخط، حروف لاتین، فارسی، تشخیص سبک نگارش، درجه غلظت روغن...
- پردازش سیگنال
  - فیلترهای تطبیقی،کدینگ، فشرده سازی، پردازش صحبت
- پیش‌بینی سریهای زمانی
  - پیش‌بینی مقادیر آینده یک سری زمانی بخصوص در حالت غیر ایستایی مثل پیش بینی بار، دما، قیمت

کاربردهای شبکه‌های عصبی مصنوعی

- مدلسازی و کنترل
  - مدلسازی سیستم های و پیاده سازی کنترلرهای پیچیده
- بهینه سازی
  - تخصیص منابع در بانکداری
- سیستمهای خبره
  - تنظیم سیستمهای خبره و فازی
  - مسایل مالی، بیمه، بازار بورس، پزشکی و...

انواع شبکه عصبی مصنوعی

- نوع پیوند یا اتصال (Connection type)
  - Feed forward (استاتیکی)
  - Feed back یا Recurrent (دینامیکی)
  - Lateral (جانبی)
- توپولوژی
  - تک لایه یا چند لایه
  - تعداد نرونهای هر لایه
- روش یادگیری (learning methods)
  - وزن ثابت (Fixed weight)
  - یادگیری بدون نظارت (Unsupervised) یا Self-Organized
  - یادگیری تحت نظارت (Supervised)
  - یادگیری تقویتی (Reinforcement)

مزایای استفاده از شبکه‌های عصبی مصنوعی

- قابلیت یادگیری
- قابلیت تعمیم و تطبیق
- پردازش موازی
- مقاوم بودن در برابر خطا
- Curve Fitting - جایی که یکسری داده ورودی- خروجی داریم که پدیده‌اش را نمی‌دانیم.
- شبکه‌های عصبی نسبت به سیستم خبره این مزیت را دارد که بر تجارب بشری تکیه نداشته و خودش رابطه ورودی- خروجی را پیدا می‌کند.

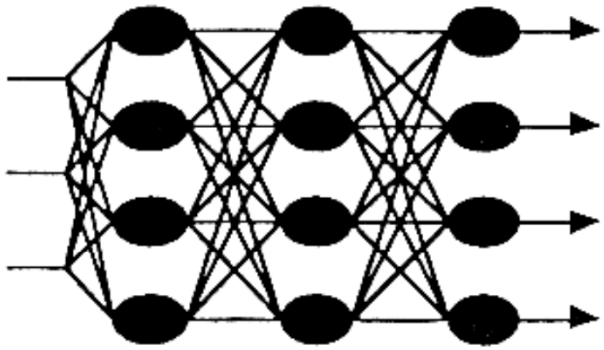
انواع اتصالات در شبکه های عصبی مصنوعی

- Feed forward(استاتیکی)
- بیشترین نوع پیوندهای شبکه ها از این نوع بوده و شامل اتصالاتی است که گره های لایه $n-1$ام را به لایه nام متصل می نماید. اطلاعات از نرونهای لایه پایین تر به نرونهای لایه بالاتر بطور پیشرو انتشار می یابند.

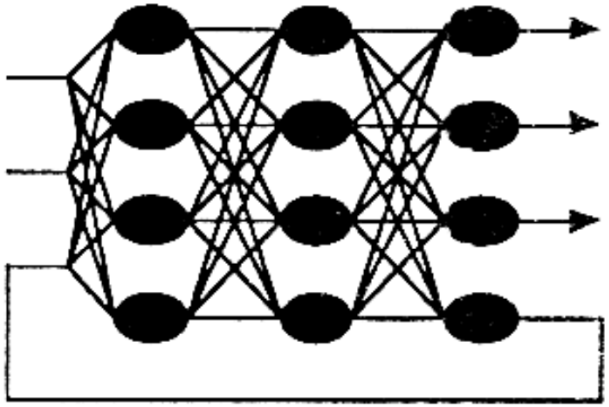
- Feed back یا Recurrent (دینامیکی)
- داده ها از نرونهای لایه بالاتر به نرونهای لایه پایینتر فیدبک می شوند.

- Lateral (جانبی)
- اتصال بین نرونهای یک لایه برقرار می شود.

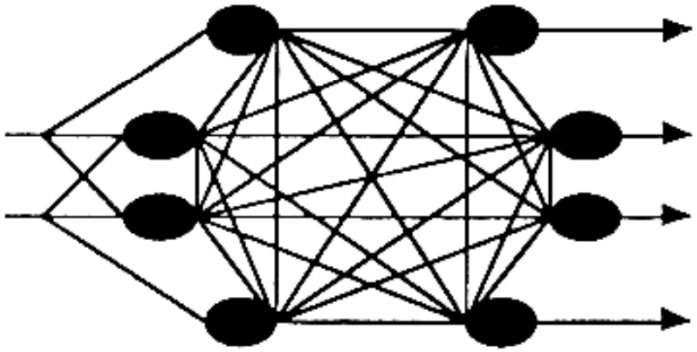
چند نوع اتصال شبکه عصبی مصنوعی



Feedforward net



Partially recurrent ne



Fully recurrent net

روش‌های یادگیری شبکه عصبی مصنوعی

- یادگیری وزن ثابت:

دراین روش عملا آموزشی در کار نیست و مقادیر وزنها بهنگام نمی شوند. مثال: کدینگ یا فشرده سازی

- یادگیری تحت نظارت: در یادگیری تحت نظارت، یک دسته از ورودیها و خروجیهای مطلوب برای تعلیم شبکه بکار می‌روند. شبکه با استفاده از این ورودیها، خروجیهایی را تولید می‌کند. این خروجیها با خروجیهای مطلوب مقایسه می‌شوند و اختلاف آنها برای تنظیم وزنها بکار می‌رود. در این روشها، یا از خروجی به وزنها ارتباط پسرو وجود دارد ویا خطا بصورت پس انتشار از لایه خروجی به ورودی توزیع شده و وزنها اصلاح میشوند (Back propagation).

روش‌های یادگیری شبکه عصبی مصنوعی

- یادگیری بدون نظارت: در این روش خروجی مطلوب برای تعلیم شبکه وجود ندارد. شبکه بایستی خود تصمیم‌گیری کند که داده‌های ورودی را چگونه دسته‌بندی کند. معمولاً به این روش تطبیق (Adaptation) می‌گویند. مثال: خوشه زنی، تشخیص الگو

- یادگیری تقویتی: در این روش، عملکرد شبکه با زمان بهبود می یابد. در یادگیری تقویتی الگوی آموزشی مشخصی بعنوان معلم وجود ندارد اما با استفاده از سیگنالی به نام سیگنال نقاد بیانی از خوب یا بد بودن خروجی شبکه بدست می آید.

این روش حالتی بین یادگیری تحت نظارت و بدون نظارت است و منطق حاکم بر آن همان تخصیص امتیاز به عملکرد سیستم است.

مراحل آموزش شبکه‌های عصبی مصنوعی

- جمع آوری اطلاعات
- نرمالیزه کردن اطلاعات
- انتخاب ساختار شبکه عصبی
- آموزش شبکه
- تست شبکه
- انتخاب سایر ساختارهای شبکه عصبی و انجام آموزش‌های اضافی

مراجع

- مبانی شبکه های عصبی، دکتر محمدباقر منهاج، انتشارات دانشگاه صنعتی امیرکبیر، (چاپ اول: آذر ۷۹)
- شبکه های عصبی مصنوعی، رابرت جی. شالکف، مترجم: دکتر محمود جورابیان، انتشارات دانشگاه شهید چمران اهواز، ۱۳۸۴
- Digital neural networks, S. Y. Kung, Prinntice Hall, 1997.
- Neural networks, fuzzy logic, and genetic algorithms, synthesis and applications,
  - By: S. Rajasekaran
  - G.A. Vijayalakshmi pai

شناسایی الگو چیست؟

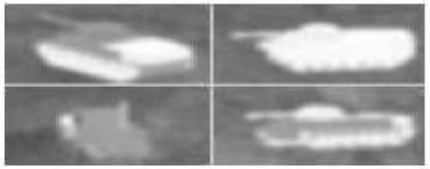
- The assignment of a physical object or event to one of several pre-specified case
- A **pattern** is an object, process or event that can be given a name.
- A **pattern** class (or category) is a set of patterns sharing common attributes and usually originating from the same source.
- During **recognition** (or **classification**) given objects are assigned to prescribed classes.
- A **classifier** is a machine which performs classification.

شناسایی الگو

- ساختار کلی یک مساله شناسایی الگو:
  - چیزی یا موضوعی اتفاق افتاده است.
  - اتفاق توسط سیگنالی نمایندگی می شود.
  - سیگنالی که مشاهده می شود، با سیگنال ارسالی یکسان نیست.
  - بر اساس سیگنال فرستاده شده، مشاهده کننده باید تصمیمی در مورد مساله یا موضوع واقع شده بگیرد.
- مثال:
  - شخصی بیمار است (اتفاق). به پزشک مراجعه می کند (مشاهده کننده)، بیمار را معاینه کرده، دستور تست آزمایشگاهی می دهد (سیگنالها که ممکن است نویزی باشند)، پزشک باید در مورد بیماری تصمیم بگیرد (تصمیم سازی).

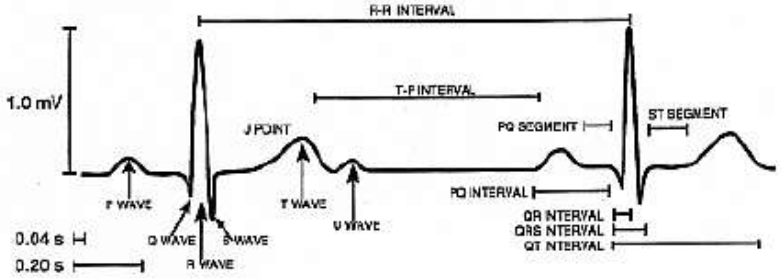
مثالهای دیگر

- **Machine vision**
  - Visual inspection
  - Imaging device detects ground target
  - Classification into “friend” or “foe”

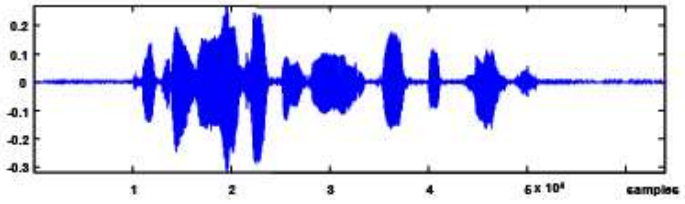


- **Character recognition**
  - Automated mail sorting, processing bank checks
  - Scanner captures an image of the text
  - Image is converted into constituent characters

- **Computer aided diagnosis**
  - Medical imaging, EEG, ECG signal analysis
  - Designed to assist (not replace) physicians
  - Example: X-ray mammography
    - 10-30% false negatives in x-ray mammograms
    - 2/3 of these could be prevented with proper analysis



- **Speech recognition**
  - Human Computer Interaction, Universal Access
  - Microphone records acoustic signal
  - Speech signal is classified into phonemes and/or words



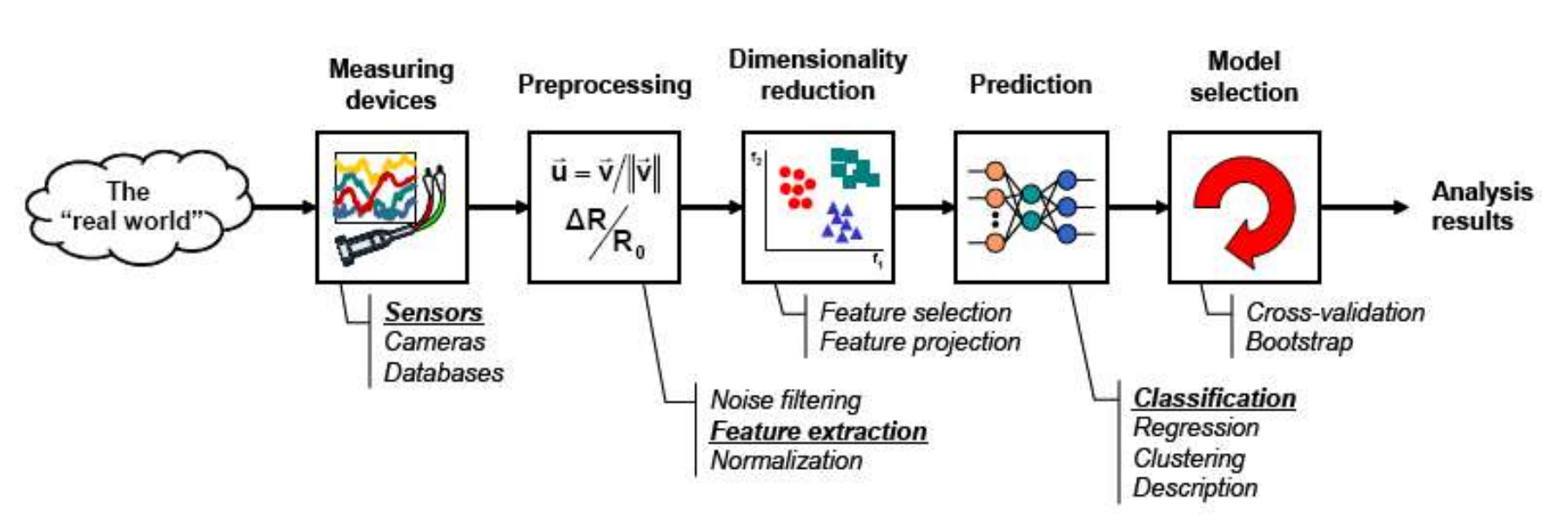
انواع مسایل درشناسایی الگو

- **Classification**
  - The PR problem of assigning an object to a class
  - The output of the PR system is an integer label
    - e.g. classifying a product as “good” or “bad” in a quality control test
- **Regression**
  - A generalization of a classification task
  - The output of the PR system is a real-valued number
    - e.g. predicting the share value of a firm based on past performance and stock market indicators
- **Clustering**
  - The problem of organizing objects into meaningful groups
  - The system returns a (sometimes hierarchical) grouping of objects
    - e.g. organizing life forms into a taxonomy of species
- **Description**
  - The problem of representing an object in terms of a series of primitives
  - The PR system produces a structural or linguistic description
    - e.g. labeling an ECG signal in terms of P, QRS and T complexes

مراحل شناسایی الگو

- تعریف: جدا سازی داده ها یا الگوهای ورودی بین دستجات مختلف.
- مراحل:
  - تبدیل اشیاء مورد شناسایی را به الگوها و یا بردارهای ورودی که به سیستم شناسا اعمال می گردند کدینگ
    - مولفه های این بردار از طریق اندازه گیری بدست می آیند.
    - هر کمیت اندازه گیری یک ویژگی از شیء مورد نظر را بیان می کند.
  - استخراج مشخصه های مهم از روی داده های اندازه گیری شده و کاهش ابعاد بردار الگوها feature extraction
  - اتخاذ روندی جهت تصمیم گیری بهینه اتوماتیک

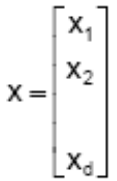
مراحل شناسایی الگو



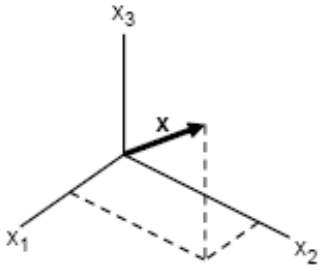
استخراج مشخصات

Feature

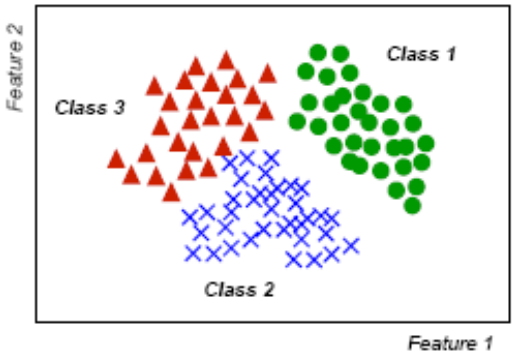
- Feature is any distinctive aspect, quality or characteristic
  - Features may be symbolic (i.e., color) or numeric (i.e., height)
- Definitions
  - The combination of d features is represented as a d-dimensional column vector called a **feature vector**
  - The d-dimensional space defined by the feature vector is called the **feature space**
  - Objects are represented as points in feature space. This representation is called a **scatter plot**



Feature vector



Feature space (3D)



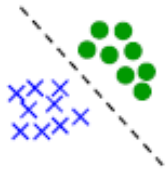
Scatter plot (2D)

Pattern

- Pattern is a composite of traits or features characteristic of an individual
- In classification tasks, a pattern is a pair of variables $\{x, \omega\}$ where
  - x is a collection of observations or features (feature vector)
  - ω is the concept behind the observation (label)

استخراج مشخصات

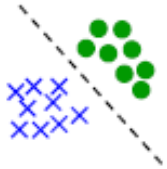
- **What makes a "good" feature vector?**
  - The quality of a feature vector is related to its ability to discriminate examples from different classes
    - Examples from the same class should have similar feature values
    - Examples from different classes have different feature values



"Good" features

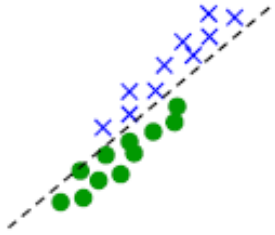
"Bad" features

- **More feature properties**

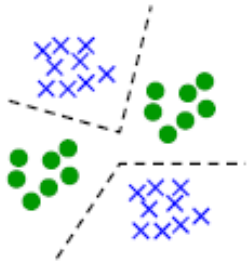


Linear separability

Non-linear separability



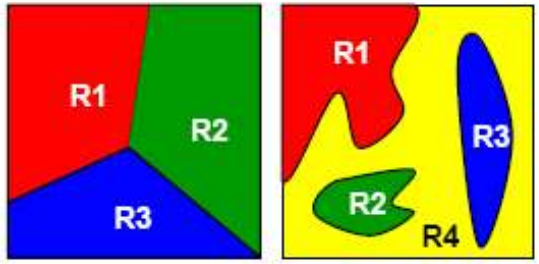
Highly correlated features



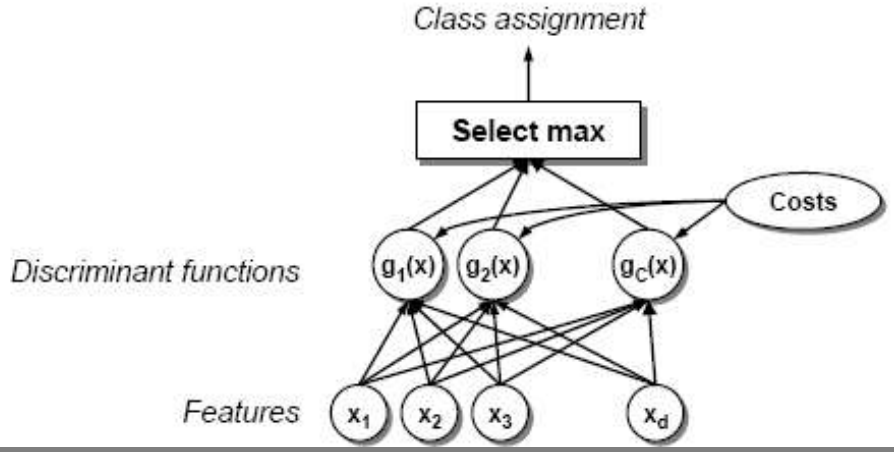
Multi-modal

Classifiers

- **The task of a classifier is to partition feature space into class-labeled decision regions**
  - Borders between decision regions are called **decision boundaries**
  - The classification of feature vector ***x*** consists of determining which decision region it belongs to, and assign ***x*** to this class



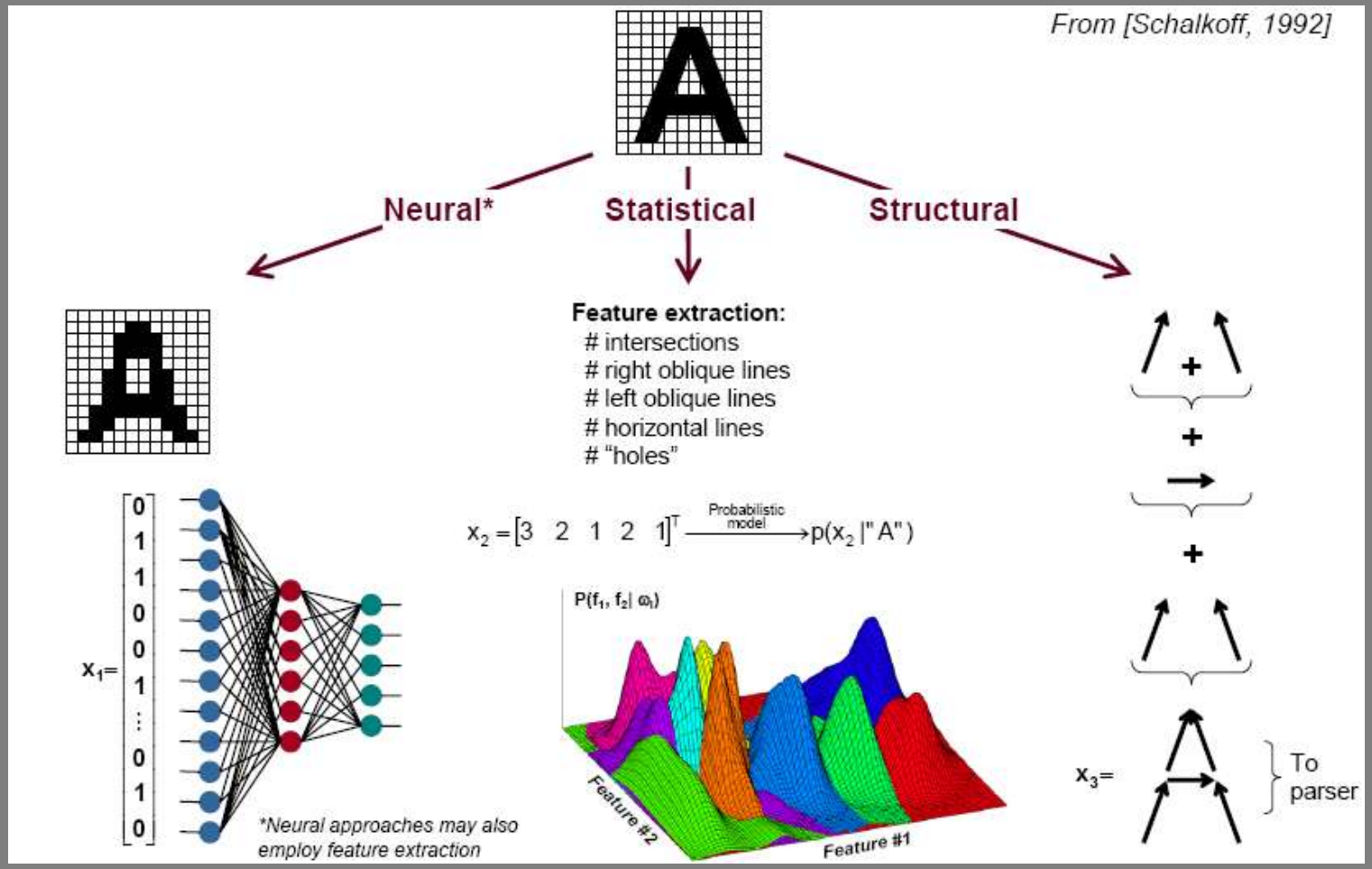
- **A classifier can be represented as a set of discriminant functions**
  - The classifier assigns a feature vector ***x*** to class ω_i if $g_i(\boldsymbol{x}) > g_j(\boldsymbol{x}) \quad \forall j \neq i$



روشهای شناسایی الگو

- **Statistical (StatPR)**
  - Patterns classified based on an underlying statistical model of the features
    - The statistical model is defined by a family of **class-conditional probability** density functions $Pr(x|c_i)$ (Probability of feature vector $\boldsymbol{x}$ given class $\boldsymbol{c_i}$)
- **Neural (NeurPR)**
  - Classification is based on the response of a network of processing units (neurons) to an input stimuli (pattern)
    - "Knowledge" is stored in the **connectivity and strength of the synaptic weights**
  - NeurPR is a trainable, non-algorithmic, black-box strategy
  - NeurPR is very attractive since
    - it requires minimum a priori knowledge
    - with enough layers and neurons, an ANN can create **any** complex decision region
- **Syntactic (SyntPR)**
  - Patterns classified based on measures of structural similarity
    - "Knowledge" is represented by means of **formal grammars or relational descriptions** (graphs)
  - SyntPR is used not only for classification, but also for description
    - Typically, SyntPR approaches formulate hierarchical descriptions of complex patterns built up from simpler sub patterns

روشهای شناسایی الگو



طرح یک مساله

- جداسازی اتوماتیک سه نوع میوه : سیب، پرتقال، گلابی
- ویژگیهای قابل بررسی: شکل، زبری و صافی سطح، وزن.
- سنسور ۱ (شکل): تقریبا گرد:۱، تقریبا بیضی: ۱-
- سنسور ۲ (حس بافت سطحی): صاف: ۱، ناصاف: ۱-
- سنسور ۳ (وزن): بیشتر از ۲۰۰گرم: ۱، کمتر از ۲۰۰ گرم: ۱-
- بردارهای متناظر (از چپ به راست):
- سیب=(۱،۱،۱) پرتقال=(۱،۱-،۱) گلابی=(۱، ۱،۱-)
- بنابراین مشخصه وزن در بین همگی مشترک است و حذف می گردد.

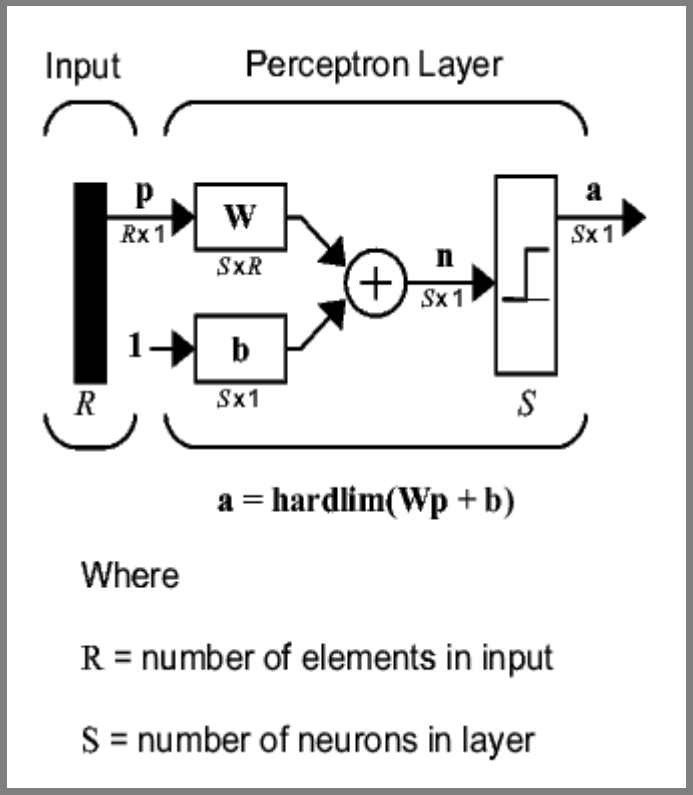
طرح یک مساله

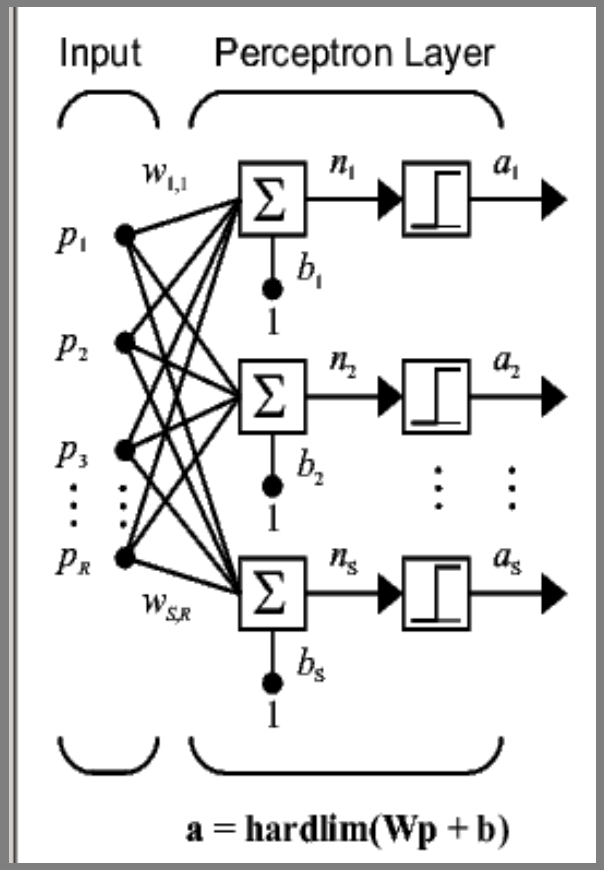
- بنابرانی الگوها را می توان دو بعدی فرض کرد:
- سیب=(۱،۱) پرتقال=(۱-،۱) گلابی=(۱،۱-)
- مساله کلی:
  - یک مساله شناسایی الگوهای باینری داریم.
  - سیستم شناسایی یک شبکه عصبی است که در هر لحظه یک بردار دو بعدی دریافت می کند و در مورد نوع میوه تصمیم گیری میکند.

حل مساله: معرفی سه شبکه عصبی ساده

- شبکه عصبی پرسپترون
  - یک لایه
  - چند لایه (MLP) : از پر کاربردی ترین شبکه های عصبی هستند که قادرند با انتخاب تعداد لایه ها و سلولهای عصبی مناسب که اغلب زیاد هم نیستند، یک نگاشت غیر خطی را با دقت دلخواه انجام دهند.
- شبکه هاپفیلد
- شبکه همینگ

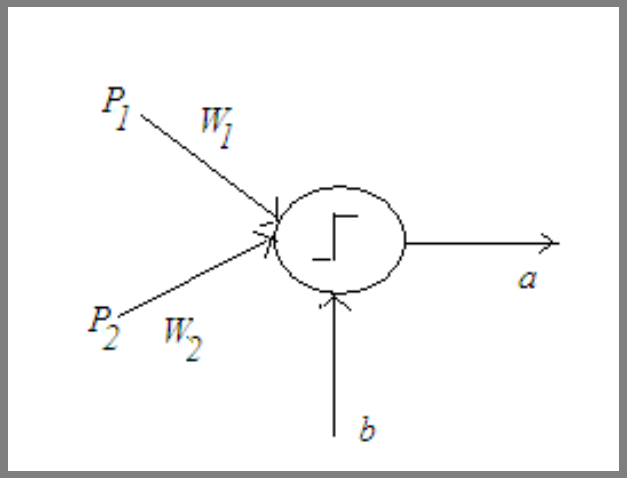
ساختار پرسپترون

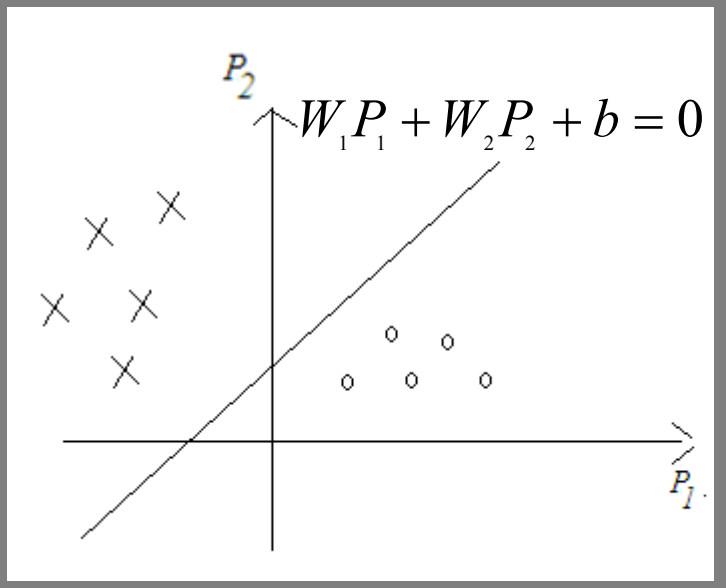




شبکه عصبی پرسپترون یک لایه

- اگر دو الگوی ورودی داشته باشیم، یک پرسپترون می تواند با توجه به بردار ورودی، فضای دو بعدی را به دو قسمت مجزا تفکیک نماید.





شبکه عصبی پرسپترون یک لایه

- اگر بردار ورودی Rتایی باشد، Wیک ماتریس و bیک بردار و مرز جدا کننده یک صفحه خواهد بود با معادله:

$$WP + b = 0$$

- و بنابراین تعداد نرونهای بیشتری نیاز داریم.

حل مساله مطروحه

- ابعاد مساله ۲ است پس Wیک ماتریس۲*۲ و bیک بردار ۱*۲ خواهد بود.
- چون سه الگو داریم پس فضا باید به سه ناحیه تقسیم شود. پس حداقل دو خط لازم داریم. بنابراین به دو نرون نیاز داریم.
- فرض کنیم ناحیه های مطلوب به ترتیب برای سیب، پرتقال و گلابی عبارتند از:

$$a_1 = \begin{bmatrix} -1 \\ 1 \end{bmatrix}, \qquad a_2 = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \qquad a_1 = \begin{bmatrix} -1 \\ -1 \end{bmatrix},$$

- اگر فرض کنیم:

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

حل مساله مطروحه

- برای ورودی سیب:

$$a_1 = WP + b = hard\ \lim\left(\begin{bmatrix}1 & -1\\ 1 & 1\end{bmatrix}\begin{bmatrix}1\\ 1\end{bmatrix} + \begin{bmatrix}-1\\ -1\end{bmatrix}\right) = \begin{bmatrix}-1\\ 1\end{bmatrix}$$

- برای ورودی پرتقال:

$$a_1 = WP + b = hard\ \lim\left(\begin{bmatrix}1 & -1\\ 1 & 1\end{bmatrix}\begin{bmatrix}1\\ -1\end{bmatrix} + \begin{bmatrix}-1\\ -1\end{bmatrix}\right) = \begin{bmatrix}1\\ -1\end{bmatrix}$$

- برای ورودی گلابی:

$$a_1 = WP + b = hard\ \lim\left(\begin{bmatrix}1 & -1\\ 1 & 1\end{bmatrix}\begin{bmatrix}-1\\ 1\end{bmatrix} + \begin{bmatrix}-1\\ -1\end{bmatrix}\right) = \begin{bmatrix}-1\\ -1\end{bmatrix}$$

- برای پرتقال بیضوی:

$$a_1 = WP + b = hard\ \lim\left(\begin{bmatrix}1 & -1\\ 1 & 1\end{bmatrix}\begin{bmatrix}-1\\ -1\end{bmatrix} + \begin{bmatrix}-1\\ -1\end{bmatrix}\right) = \begin{bmatrix}-1\\ -1\end{bmatrix}$$

حل مساله مطروحه

- آیا شبکه فوق برای مسایل مشابه مطلوب است؟ خیر، به دنبال شبکه ای هستیم که با گذشت زمان در نهایت به جواب قابل قبولی همگرا شود.
- نیازمند نوعی قانون بازگشتی هستیم.
- چنین قانونی به قانون یادگیری موسوم است

شبیه سازی پرسپترون با متلب

- Perceptron can be created with the function newp,

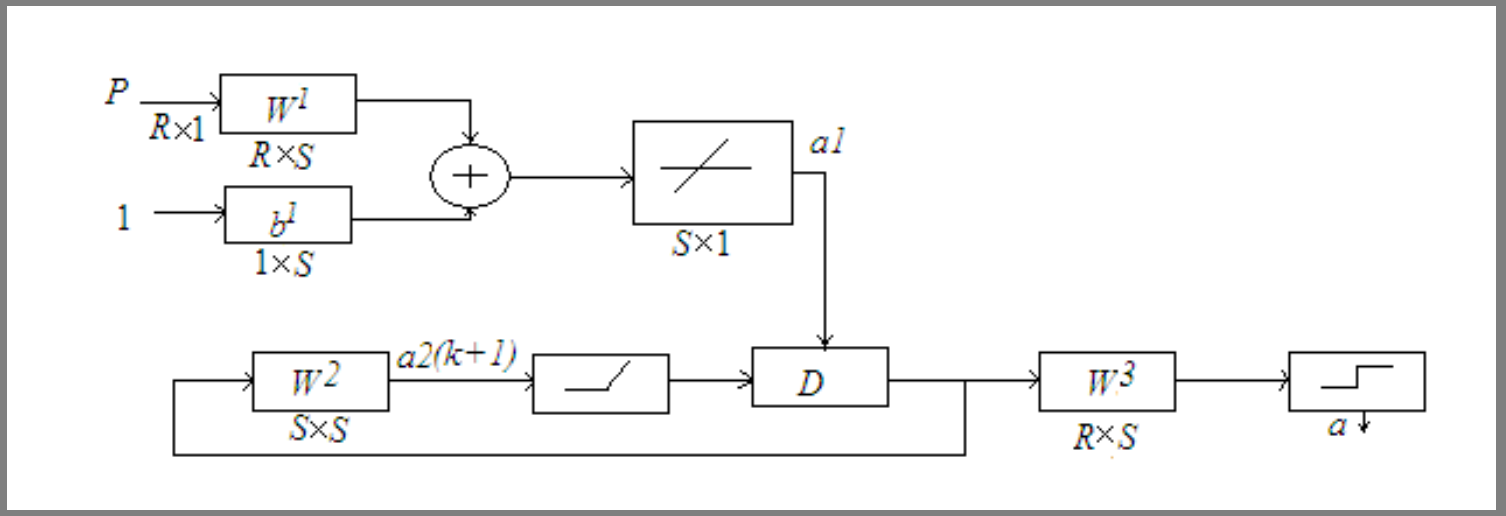
>> net = newp (P,T)

- Where input arguments are as follows:
  - P is an R-by-Q matrix of Q input vectors of R elements each.
  - T is an S-by-Q matrix of Q target vectors of S elements each.

شبکه همینگ

- شبکه همینگ اولین بار توسط اشتاین بوخ در سال ۱۹۶۱ مطرح شد.
- این شبکه اساسا جهت حل مساله شناسایی الگو باینری (الگوهایی که عناصرشان فقط دو مقدار ۱ و ۱- را می پذیرند) بکار برده می شود.
- شبکه همینگ شامل دو ساختار فیدفوروارد و فیدبک ساخته شده است.
- هدف اصلی در این شبکه آنست که تشخیص دهد کدام الگوی مرجع بیشترین نزدیکی را به الگوی ورودی دارد.

ساختار شبکه همینگ



شبکه همینگ

- شبکه همینگ از سه لایه تشکیل شده است:
- لایه فیدفوروارد اول: نخستین لایه که با ماتریس وزن W^1 و بردار بایاس b^1 و تابع تبدیل خطی بیان می شود، همبستگی و یا ضرب داخلی بین بردارهای مرجع با بردار ورودی را محاسبه می کند. بردارهای مرجع شامل الگوهایی است که قصد شناسایی آنها را داریم.
- برای تعیین میزان همبستگی بردار ورودی با بردارهای مرجع ماتریس وزن W^1 توسط بردارهای مرجع ساخته می شود:

$$W^1 = \begin{bmatrix} p_1^T \\ p_2^T \\ p_3^T \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \\ -1 & 1 \end{bmatrix}$$

شبکه همینگ–لایه اول

- هدف در اینجا آنست که اگر یکی از الگوها در ورودی داده شود، خروجی همان الگو باشد، اما اگر ورودی اختیاری اعمال کردیم، شبکه پاسخ مناسبی بدهد.
- باید معیاری از نزدیکی ورودی به الگوها داشته باشیم.
- به چنین شبکه هایی شبکه های حافظه ای خود انجمنی (Autoassociative memory) گفته می شود.
- پروسه محاسباتی در شبکه های حافظه ای خود انجمنی:
  - ذخیره سازی الگوها
  - بازیابی اطلاعات (الگوهای) مربوطه از الگوهای ذخیره شده

شبکه همینگ–لایه اول

- فاصله همینگ:

$$P_H^j = \frac{1}{2}\sum_{i=1}^{R}\left|P_i^j - P_i\right|, \qquad P, P^j \in \{-1,1\}^R$$

- که در آن، P^j بردار مرجع؛ P_i عنصر i ام از بردار P و R تعداد عناصر بردار ورودی است. داریم:

$$P_H^j = \frac{1}{2}\sum_{i=1}^{R}\left|P_i^j\right|\left|1-(P_i^j)^{-1}P_i\right| = \frac{1}{2}\sum_{i=1}^{R}(1-P_i^jP_i) = \frac{R}{2} - \frac{1}{2}\sum_{i=1}^{R}(P_i^jP_i)$$

$$P_H^j = \frac{R}{2} - \frac{1}{2} < P^j, P >$$

- که در آن $<P^j, P>$ نماد ضرب داخلی دو بردار است

شبکه همینگ–لایه اول

- بر اساس فاصله همینگ، شباهت دو بردار بصورت زیر تعریف می شود:

$$C^i = R - kP_H^j, \quad k > 0$$

- که در آن C^i با افزایش P_H^j، کاهش و با افزایش P_H^j، کاهش می یابد. اگر $k=1$ باشد:

$$C^i = \frac{R}{2} + \frac{1}{2}\sum_{i=1}^{R} P_i^j P_i = \frac{R}{2} + \sum_{i=1}^{R}\left(\frac{P_i^j}{2}\right) P_i,$$

- یعنی شبکه ای از نرونها که وزنهاو بایاسش برابر خواهد بود با:

$$W_{j,i}^1 = \frac{P_i^j}{2}, \quad b_j^1 = \frac{R}{2}$$

- پس به تعداد الگوهای مرجع نرون لازم داریم.

شبکه همینگ

- برای شبکه همینگ

$$W^1 = [P_1 \quad P_2 \quad \dots \quad P_3]^T_{S\times R},$$
$$b^1 = [R \quad R \quad \dots \quad R]^T_{S\times 1}$$

- برای مساله شناسایی میوه، خروجی لایه اول برابر است با:

$$a^1 = W^1 P + b = \begin{bmatrix} P_1^T P + R \\ P_2^T P + R \\ P_3^T P + R \end{bmatrix}$$

- تمامی عناصر بردار خروجی لایه ۱ بین ۰ و $2R$ خواهد بود.
- باید در لایه دوم بیشترین شباهت انتخاب شود.

شبکه همینگ–لایه برگشتی

- تکنیک بکار رفته: WTA (Winner Take All)
- خروجی با بیشترین شباهت را از بقیه جدا کنیم.
- لایه میانی شبکه همینگ، یک شبکه رقابتی است.
- در شبکه میانی، خروجی لایه اول بعنوان مقادیر اولیه ورودی در نظر گرفته میشود.

$$a_j^2(0) = c^j$$

- در هر دور فیدبک از مقدار a_j^2 کم می شود تا همه بجز یکی صفر شوند. سلول برنده مبین بیشترین الگوی مرجع مربوطه به ورودی می باشد.
- در این حالت شبکه همینگ به تعادل رسیده است.

رفتار لایه میانی شبکه همینگ

$$a^2(0) = a^1$$

$$a^2(k+1) = posl\ (W^2 a^2(k)),$$

$$posl\ (x) = \begin{cases} x & x \geq 0 \\ 0 & x < 0 \end{cases}$$

$$W_{ij}^2 = \begin{cases} 1 & i = j \\ -\varepsilon & i \neq j \end{cases}, \qquad 0 < \varepsilon < 1,$$

$$\varepsilon = \frac{1}{S-1}$$

مثال– شناسایی میوه

- برای پرتقال بیضوی

$$P = \begin{bmatrix} -1 \\ -1 \end{bmatrix} \rightarrow a^2(0) = a^1 = \begin{bmatrix} 0 \\ 2 \\ 2 \end{bmatrix}$$

$$W^2 = \begin{bmatrix} 1 & -\varepsilon & -\varepsilon \\ -\varepsilon & 1 & -\varepsilon \\ -\varepsilon & -\varepsilon & 1 \end{bmatrix} \qquad \varepsilon = \frac{1}{3-1} = 0.5$$

$$a^2(1) = posl\ (W^2 a^2(0)) = posl\ \left(\begin{bmatrix} -2 \\ 1 \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}$$

$$a^2(2) = posl\ (W^2 a^2(1)) = posl\ \left(\begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0.5 \\ 0.5 \end{bmatrix}$$

- برای سیب

$$a^2(2) = \begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}$$

لایه سوم شبکه همینگ

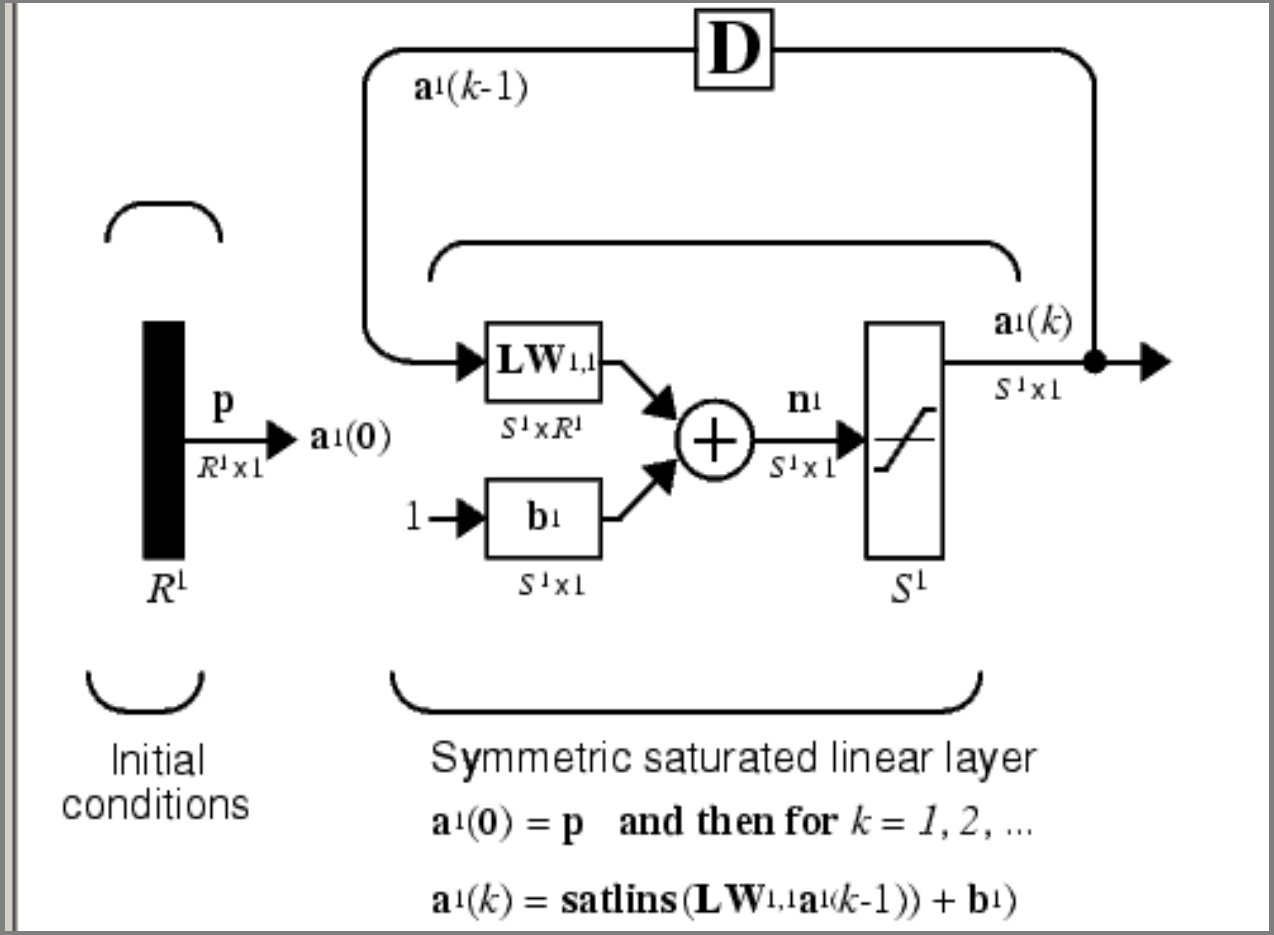
- لایه سوم، یک لایه فیدفوروارد با ماتریس وزن و تابع تبدیل آستانه ای دومقداره است.
- وظیفه لایه سوم آنست که پس از همگرایی خروجی لایه دوم، بردار الگوی متناظر در خروجی شبکه ظاهر شود.
- جهت این کار وزنهای لایه سوم به ترتیب زیر انتخاب می شوند:

$$W^{3}_{i,j} = P^{i}_{j}$$

- اگر هدف ارتباط دادن زوج (P^{i}_{j}, Q^{i}_{j}) به هم باشد، وزنهای لایه سوم عبارتند از:

$$W^{3}_{i,j} = Q^{i}_{j}$$

ساختار شبکه هاپفیلد

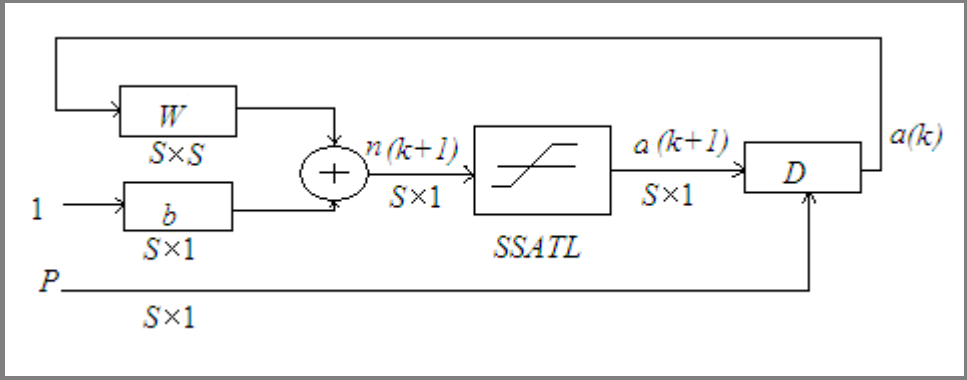


شبکه هاپفیلد

- تقریبا شبیه لایه میانی شبکه هاپفیلد عمل می کند.
- فرقی نمیکند ورودی به کجا اعمال شود، ورودی تنها بعنوان شرط اولیه عمل می کند و سپس شبکه تا همگرایی خود را تکرار می کند.

$$a(k+1) = ssatl\ (Wa(k) + b))$$

$$a(0) = P$$



- مشکل: ممکن است به مقداری بجز یکی از الگوها همگرا شود.

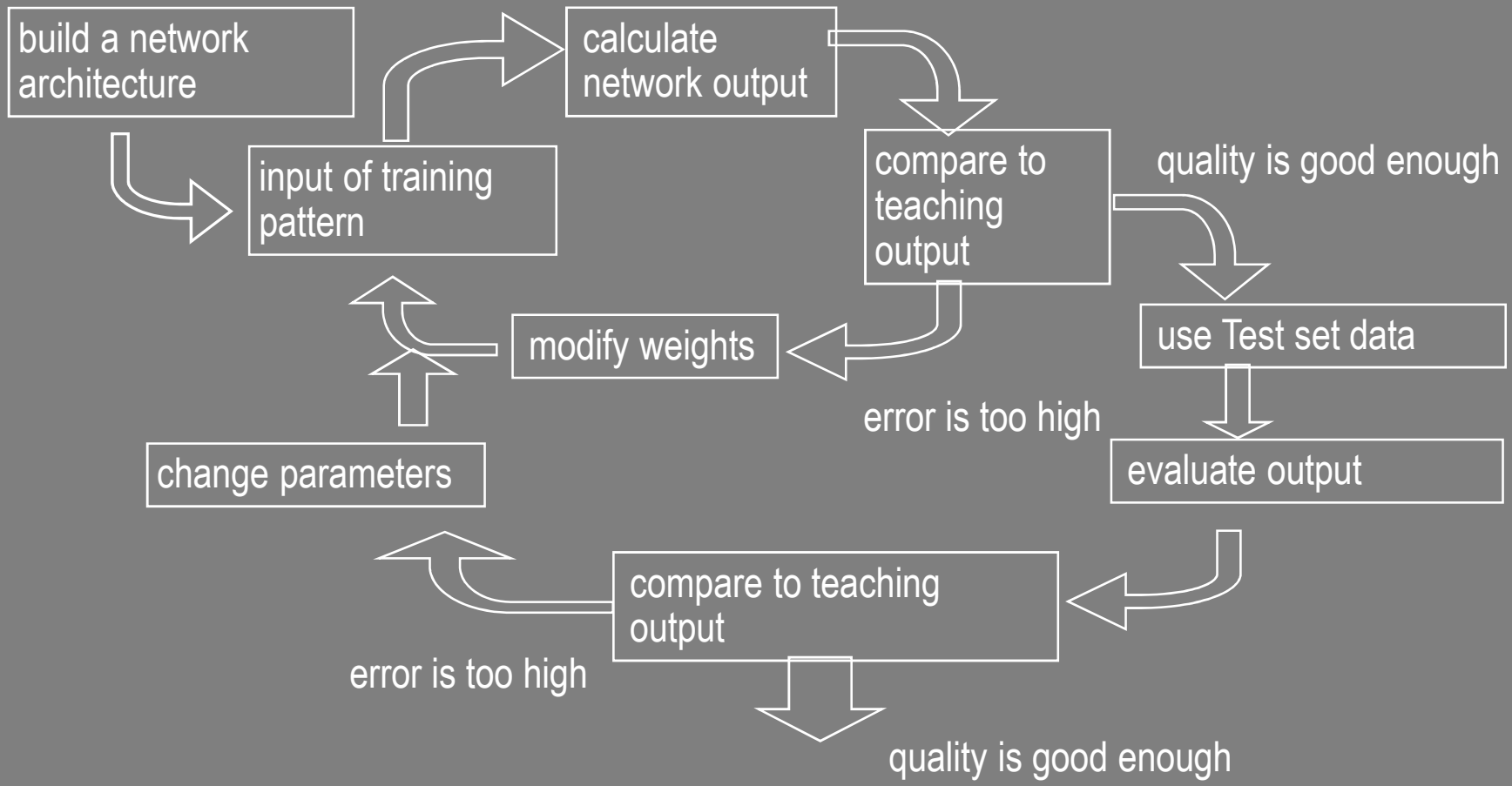
فرآیند یادگیری در شبکه های عصبی

- مبنای فرآیند یادگیری در انسان:
  - هیچ چیز بدون دلیل اتفاق نمی افتد
  - مارگزیده از ریسمان سفید و سیاه می ترسد
  - از علل مشابه انتظار عواقب مشابه داریم
- ضرورت یادگیری: اگر همه مقاصد و اهداف مشخص باشد، نیازی به یادگیری نداریم. زمانی به پروسه یادگیری نیاز است که اطلاعات کامل در مورد اهداف موجود نباشد.
- رفتار سیستمهای یادگیر () توسط الگوریتمهای بازگشتی بیان می شود.
- قوانین یادگیری اغلب بصورت دینامیکی توسط معادلات تفاضلی (دیفرانسیلی بیان می شود.

فرآیند یادگیری در شبکه های عصبی

- الگوریتمهای یادگیری روی اطلاعات موجود آنگونه پردازش می کند که شاخص اجرایی معلومی بهینه گردد.
- **برای انجام عملیات یادگیری، غنی بودن داده ها شرط اساسی است.**
- فرآیند یادگیری:
  - سیستم یادگیرنده توسط محیط تحریک می شود.
  - قانون یادگیری با رجوع به نتیجه تحریک، پارامترهای سیستم یادگیری را تغییر می دهد.
  - سیستم یادگیرنده به خاطر تغییرات داخلی صورت گرفته پاسخ مناسبتری به محیط می دهد.
  - عملیات فوق تکرار می شود.

Development of an NN-application



فرآیند یادگیری در شبکه های عصبی

- ابزار آموزش:
- داده های مناسب شامل داده های آموزش و داده های ارزیابی
- الگوریتم یادگیری مناسب
  - دقت و کارآیی
  - سرعت همگرایی بالا
- در شبکه های عصبی منظور از یادگیری تنظیم ماتریس وزنها و بردارهای بایاس به گونه ای است که
  - برای داده های آموزش، پاسخ با دقت بالایی به خروجی مورد نظر نزدیک باشد.
  - برای داده های تست، خروجی مناسبی حاصل شود.

معادلات یادگیری در حالت کلی

- اگر بردار های معرف نرون و ورودی بصورت زیر تعریف شود:

$$W = [W_1, W_2, \dots, W_R, b]^T, \qquad P = [P_1, \dots, P_R, 1]^T$$

- محیط یا منابع اطلاعاتی را می توان بصورت زیر مدل کرد:
  - یادگیری تحت نظارت $\{P, T, prob(P, T)\}$
  - یادگیری بدون نظارت $\{P, prob(P, S)\}$

که در آن P بردار ورودی، T بردار خروجی، $prob$ تابع توزیع احتمال و S بردار حالت نرون است.

- در این صورت قانون یادگیری را می توان بصورت زیر در نظر گرفت:

$$\dot{W} = -\alpha W + \eta l p$$
$$W(k+1) = (1-\alpha)W(k) + \eta l p(k)$$

که در l آن سیگنال یادگیری

و ضرایب α و η ثوابت کوچکی هستند

یادگیری شبکه

- در یک شبکه هر نرون بردابردار وزنی خودش را مطابق با قانون یادگیری خود تغییر می دهد، اما با توجه به رفتار سایر نرونها. داریم:

$$\dot{W}_{ij} = -\alpha) W_{ij} + \Delta W_{ij}$$
$$W_{ij}(k+1) = (1-\alpha) W_{ij}(k) + \Delta W_{ij}(k)$$

- ΔW_{ij} ترم اصلاحی مربوط به وزن متناظر آن می باشد که باید تعیین شود.

یادگیری با ناظر

- در یادگیری با ناظر، مجموعه ای از زوجهای داده به نام داده های آموزش شامل ورودیها و خروجیهای مطلوب به شبکه اعمال می شوند.
- اختلاف خروجی شبکه با خروجی مطلوب، جهت تنظیم پارامترهای شبکه بکار میرود، بطوریکه اگر دفعه بعد ورودی به شبکه اعمال شود پاسخ بهتری دریافت شود.
- قانون یادگیری پرسپترون، الگوریتم LMS و پس انتشار خطا نمونه ای از این الگوریتمها هستند.

آموزش پرسپترون

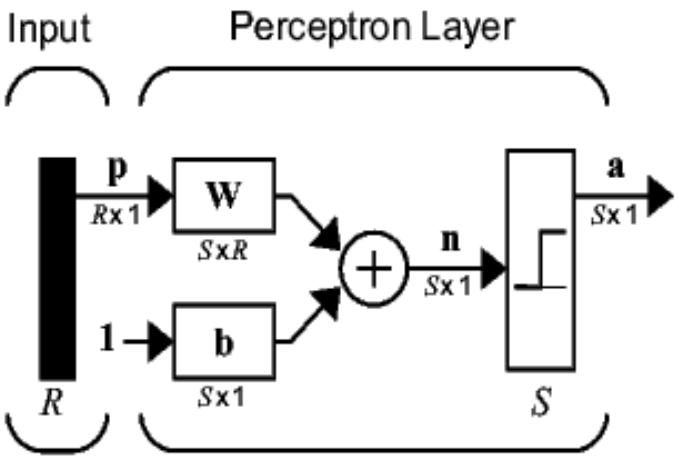
- چگونه وزنهای یک پرسپترون واحد را آموزش دهیم به نحوی که پرسپترون برای مثالهای آموزشی مقادیر صحیح را ایجاد نماید؟
- دو راه مختلف:
  - قانون پرسپترون
  - قانون دلتا

یادگیری در شبکه های پرسپترون تک لایه
Single Layer Perceptron Rule (SLPR)

- این نوع یادگیری از نوع یادگیری با ناظر است.
- محرک، پاسخ درست، ورودی و خروجی مطلوب، الگو و اینکه الگو به چه طبقه ای متعلق است در اختیار می باشد.
- در هر مرحله، از خطای یادگیری جهت تنظیم پارامترهای شبکه طوری استفاده می شود که در صورت اعمال دوباره همان ورودی خطا کمتر شود.
- یادگیری SLPR فقط برای شبکه های پرسپترون تک لایه با تابع تبدیل آستانه ای حدی بکار می رود.

SLPR

Let S=1 R=2

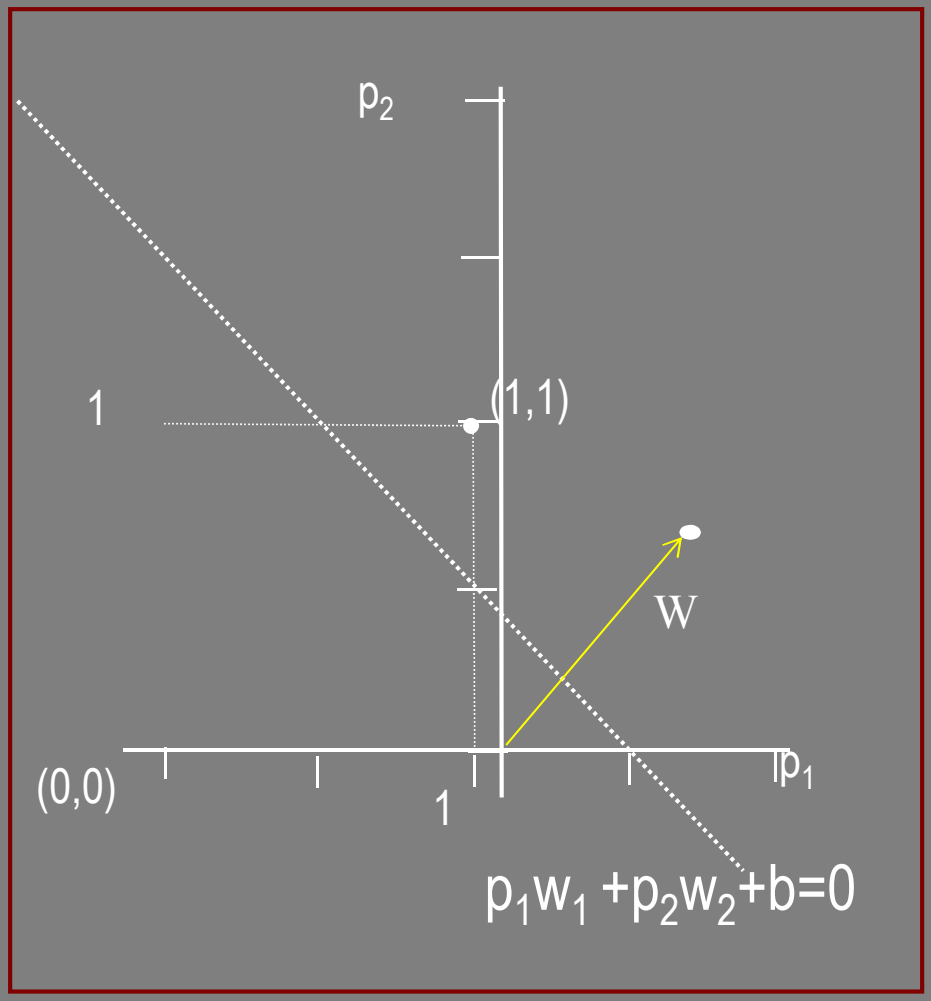


$$\mathbf{a} = \mathbf{hardlim}(\mathbf{Wp} + \mathbf{b})$$

Where

R = number of elements in input

S = number of neurons in layer



SLPR

- اگرS برابر ۲ یا بیشتر شود، محل برخورد خطوط فضا را به ۲^S قسمت تقسیم می کند.
- برای هر قسمت یک خروجی باینا‌ری مطلوب در نظر می گیریم.
- مثال: فرض کنید هدف جدا کردن ورودیهای (۰، ۲) و (۲، ۱) از ورودیهای (۰، ۱-) و (۲، ۲-) باشد.
- واضح است که به یک نرون بیشتر احتیاج نداریم. برای دو ورودی اول خروجی ۱ و برای دو ورودی دوم خروجی ۱- را در نظر می گیریم.
- فرض کنیم ترم بایاس را صفر و وزن را بصورت (۱، ۱-) انتخاب کنیم.
- معادله خط مرزی عبارت است از: $-p_1 + p_2 = 0$

SLPR

$$Y=-p_1+p_2$$

- مشاهده می شود که برای ورودی (۲، ۱) خروجی مثبت (مطلوب) و برای (۰، ۲) خروجی منفی (نا مطلوب) است.
- پس وزنها باید تغییر کنند طوری که به سمت (۰، ۲) نشانه رود.
- چون خط جدا کننده بر بردار w عمود است اگر، wرا برابر (۰، ۲) بگذاریم مطمئن هستیم جواب ۱ خواهد شد (نشانه روی کامل).
- اما اینکار باعث غیر همگرایی می شود، پس وزن قبلی را با (۰، ۲) جمع می کنیم، خروجی مثبت خواهد شد.

$$Y=p_1+p_2$$

SLPR

- برای ورودی (۲، ۲-) مقدار خروجی جدید ۰ و طبق منطق تابع آستانه ای خروجی ۱ می شود که مطلوب نیست.
- پس باید در جهت کاهش خروجی یا خلاف جهت بردار (۲، ۲-) حرکت کنیم. وزن قبلی را با منفی بردار (۲، ۲-) جمع میکنیم.

$$Y=3p_1 - p_2$$

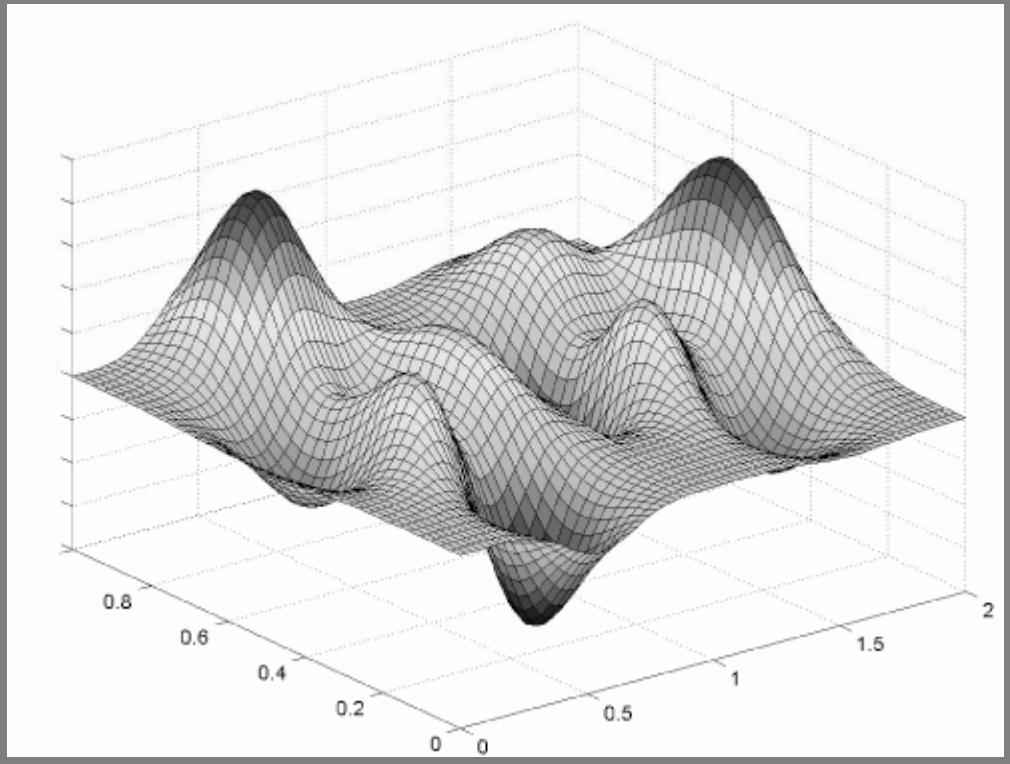
- در این حالت خروجی برای تمام ورودیها پاسخ درست خواهد داد.

آموزش پرسپترون

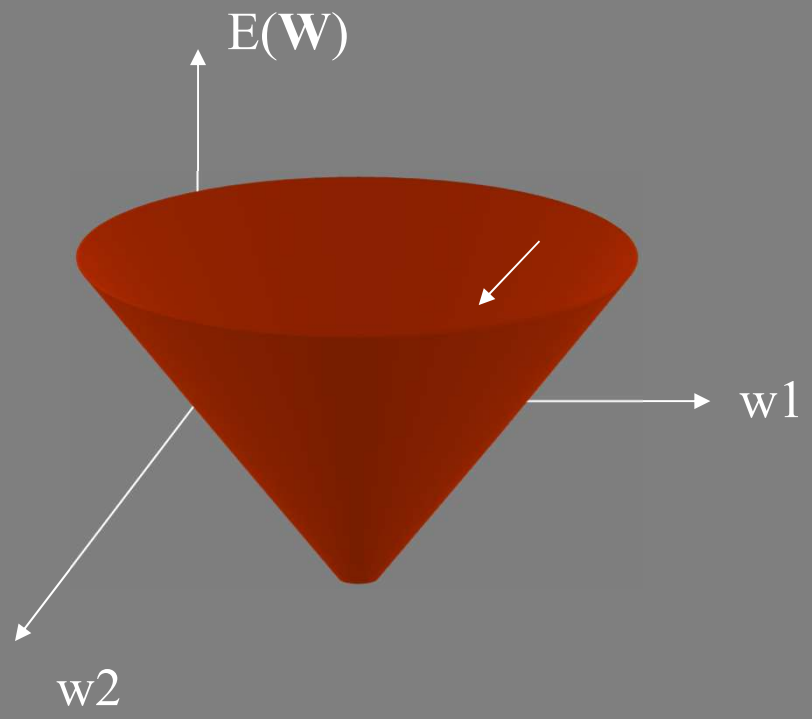
الگوریتم یادگیری پرسپترون

1. مقادیری تصادفی به وزنها نسبت میدهیم
2. پریسپترون را به تک تک مثالهای آموزشی اعمال میکنیم .اگر مثال غلط ارزیابی شود مقادیر وزنهای پرسپترون را تصحیح میکنیم.
3. آیا تمامی مثالهای آموزشی درست ارزیابی میشوند:
   - بله ← پایان الگوریتم
   - خیر←به مرحله 2 برمیگردیم

Error Curve



الگوریتم gradient descent



- با توجه به نحوه تعریف E سطح خطا بصورت یک سهمی خواهد بود .ما بدنبال وزنهائی هستیم که حداقل خطا را داشته باشند .
- الگوریتم gradient descent در فضای وزنها بدنبال برداری میگردد که خطا را حداقل کند.
- .این الگوریتم از یک مقدار دلخواه برای بردار وزن شروع کرده و در هر مرحله وزنها را طوری تغییر میدهد که در جهت شیب کاهشی منحنی فوق خطا کاهش داده شود.

بدست آوردن قانون gradient descent

- ایده اصلی :گرادیان همواره در جهت افزایش شیب E عمل میکند.
- گرادیان E نسبت به بردار وزن w بصورت زیر تعریف میشود:

$$\nabla E(W) = [\, E'/w0,\ E'/w1,\ \ldots,\ E'/wn]$$

- که در آن $\nabla E(W)$ یک برداروE’ مشتق جزئی نسبت به هر وزن میباشد.

قانون دلتا Delta Rule

- برای یک مثال آموزشی $X = (x_1, x_2, \ldots, x_n)$ در هر مرحله وزنها بر اساس قانون دلتا بصورت زیر تغییر میکند:

$$w_i(k+1) = w_i(k) + \Delta w_i$$
$$b_i(k+1) = b_i(k) + \Delta b_i$$

که در آن

Where $\Delta w_i = -\eta E'(\mathbf{W})/wi$

η: learning rate (e.g., 0.1)

علامت منفی نشاندهنده حرکت در جهت کاهش شیب است.

محاسبه گرادیان

- با مشتق گیری جزئی از رابطه خطا میتوان بسادگی گرادیان را محاسبه نمود:

$$E'(W)/\ w_i = \Sigma_i\ (t_i - o_i)\ (-x_i)$$

- لذا وزنها طبق رابطه زیر تغییر خواهند نمود.

$$\Delta w_i = \eta\ \Sigma i\ (t_i - o_i)\ x_i$$

خلاصه یادگیری قانون دلتا

الگوریتم یادگیری با استفاده از قانون دلتا بصورت زیر میباشد.

1. به وزنها مقدار تصادفی نسبت دهید.
2. تا رسیدن به شرایط توقف مراحل زیر را ادامه دهید
   - هر وزن ∇w_i را با مقدار صفر عدد دهی اولیه کنید.
   - برای هر مثال :وزن ∇w_i را بصورت زیر تغییر دهید:

$$\nabla w_i = \nabla w_i + \eta (t - o) x_i$$

مقدار w_i را بصورت زیر تغییر دهید:

$$w_i = w_i + \nabla w_i$$

تا خطا بسیار کوچک شود.

مشکلات روش gradient descent

1. ممکن است همگرا شدن به یک مقدار مینیمم زمان زیادی لازم داشته باشد.
2. اگر در سطح خطا چندین مینیمم محلی وجود داشته باشد تضمینی وجود ندارد که الگوریتم مینیمم مطلق را پیدا بکند.

در ضمن این روش وقتی قابل استفاده است که:

- فضای فرضیه دارای فرضیه های پارامتریک پیوسته باشد.
- رابطه خطا قابل مشتق گیری باشد

تقریب افزایشی gradient descent

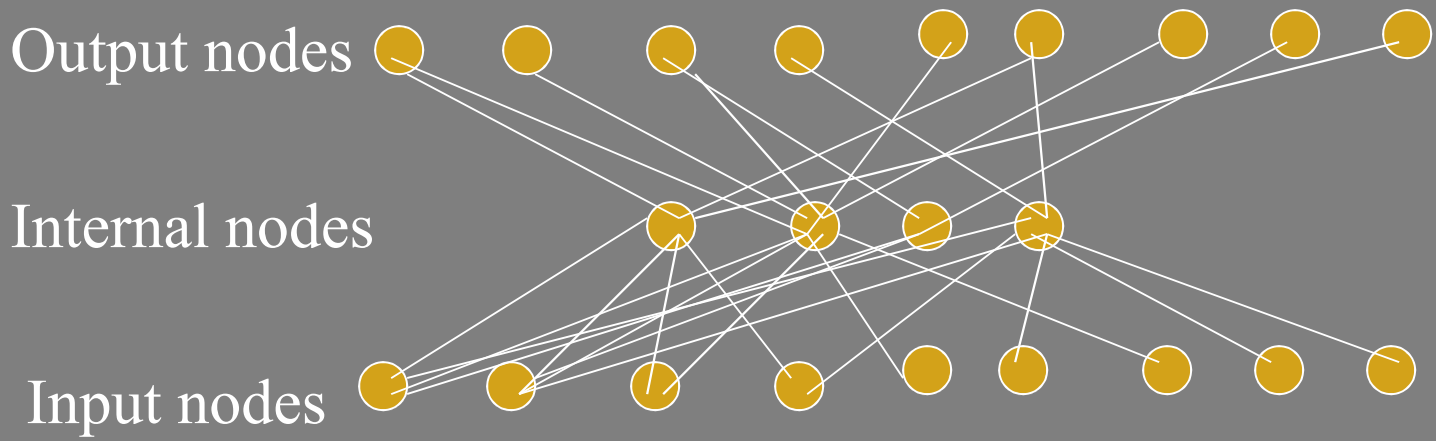
- میتوان بجای تغییر وزنها پس از مشاهده همه مثالها، آنها را بازا هر مثال مشاهده شده تغییر داد .در این حالت وزنها بصورت افزایشی incrementalتغییر میکنند .این روش را stochastic gradient descent نیزمینامند.

$$\nabla w_i = \eta (t-o) x_i$$

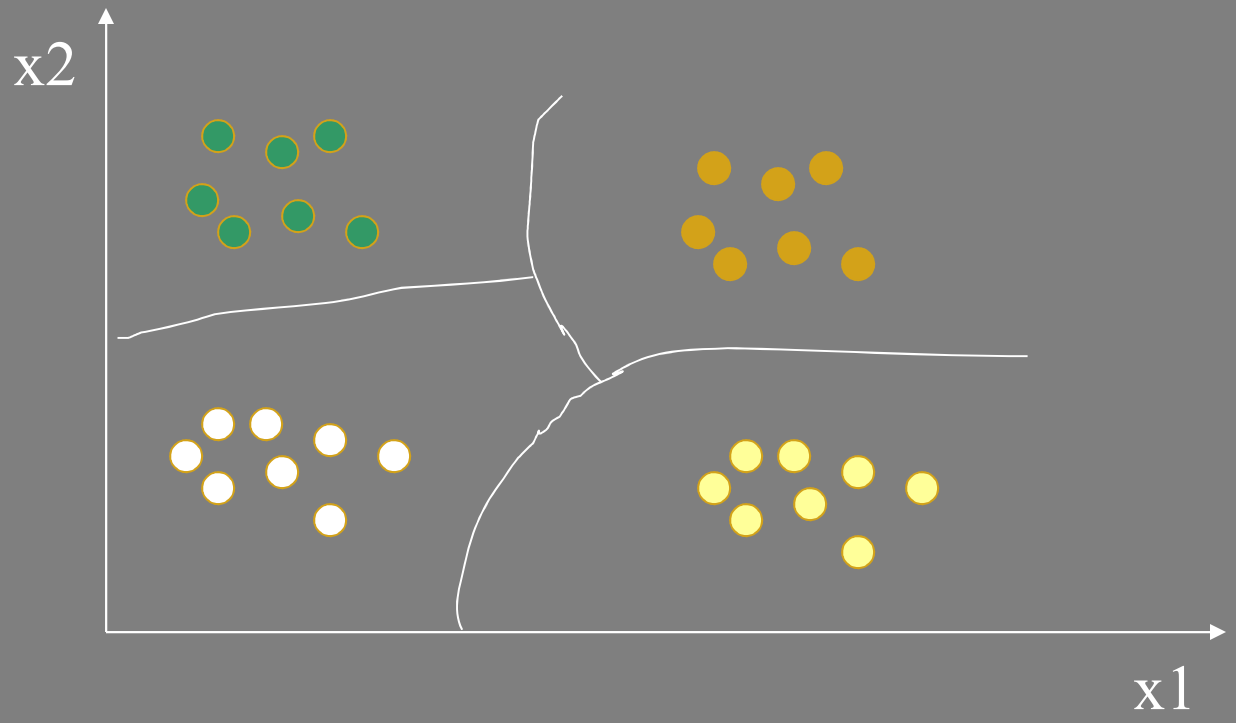
در بعضی موارد تغییر افزایشی وزنها میتواند از بروز مینیمم محلی جلوگیری کند .روش استاندارد نیاز به محاسبات بیشتری دارد درعوض میتواند طول step بزرگتری هم داشته باشد.

شبکه های چند لایه

بر خلاف پرسپترونها شبکه های چند لایه میتوانند برای یادگیری مسائل غیر خطی و همچنین مسائلی با تصمیم گیری های متعدد بکار روند.

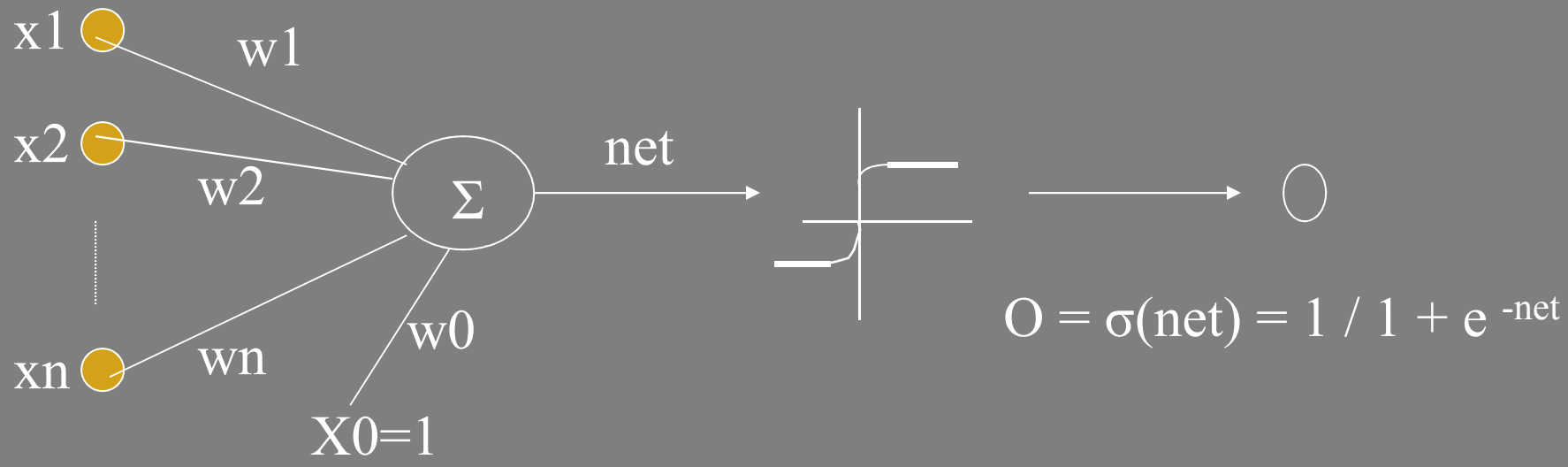


مثال



یک سلول واحد

برای اینکه بتوانیم فضای تصمیم گیری را بصورت غیر خطی از هم جدا بکنیم، لازم است تا هر سلول واحد را بصورت یک تابع غیر خطی تعریف نمائیم .مثالی از چنین سلولی میتواند یک واحد سیگموئید باشد:



تابع سیگموئید

خروجی این سلول واحد را بصورت زیر میتوان بیان نمود:

$$O(x1,x2,\ldots,xn) = \sigma (WX)$$

where: $\sigma (WX) = 1 / 1 + e^{-WX}$

تابع σ تابع سیگموئید یا لجستیک نامیده میشود .این تابع دارای خاصیت زیر است:

$$d\,\sigma(y) / dy = \sigma(y)\ (1 - \sigma(y))$$

الگوریتم Back propagation

- برای یادگیری وزن های یک شبکه چند لایه از روش Back Propagationاستفاده میشود .در این روش با استفاده از gradient descentسعی میشود تا مربع خطای بین خروجی های شبکه و تابع هدف مینیمم شود. ← یادگیری تحت نظارت
- خطا بصورت زیر تعریف میشود:

$$E(\vec{W}) \equiv \frac{1}{2} \sum_{i \in I} \sum_{k \in outputs} \left(t_{ki} - o_{ki} \right)^2$$

مراد از outputsخروجیهای مجموعه واحد های لایه خروجی و t_{ki} و o_{ki} مقدار هدف و خروجی متناظر با k امین واحد خروجی و مثال i ام آموزشی است.

الگوریتم Back propagation

- فضای فرضیه مورد جستجو در این روش عبارت است از فضای بزرگی که توسط همه مقادیر ممکن برای وزنها تعریف میشود .
- روش gradient descent سعی میکند تا با مینیمم کردن خطا به فرضیه مناسبی دست پیدا کند .اما تضمینی برای اینکه این الگوریتم به مینیمم مطلق برسد وجود ندارد.

الگوریتم BP

1. شبکه ای با n_{in} گره ورودی، n_{hidden} گره مخفی، و n_{out} گره خروجی ایجاد کنید .
2. همه وزنها را با یک مقدار تصادفی کوچک عدد دهی کنید.
3. تا رسیدن به شرط پایانی (کوچک شدن خطا)مراحل زیر را انجام دهید:

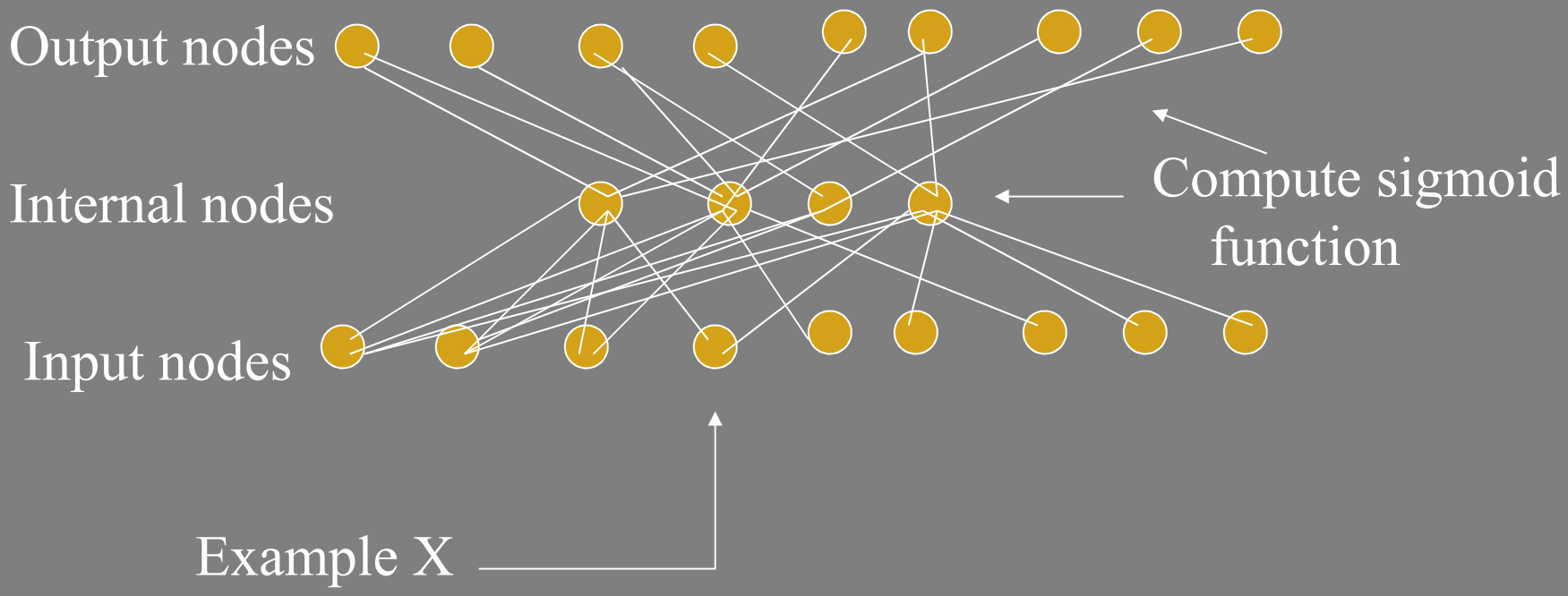
برای هر X متعلق به مثالهای آموزشی:

مثال X را به سمت جلو در شبکه انتشار دهید

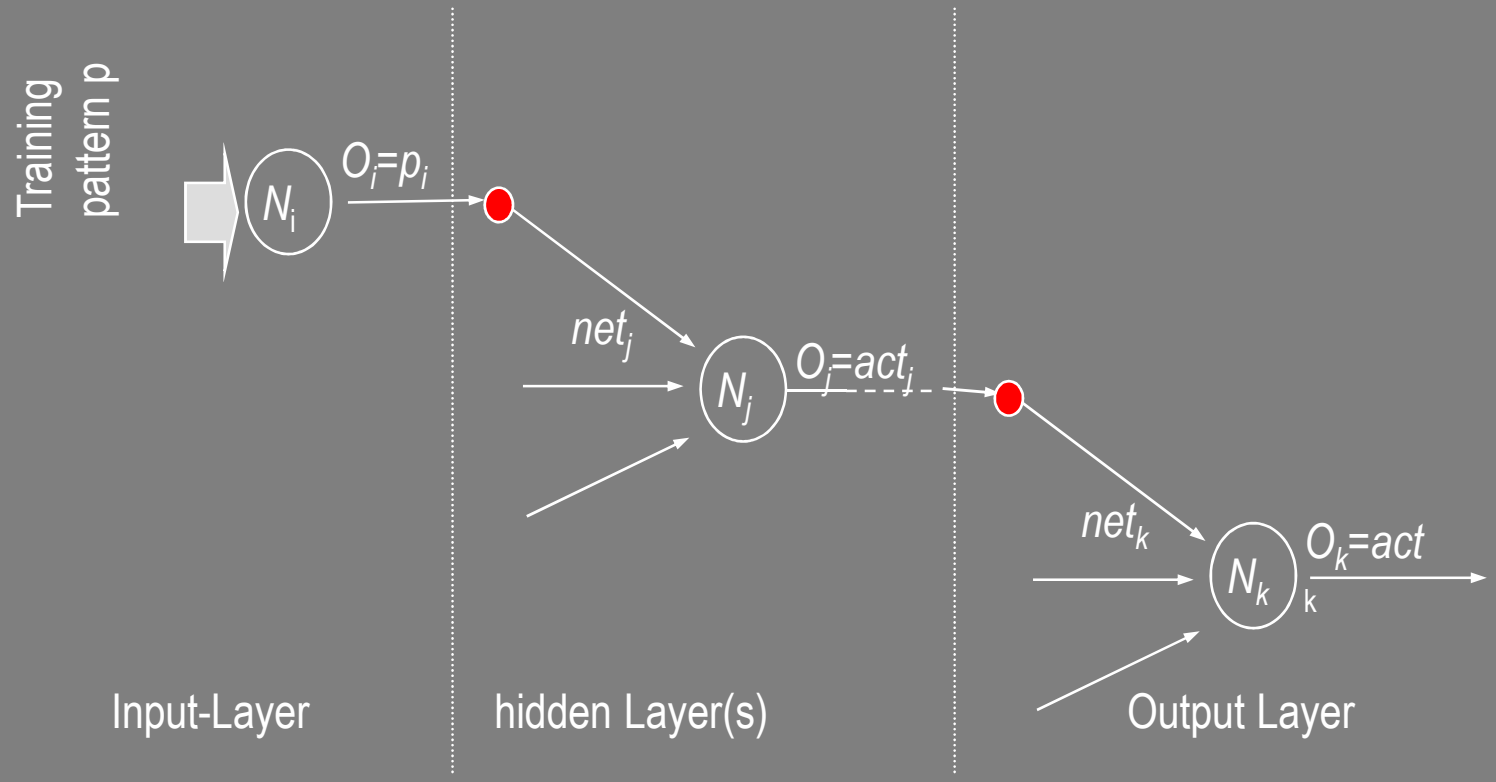
خطای E را به سمت عقب در شبکه انتشار دهید.

انتشار به سمت جلو

- برای هر مثال X مقدار خروجی هر واحد را محاسبه کنید تا به گره های خروجی برسید.



محاسبه خروجی در یک شبکه پیش خور



محاسبه خروجی در یک شبکه پیش خور

- مقادیر گره‌های لایه ورودی عموماً بدون تغییر و یا حداکثر پس از نرمالیزه شدن در بردارهای وزن بین این لایه و لایه بعدی (لایه پنهان) ضرب می‌شوند و حاصل آن به عنوان ورودی لایه بعدی محسوب می‌گردد.

$$net_j = \sum_i W_{ji} U_i$$

- مقادیر بدست آمده پس از عبور از تابع عملکرد f، مقدار خروجی آن لایه را بدست می‌دهند.

$$O_j = f(net_j)$$

محاسبه خروجی در یک شبکه پیش خور

- مقادیر گره‌های لایه ورودی عموماً بدون تغییر و یا حداکثر پس از نرمالیزه شدن در بردارهای وزن بین این لایه و لایه بعدی (لایه پنهان) ضرب می‌شوند و حاصل آن به عنوان ورودی لایه بعدی محسوب می‌گردد.

$$net_j = \sum_i W_{ji} U_i$$

- مقادیر بدست آمده پس از عبور از تابع عملکرد f، مقدار خروجی آن لایه را بدست می‌دهند.

$$O_j = f(net_j)$$

محاسبه خروجی در یک شبکه پیش خور

- ورودی هر گره از لایه k ام برابر است با جمع وزنی تمامی خروجی‌های لایه پیشین:

$$net_k = \sum_j W_{kj} O_j$$

- و خروجی گره k ام برابر است با:

$$O_k = f(net_k)$$

انتشار به سمت عقب

1. برای هر واحد خروجی جمله خطا را بصورت زیر محاسبه کنید:

$$\delta_k = O_k (1-O_k)(t_k - O_k)$$

2. برای هر واحد مخفی جمله خطا را بصورت زیر محاسبه کنید:

$$\delta_h = O_h (1-O_h) \Sigma_k W_{kh} \delta_k$$

3. مقدارهر وزن را بصورت زیر تغییر دهید: $W_{ji} = W_{ji} + \Delta W_{ji}$

$$\Delta W_{ji} = \eta \delta_j X_{ji}$$

که در آن:

η عبارت است از نرخ یادگیری

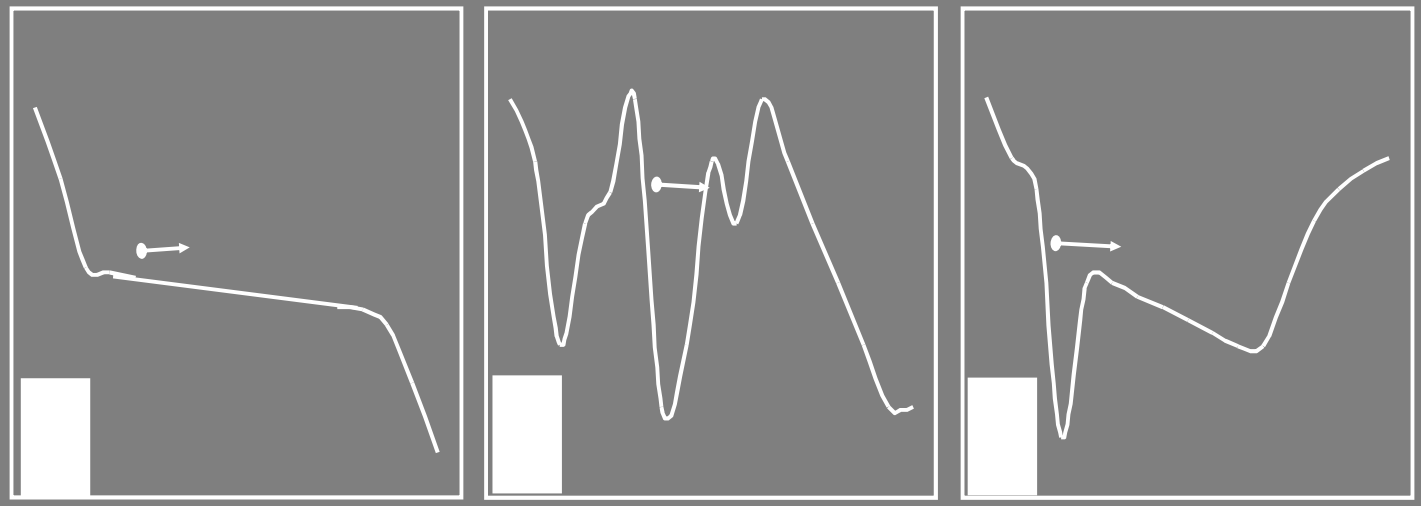
شرط خاتمه

- معمولا الگوریتم BP پیش از خاتمه بارها با استفاده همان داده های آموزشی تکرار میگردد شروط مختلفی را میتوان برای خاتمه الگوریتم بکار برد:
  - توقف بعد از تکرار به دفعات معین (epocs)
  - توقف وقتی که خطا از یک مقدار تعیین شده کمتر شود.

  - net.trainParam.epochs = 50;
  - net.trainParam.goal = 0.01;

  - توقف وقتی که خطا در مثالهای مجموعه تائید از قاعده خاصی پیروی نماید.
  - اگر دفعات تکرار کم باشد خطا خواهیم داشت و اگر زیاد باشد مسئله Overfitting رخ خواهد داد.

Backpropagation - Problems



Backpropagation - Problems

- A: flat plateau
  - weight adaptation is slow
  - finding a minimum takes a lot of time
- B: Oscillation in a narrow gorge
  - it jumps from one side to the other and back
- C: leaving a minimum
  - if the modification in one training step is to high, the minimum can be lost

مرور الگوریتم BP

- این الگوریتم یک جستجوی gradient descent در فضای وزنها انجام میدهد.
- ممکن است در یک مینیمم محلی گیر بیافتد
- در عمل بسیار موثر بوده است

برای پرهیز از مینیمم محلی روشهای مختلفی وجود دارد:

– افزودن ممنتم
– استفاده ازstochastic gradient descent
– استفاده ازشبکه های مختلف با مقادیر متفاوتی برای وزنهای اولیه

افزودن ممنتم

- میتوان قانون تغییر وزنها را طوری در نظر گرفت که تغییر وزن در تکرار nام تا حدی به اندازه تغییروزن در تکرار قبلی بستگی داشته باشد.

$$\Delta W_{ji}(n) = \underbrace{\eta\, \delta_j\, X_{ji}}_{\text{قانون تغییر وزن}} + \underbrace{\alpha \Delta W_{ji}(n-1)}_{\text{عبارت ممنتم}}$$

که در آن مقدارممنتم α بصورت $0 <= \alpha <= 1$ میباشد.

افزودن ممنتم باعث میشود تا با حرکت در مسیر قبلی در سطح خطا:

- از گیر افتادن در مینیم محلی پرهیز شود
- از قرارگرفتن در سطوح صاف پرهیز شود
- با افزایش تدریجی مقدار پله تغییرات، سرعت جستجو افزایش یابد.

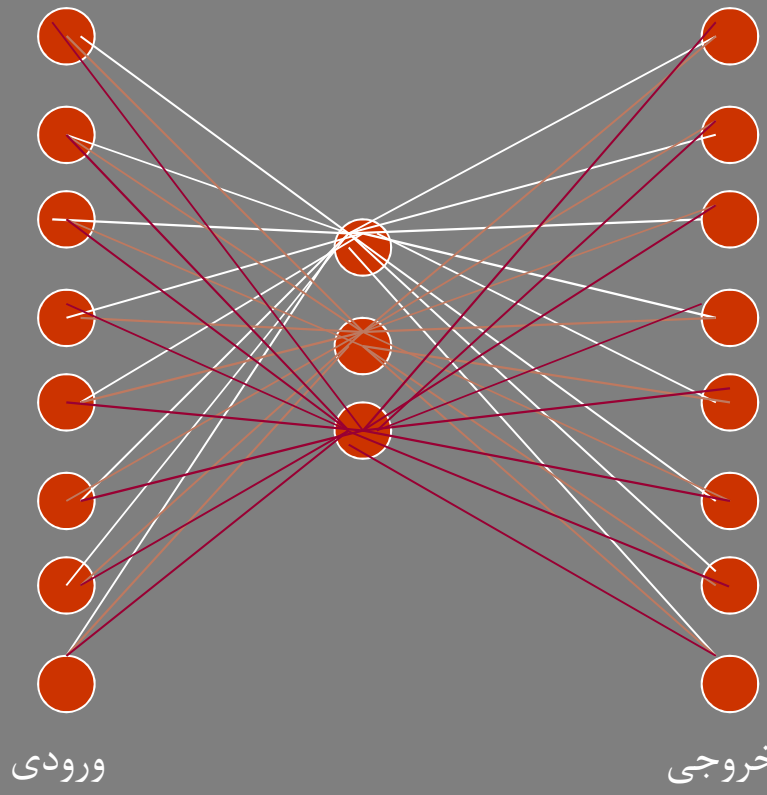
قدرت نمایش توابع

- گرچه قدرت نمایش توابع توسط یک شبکه feedforward بسته به عمق و گستردگی شبکه دارد، با این وجود موارد زیر را میتوان به صورت قوانین کلی بیان نمود:
- توابع بولی :هر تابع بولی را میتوان توسط یک شبکه دو لایه پیاده سازی نمود.
- توابع پیوسته :هر تابع پیوسته محدود را میتوان توسط یک شبکه دو لایه تقریب زد .تئوری مربوطه در مورد شبکه هائی که از تابع سیگموئید در لایه پنهان و لایه خطی در شبکه خروجی استفاده میکنند صادق است.
- توابع دلخواه :هر تابع دلخواه را میتوان با یک شبکه سه لایه تا حد قابل قبولی تفریب زد.

با این وجود باید درنظر داشت که فضای فرضیه جستجو شده توسط روش gradient deescent ممکن است در برگیرنده تمام مقادیر ممکن وزنها نباشد.

قدرت نمایش لایه پنهان

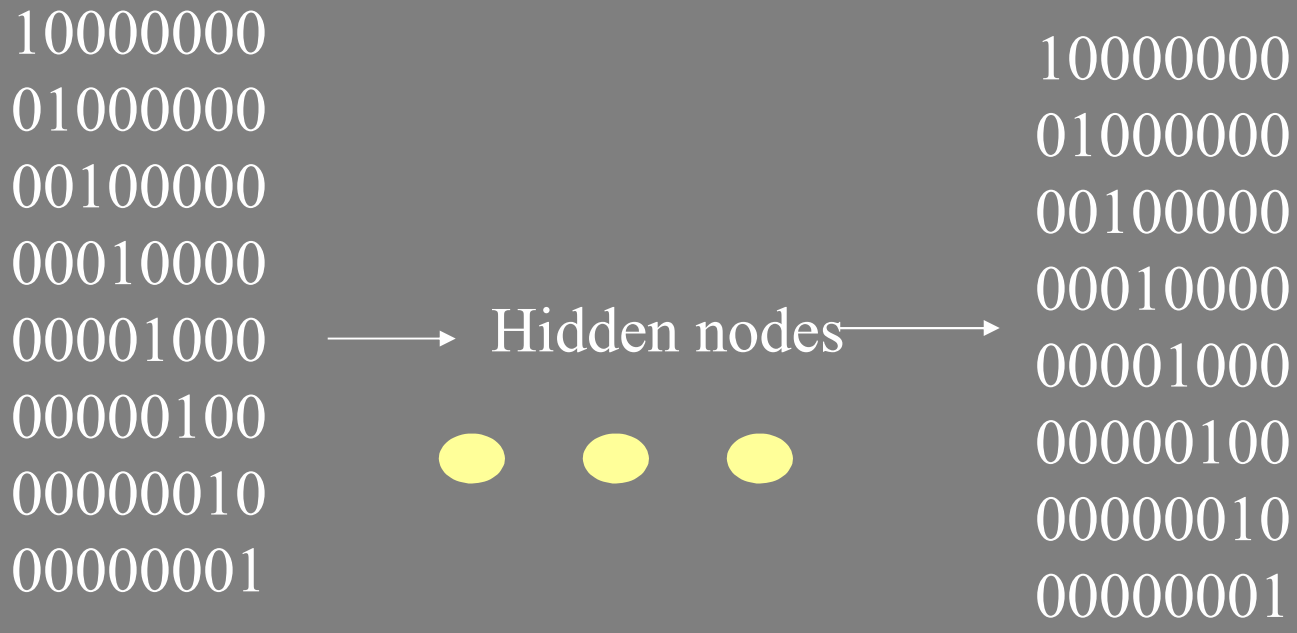
- یکی از خواص BP این است که میتواند در لایه های پنهان شبکه ویژگیهای نا آشکاری از داده ورودی نشان دهد.



برای مثال شبکه8 * 3 * 8 زیر طوری آموزش داده میشود که مقدارهرمثال ورودی را عینا در خروجی بوجو د آورد (تابع f(x)=x را یاد بگیرد). ساختار خاص این شبکه باعث میشود تا واحد های لایه وسط ویژگی های مقادیر ورودی را به نحوی کد بندی کنند که لایه خروجی بتواند از آنان برای نمایش مجدد داده ها استفاده نماید .

قدرت نمایش لایه پنهان

در این آزمایش که به تعداد 5000 بار تکرار شده از 8 داده مختلف به عنوان ورودی استفاده شده و شبکه با استفاده از الگوریتم BP موفق شده تا تابع هدف را بیاموزد.



با مشاهده خروجی واحد های لایه میانی مشخص میشود که بردار حاصل معادل انکدینگ استاندارد داده های ورودی بوده است (000,001,,...,111) .

الگوریتم Levenberg-Marquardt

- بر پایه الگوریتم نیوتن بنا شده است:
  - اگر تابع خطا بصورت زیر بیان شود:

$$V(\vec{w}) = \sum_{h=1}^{N} e^2(\vec{x}_h, \vec{w})$$

  - قانون بروز رسانی نیوتن برای مینیمم کردن تابع خطا بصورت زیر می باشد:

$$\Delta \vec{w} = -\left[\nabla^2 V(\vec{w})\right]^{-1} \nabla V(\vec{w})$$

  - که در آن $\nabla^2 V(\vec{w})$ ماتریس $\nabla V(\vec{w})$ و ماتریس گرادیان می باشد.
  - داریم:

$$\nabla V(\vec{w}) = 2J^T(\vec{w})e(\vec{w})$$

$$\nabla^2 V(\vec{w}) = 2J^T(\vec{w})J(\vec{w}) + 2S(\vec{w})$$

$$J_{i,j}(\vec{w}) = \frac{\partial e(\vec{x}_i, \vec{w})}{\partial w_j}, \quad i = 1,\ldots,N \text{ and } j = 1,\ldots,R$$

$$S(\vec{w}) = \sum_{h=1}^{N} e(\vec{x}_h, \vec{w})\nabla^2 e(\vec{x}_h, \vec{w})$$

Levenberg-Marquardt الگوریتم

- Neglecting the second-order derivatives of the error vector we obtain the Gauss–Newton update, as:

$$\Delta\vec{w} = -\left[J^T(\vec{w})J(\vec{w})\right]^{-1} J^T(\vec{w})e(\vec{w})$$

- The advantage of Gauss–Newton over the standard Newton's method is that it does not require calculation of second-order derivatives.
- The matrix may not be invertible. This is overcome with the Levenberg-Marquardt algorithm

$$\Delta\vec{w} = -\left[J^T(\vec{w})J(\vec{w}) + \mu I\right]^{-1} J^T(\vec{w})e(\vec{w})$$

- Where μ is small the algorithm becomes Gauss-Newton. When μ is large, the algorithm is backpropagation.

حل یک مثال با شبکه عصبی

- یک سیستم جعبه سیاه را با ورودی P و خروجی T درنظر بگیرید:

- P = [0 1 2 3 4 5 6 7 8 9 10];
- T = [0 1 2 3 4 3 2 1 2 3 4];

تابع newff

- Newff(P,T,[S1 S2...S(N-l)],{TF1 TF2...TFNl},
- P: R x Q1 matrix of Q1 sample R-element input vectors
- T: SN x Q2 matrix of Q2 sample SN-element target vectors
- Si: Size of ith layer, for N-1 layers, default = [ ]. Output layer is determined from data
- Tfi: Transfer function of ith layer. (Default = 'tansig' for hidden layers and 'purelin' for output layer.)
- BTF: Backpropagation network training function (default = 'trainlm')
- BLF: Backpropagation weight/bias learning function (default = 'learngdm')
- IPF: Row cell array of input processing functions.
- OPF:Row cell array of output processing functions.
- DDF: Data divison function

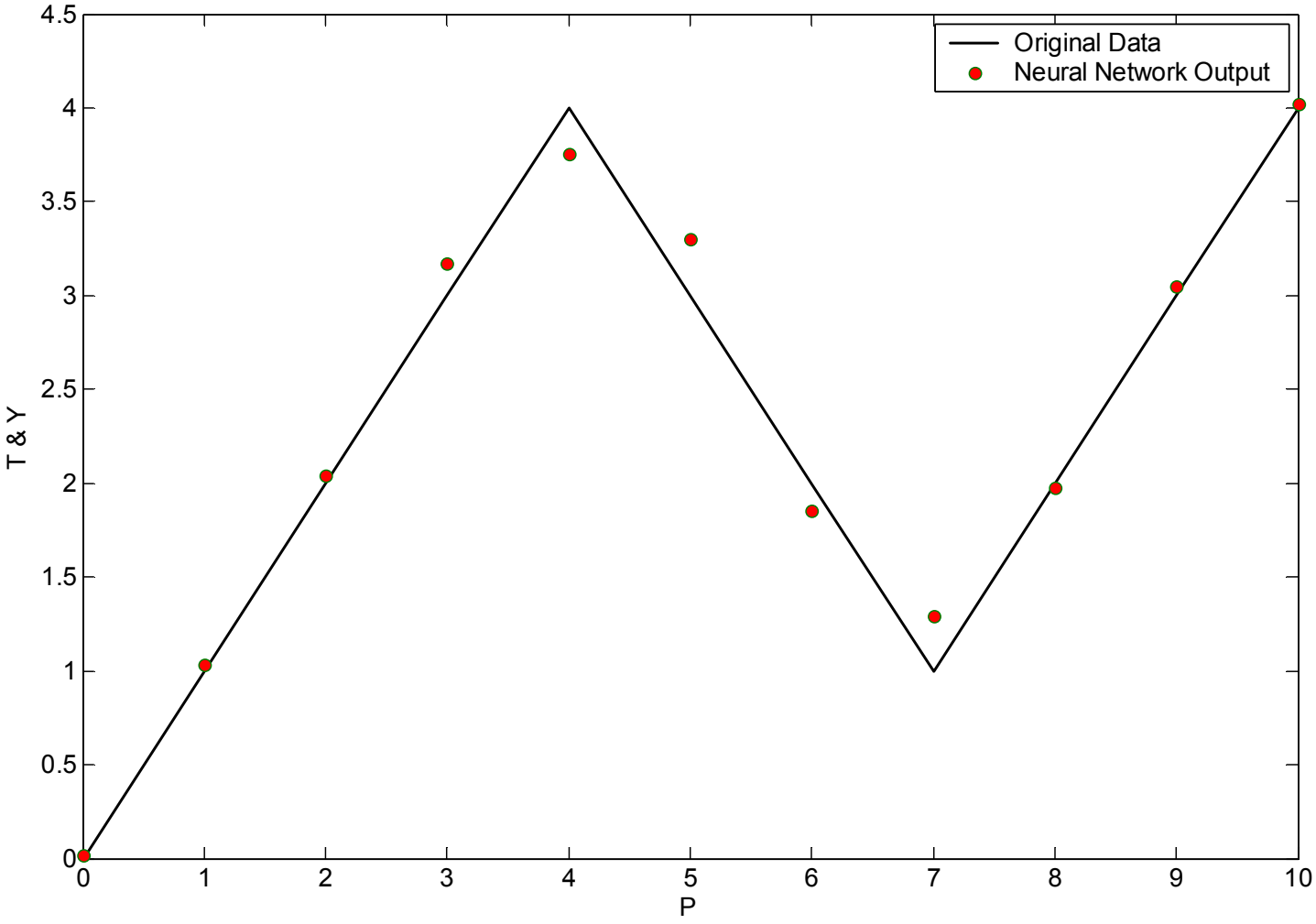
مدلسازی سیستم

- در اینجا یک شبکه دو لایه تغذیه مستقیم برای مدلسازی سیستم درنظر گرفته می‌شود.
- دامنه تغییرات ورودی از ۰ تا ۱۰ است.
- لایه اول با ۵ نرون Tansig و لایه دوم با یک نرون Purelin
- روش آموزش: TRAINLM
- net = newff(P,T,5,{'tansig' 'purelin'},’trainlm’);

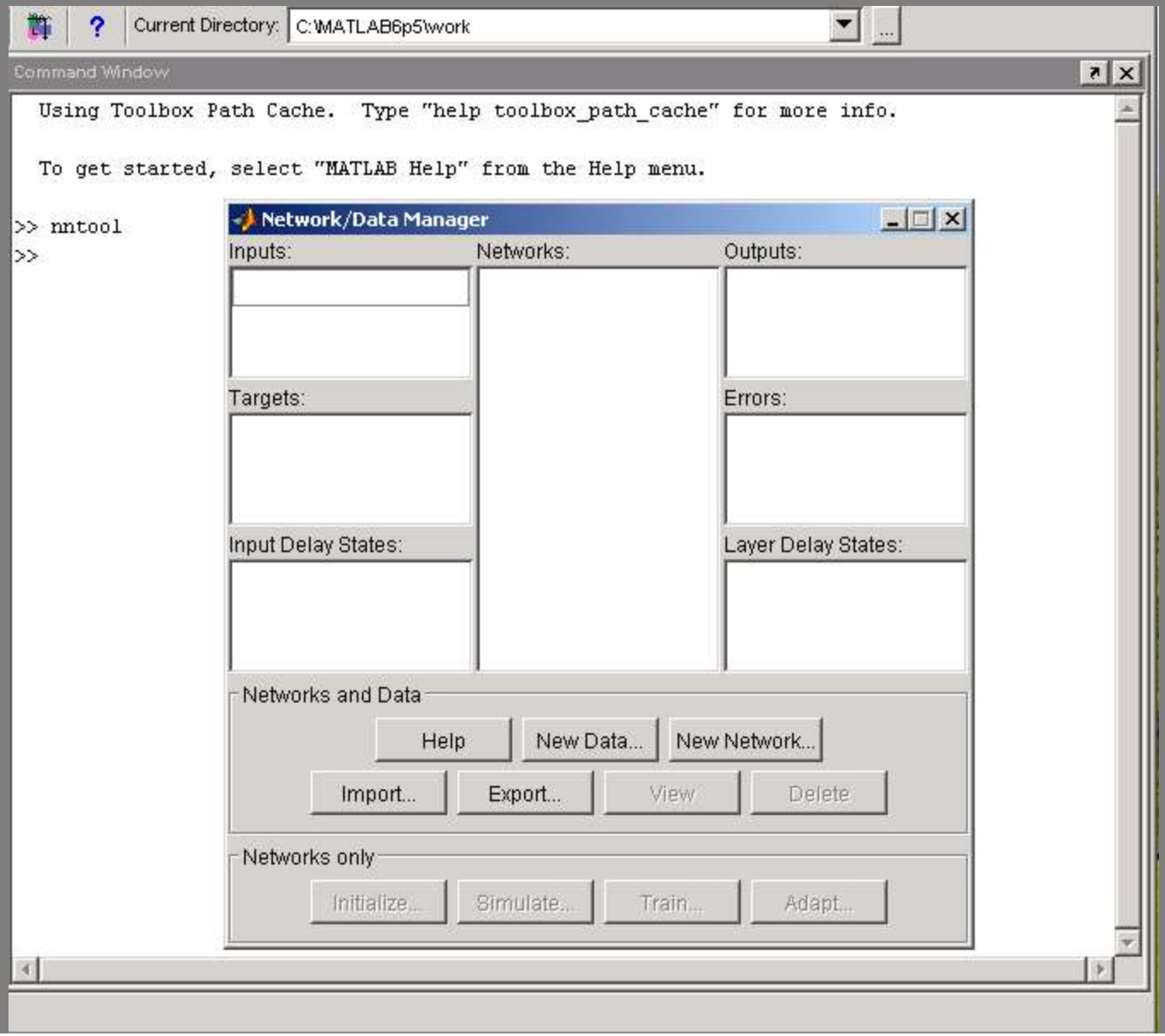
آموزش شبکه

- به شبکه اجازه داده می‌شود که ۵۰ بار از داده‌های ورودی در آموزش استفاده شود

- net.trainParam.epochs = 50;
- net = train(net,P,T);
- Y = sim(net,P);
- plot(P,T,P,Y,'o')



استفاده از رابط گرافیکی کاربر GUI



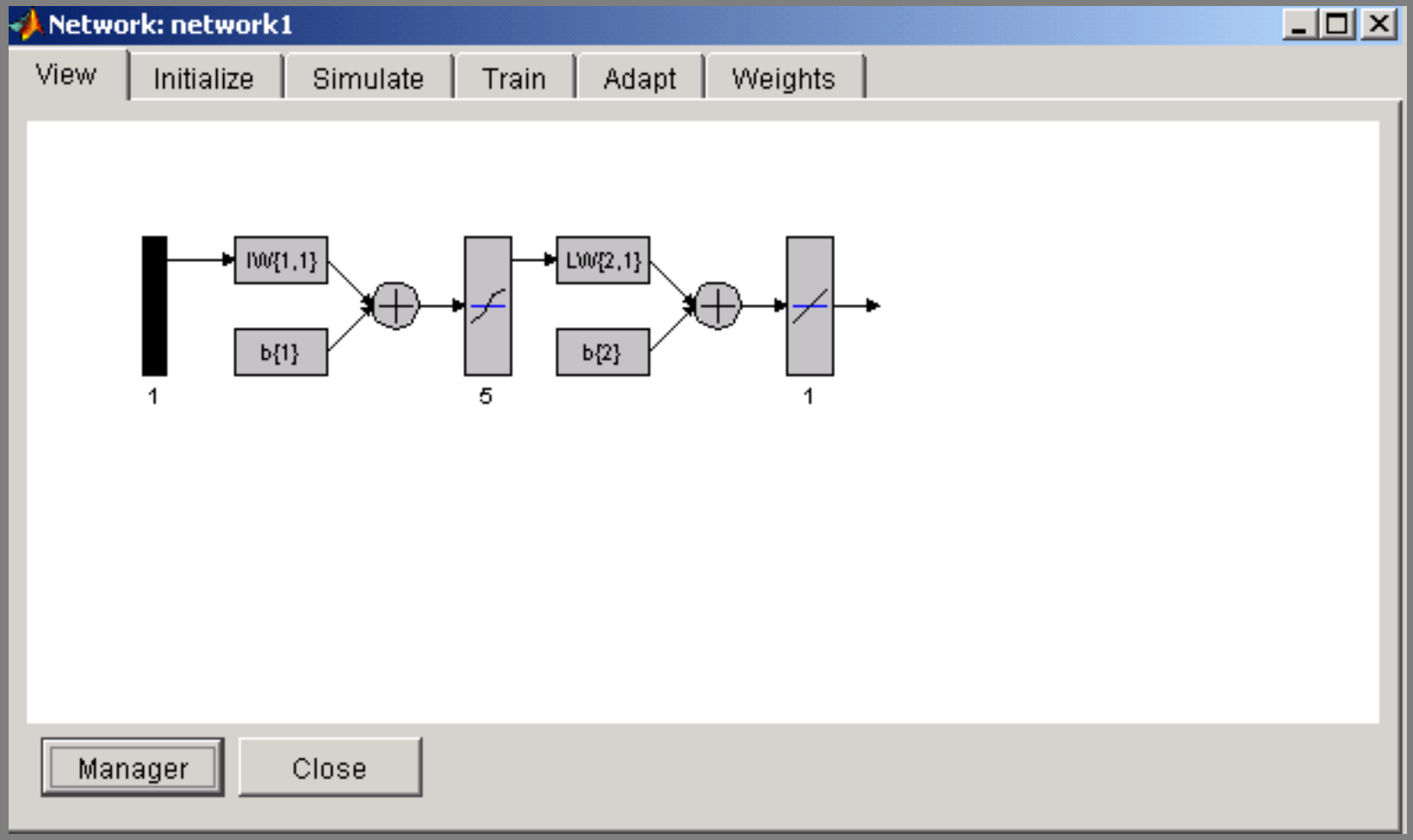
ورود اطلاعات ورودی–خروجی در nntool

- ورود اطلاعات P و T در نرم‌افزار Matlab
- انتخاب گزینه import در nntool
- معرفی P بعنوان ورودی و انتخاب گزینه import
- انتخاب گزینه import در منوی اصلی
- معرفی T بعنوان هدف و انتخاب گزینه import
- مشاهده ورودی و خروجی در Network Manager

ورود اطلاعات شبکه در nntool

- معرفی مدل شبکه با انتخاب new network
- انتخاب نوع شبکه (بعنوان مثال feedforward backpropagation)
- انتخاب مجدد ورودی
- انتخاب روش یادگیری و روش آموزش
- انتخاب تابع بهینه‌سازی
- تعیین تعداد لایه‌ها
- تعیین تعداد نرون‌ها و توابع عملکرد هر لایه
- مشاهده شبکه در Network Manager

شمای شبکه در nntool



آموزش شبکه در nntool

- انتخاب شبکه معرفی‌شده توسط کاربر
- انتخاب گزینه Initialize
- انتخاب گزینه Train
  - مشاهده خروجی شبکه و خطا با انتخاب گزینه Export و انتخاب متغیرهای مربوطه

Y=[-0.0001 1.0006 1.9962 3.0078 3.9751
3.0608 1.9317 1.0304 1.9979 3.0010
4.0002]

وزن‌های شبکه در nntool

- دستیابی به مقادیر وزن‌های لایه‌ها با انتخاب گزینه Weights
- وزن‌های لایه ۱

- [1.4592;
- -2.8307;
- -0.088722;
- -3.657;
- -2.1814]

- وزن‌های لایه ۲

- [2.5401 -1.8955 11.4056 -1.9233 -1.8881]