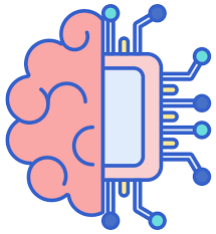
هوش محاسباتی

Computational Intelligence

میلاد سلطانی

دانشگاه آزاد اسلامی



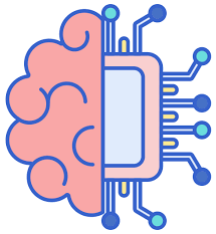
فهرست مطالب فصل دوم

- شبکه عصبی به زبان ساده
- تاریخچه و کاربردهای شبکه عصبی
- تفاوت شبکه عصبی و سایر الگوریتم های یادگیری ماشین
- ساختار و نحوه عملکرد سلول عصبی
- مدل سازی و نحوه عملکرد نورون پایه
- انواع توابع فعال سازی
- شبکه عصبی تک لایه و چند لایه
- پیش خور (Feedforward)
- الگوریتم پس انتشار (Backpropagation)
- الگوریتم گرادیان نزولی (Gradient Descent)

شبکه های عصبی مصنوعی

Artificial Neural Networks

(قسمت اول)



مقدمه‌ای بر شبکه های عصبی

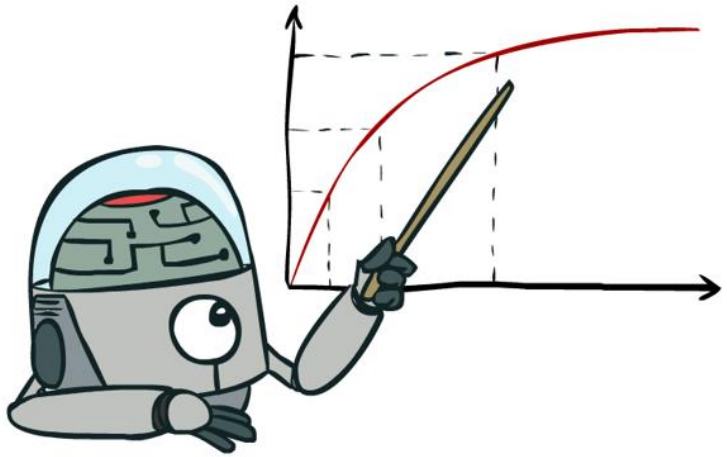
- شبکه‌های عصبی مصنوعی (Artificial Neural Networks) از ساختار و عملکرد مغز انسان الهام گرفته و شامل مجموعه‌ای از واحدهای محاسباتی به نام نورون متصل به هم بوده و به صورت لایه‌ای سازماندهی می‌شوند.
  - در شبکه های مصنوعی تلاش بر این است که با استفاده از عناصر عملکردی سیستم عصبی و مغز، از بخش خاصی از قدرت شناختی انسان (به ویژه قدرت یادگیری) تقلید شود.
- **ایده اصلی**
  - شبیه‌سازی نحوه‌ی یادگیری و پردازش اطلاعات در مغز انسان
  - مغز از میلیاردها سلول عصبی (نورون) تشکیل شده که از طریق سیناپس به یکدیگر متصل هستند.
  - هر نورون با دریافت ورودی، آن را پردازش کرده و خروجی تولید شده را به نورون‌های دیگر منتقل می‌کند.

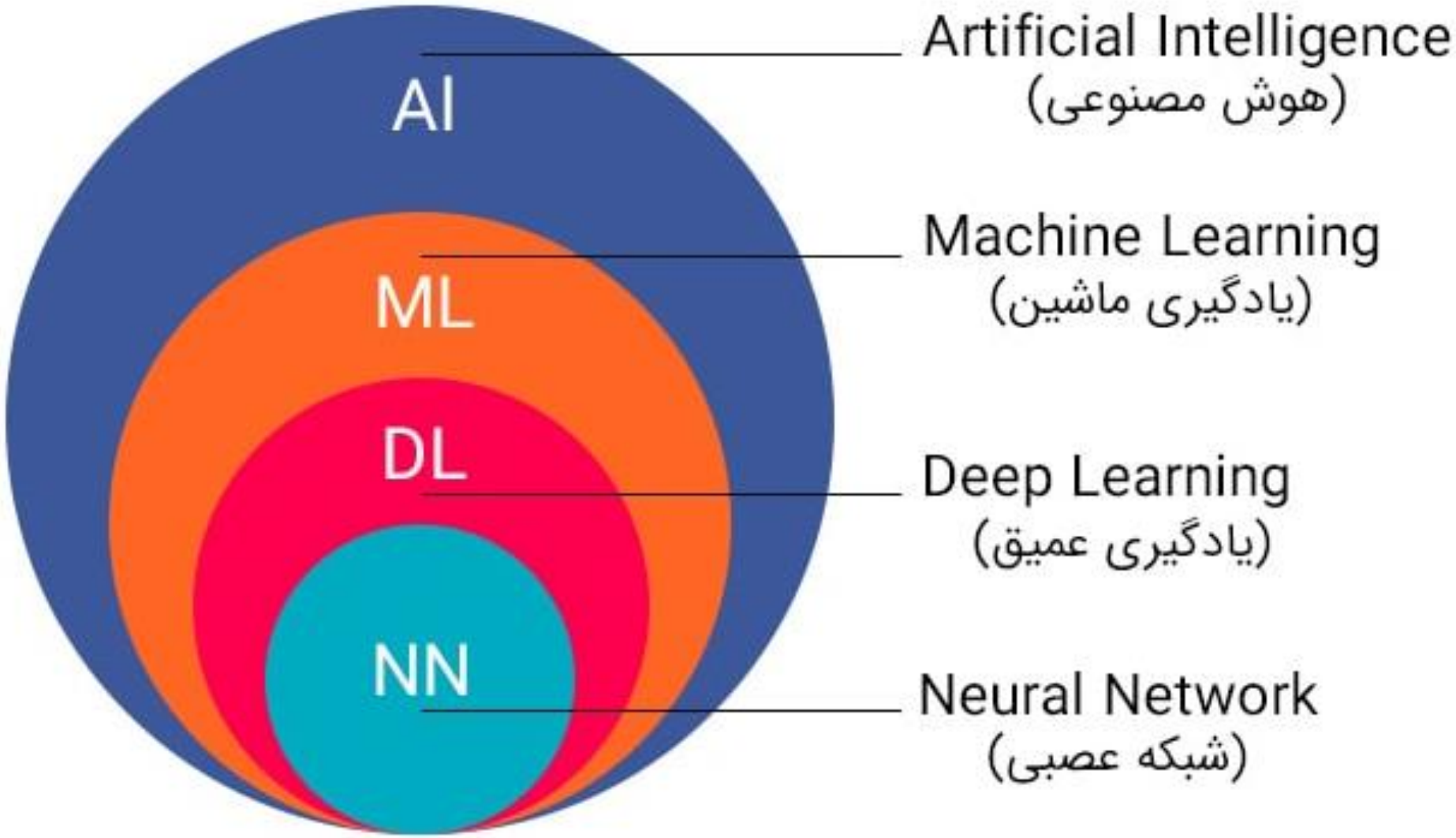
مفهوم شبکه عصبی به زبان ساده

- مفهوم شبکه عصبی مثل این است که بخواهیم به یک کودک یاد بدهیم که چگونه از بین اشکال مختلف، شکل دایره را تشخیص دهد.

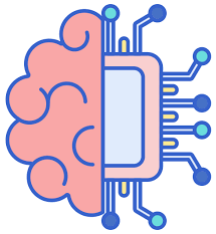
* به او چندین عکس از دایره‌ها در ابعاد و رنگ‌های مختلف نشان می‌دهیم. پس از مدتی یاد می‌گیرد که دایره چیست و می‌تواند از میان همه تصاویری که به او نشان داده می‌شود، دایره‌ها را تشخیص دهد.

* این عملکرد دقیقاً همان کاری‌ست که به کمک شبکه‌های عصبی برای آموزش به یک ماشین انجام می‌دهیم؛ آموزش دادن به ماشین نهایتا باعث ایجاد هوش مصنوعی در آن می‌شود.



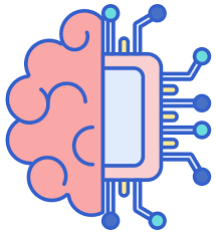


شبکه عصبی و هوش مصنوعی



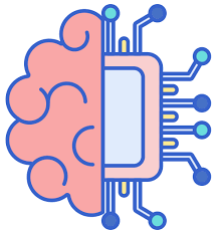
کاربردهای شبکه عصبی

- پردازش تصویر و بینایی کامپیوتر
  - * تشخیص اشیا، چهره‌ها و الگوها
  - * پردازش و تحلیل تصاویر پزشکی
- پردازش زبان طبیعی (NLP)
  - * ترجمه ماشینی و تشخیص گفتار
  - * تولید متون و پاسخ دهی خودکار
  - * تحلیل احساسات
- علوم پزشکی
  - * پیش‌بینی بیماری‌ها
  - * طراحی داروها
  - * تحلیل داده‌های ژنتیکی
- داده کاوی و تحلیل داده
  - * تشخیص الگوها در داده‌های بزرگ
  - * تحلیل بازار و پیش‌بینی قیمت سهام
  - * شناسایی تقلب در تراکنش‌های بانکی
- سیستم‌های توصیه‌گر
  - * پیشنهاد فیلم، موسیقی یا کتاب
  - * پیشنهاد کالا در فروشگاه های آنلاین
- مدل‌سازی و شبیه‌سازی
  - * مدل‌سازی پدیده‌های پیچیده فیزیکی، شیمیایی یا زیستی
  - * شبیه‌سازی فرآیندهای صنعتی و علمی



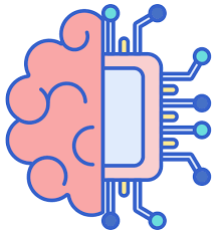
کاربردهای شبکه عصبی

- کنترل و بهینه‌سازی
  - کنترل فرآیندهای صنعتی
  - بهینه‌سازی مصرف انرژی
  - طراحی خودکار سیستم‌های مهندسی
- بازی‌ها و سرگرمی
  - طراحی هوش مصنوعی بازی‌ها
  - تولید محتوای گرافیکی و صوتی
  - شبیه‌سازی شخصیت‌ها
- رباتیک
  - کنترل حرکتی ربات‌ها
  - درک و پاسخ به محیط
  - یادگیری وظایف جدید توسط ربات‌ها
- خودروهای خودران
  - مسیریابی و ناوبری
  - تشخیص موانع و علائم ترافیکی
  - تحلیل داده‌های سنسورها



تاریخچه شبکه‌های عصبی مصنوعی

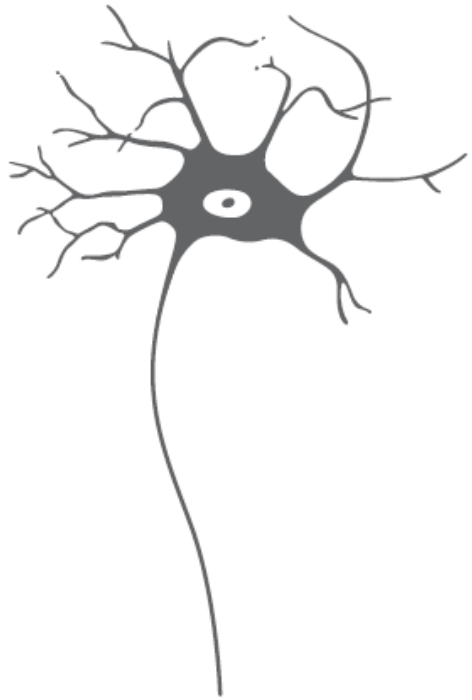
- **آغاز ایده (دهه ۱۹۴۰)**
  - اولین مدل شبکه های عصبی مصنوعی در سال ۱۹۴۳ با تشریح اولین مدل رسمی یک نورون محاسباتی ابتدایی توسط وارن مک کالک (Warren McCulloch) فیزیولوژیست عصبی و والتر پیتز (Walter Pitts) ریاضیدان ارائه شد.
- **ظهور پرسپترون (دهه ۱۹۵۰ و ۱۹۶۰)**
  - فرانک روزِنبلات در سال ۱۹۵۸، اولین مدل شبکه عصبی قابل یادگیری به نام "پرسپترون" را توسعه داد.
  - پرسپترون قادر به یادگیری از داده‌ها بود، اما محدودیت‌هایی داشت، از جمله عدم توانایی در حل مسائل غیرخطی.

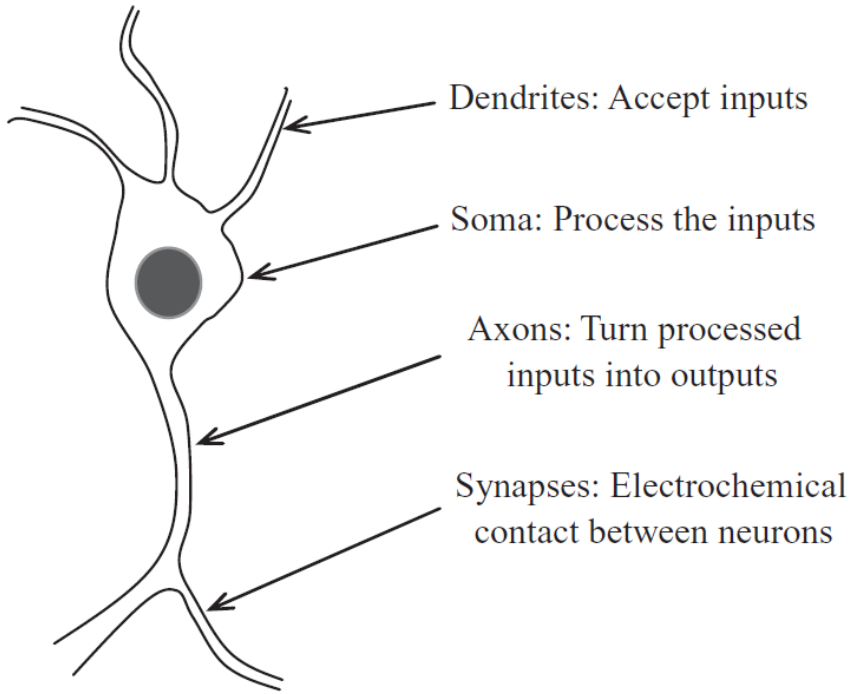


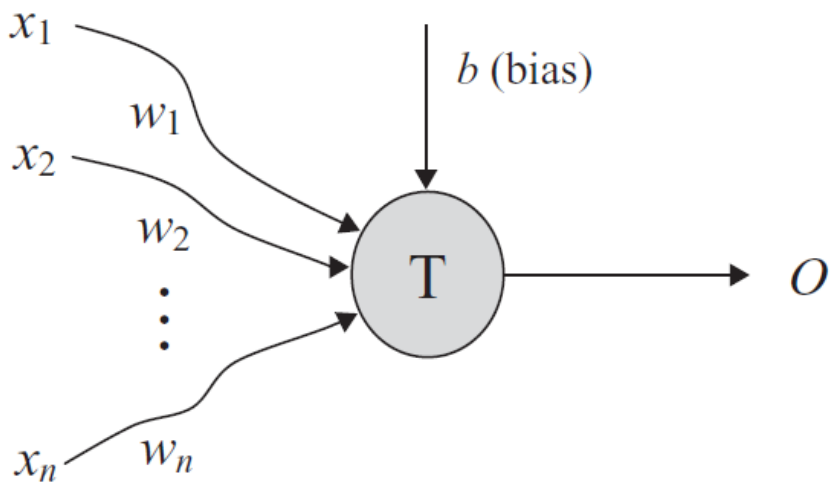
تاریخچه شبکه‌های عصبی مصنوعی

- در سال ۱۹۵۹، برنارد ویدرو (Bernard Widrow) و مارسیان هاف (Marcian Hoff) از دانشگاه استنفورد مدل هایی را توسعه دادند که اولین شبکه‌های عصبی به کار گرفته شده در مسائل واقعی بودند.
  - ADALINE (Adaptive Linear Elements)
  - MADALINE (Multiple ADALINE)

- این مدل ها، قانون یادگیری ویدرو-هاف و اولین شبکه عصبی را تشکیل دادند که برای یک مسأله دنیای واقعی اعمال شد.

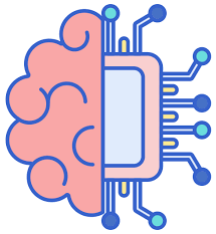






$$O = \begin{cases} 1 & \text{if } \sum_{i=1}^{n} w_i x_i - b \geq T \\ 0 & \text{if } \sum_{i=1}^{n} w_i x_i - b < T \end{cases}$$

where $i = 1, 2, \ldots, n$ and b is called bias.



تاریخچه شبکه‌های عصبی مصنوعی

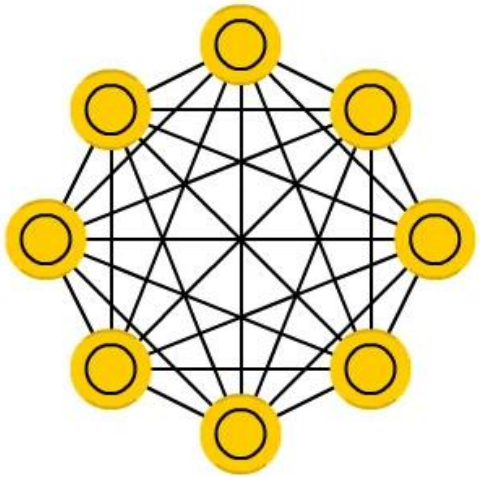
- **دوران رکود (دهه ۱۹۷۰)**
  - **انتقاد ماروین مینسکی و سیمور پاپرت:** در کتابی به نام Perceptrons، محدودیت‌های پرسپترون تک لایه‌ای را برجسته کردند. این انتقادها باعث کاهش علاقه به شبکه‌های عصبی شد.
  - عدم وجود سخت‌افزار و الگوریتم‌های مناسب نیز یکی از دلایل رکود بود.

- **احیا و توسعه (دهه ۱۹۸۰)**
  - **ابداع الگوریتم پس‌انتشار خطا (Backpropagation):** در دهه ۱۹۸۰، روش پس‌انتشار خطا برای آموزش شبکه‌های عصبی چندلایه ارائه شد. این الگوریتم توانست مسائل غیرخطی را حل کند و باعث احیای شبکه‌های عصبی شد.
  - پژوهشگران مانند دیوید رومِل‌هارت و جِفری هینتون نقش کلیدی در توسعه این روش داشتند.

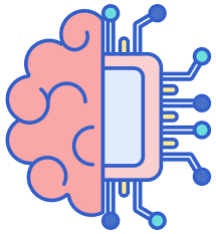
تاریخچه شبکه‌های عصبی مصنوعی

- **احیا و توسعه (دهه ۱۹۸۰)**

Hopfield Network (HN)



* شبکه‌های خود سازمان‌ده (Self-Organizing Maps - SOM)
  - توسط تئوو کوهونن (Teuvo Kohonen) در دهه ۱۹۸۰ توسعه یافت.
  - این شبکه‌ها برای کاهش ابعاد داده و خوشه‌بندی (Clustering) استفاده می‌شوند.
* شبکه‌های بولتزمن (Boltzmann Machines)
  - توسط جفری هینتون معرفی شدند.
  - این مدل‌ها برای یادگیری احتمالاتی و مسائل بهینه‌سازی استفاده می‌شوند.
* شبکه‌های بازگشتی (Recurrent Neural Networks - RNN)
  - جان هاپفیلد (John Hopfield) مدل‌های بازگشتی را توسعه داد که امکان مدل‌سازی سری‌های زمانی و حافظه کوتاه‌مدت را فراهم کردند.

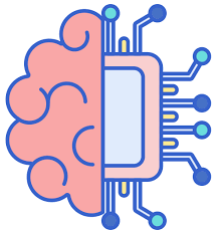


تاریخچه شبکه‌های عصبی مصنوعی

توسعه الگوریتم‌ها و مدل‌های پیشرفته (دهه ۱۹۹۰)

* شبکه‌های عصبی پیچشی (Convolutional Neural Networks - CNNs)
  - شبکه‌های عصبی پیچشی که در دهه ۱۹۸۰ معرفی شدند، در این دهه توسط یان لیکان (Yann LeCun) و تیم او توسعه یافتند.
  - CNNها به دلیل توانایی بالا در پردازش تصویر، به یکی از پُر کاربردترین مدل‌های شبکه‌های عصبی تبدیل شدند.

* شبکه‌های بازگشتی (Recurrent Neural Networks - RNNs)
  - RNNها در دهه ۱۹۹۰ برای پردازش داده‌های سری زمانی و زبان‌های طبیعی پیشرفت‌های چشمگیری داشتند.
  - الگوریتم‌های پیشرفته مثل Long Short-Term Memory (LSTM) توسط سِپ هوچریتر (Sepp Hochreiter) و یورگن اشمید هوبر (Jürgen Schmidhuber) در سال ۱۹۹۷ معرفی شدند.

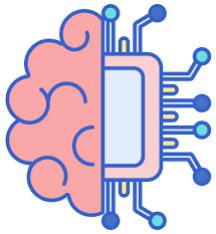


تاریخچه شبکه‌های عصبی مصنوعی

- **رشد چشمگیر (دهه ۲۰۰۰)**

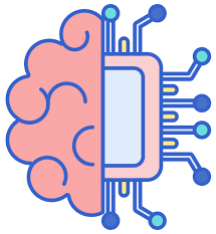
  * **ظهور یادگیری عمیق(Deep Learning)**
    - با پیشرفت سخت‌افزار (بویژه پردازنده‌های گرافیکی) و دسترسی به داده‌های بزرگ، شبکه‌های عصبی عمیق با لایه‌های متعدد مورد استفاده قرار گرفتند.
    - معماری هایی مانند شبکه‌های پیچشی (CNN) برای پردازش تصویر و شبکه‌های بازگشتی (RNN) برای پردازش سری‌های زمانی و زبان توسعه یافتند.

  * **پیشرفت سخت‌افزاری**
    - استفاده از پردازنده‌های گرافیکی (GPUs) به دلیل توانایی پردازش موازی بالا، سرعت آموزش شبکه‌های عصبی را به طرز چشمگیری افزایش دادند.



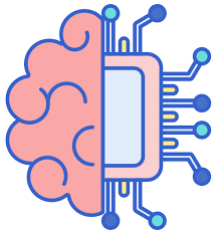
تاریخچه شبکه‌های عصبی مصنوعی

- **کاربردهای مدرن (دهه ۲۰۱۰)**
  - **گسترش یادگیری عمیق (Deep Learning)**
    - ImageNet در سال ۲۰۱۲ با استفاده از CNNها، دقت تشخیص تصویر را به‌طور قابل توجهی افزایش داد.
    - AlphaGo محصول شرکت (DeepMind) با شکست لی سدول، قهرمان جهانی بازی Go، قدرت یادگیری عمیق و یادگیری تقویتی (Reinforcement Learning) را نشان داد.
  - **پیشرفت در مدل‌های شبکه‌های عصبی**
    - استفاده از Transformerها و مدل‌هایی مانند GPT و BERT برای پردازش زبان طبیعی و کاربردهای پیشرفته‌تر
  - **توسعه ابزارهای متن باز و چارچوب‌ها**
    - TensorFlow (2015)
    - PyTorch (2016)
    - Keras



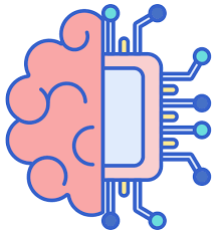
تاریخچه شبکه‌های عصبی مصنوعی

- **دهه فعلی کاربردهای فوق چشمگیر (از دهه ۲۰۲۰ به بعد)**
  - ظهور مدل‌های بزرگ زبان (Large Language Models - LLMs)
    - معماری هایی مانند Transformer و مدل‌هایی مانند GPT و BERT برای پردازش زبان طبیعی و کاربردهای پیشرفته‌تری به کار گرفته شدند.
  - ترکیب شبکه‌های عصبی با روش‌های جدید
    - معماری‌های چند وجهی (Multimodal Models) مانند CLIP (OpenAI) - DALL-E (OpenAI) – Stable Diffusion - MidJourney
  - پیشرفت در توان محاسباتی
    - تراشه‌های تخصصی
    - رشد محاسبات ابری: دسترسی به منابع محاسباتی عظیم توسط AWS، Google Cloud و Microsoft Azure
  - یادگیری انتقالی (Transfer Learning)



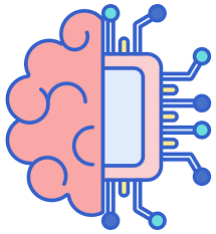
تفاوت شبکه‌های عصبی با سایر روش‌های یادگیری ماشین

- **ساختار و الهام‌گیری زیستی**
  - **شبکه‌های عصبی مصنوعی**
    - ساختار شبکه‌های عصبی از مغز انسان الهام گرفته شده است.
  - **سایر روش‌های یادگیری ماشین:**
    - بیشتر آن‌ها بر اساس مفاهیم آماری یا الگوریتم‌های خاص طراحی شده و معمولاً ساختاری ساده‌تر دارند.
- **قابلیت یادگیری ویژگی‌ها**
  - **شبکه‌های عصبی:**
    - شبکه‌های عصبی توانایی یادگیری خودکار ویژگی‌ها (Feature Learning) از داده‌های خام را دارند.
  - **سایر روش‌ها:**
    - معمولاً نیاز به پیش‌پردازش و استخراج ویژگی‌های دستی دارند.



تفاوت شبکه‌های عصبی با سایر روش‌های یادگیری ماشین

- **عملکرد در مسائل پیچیده و داده‌های بزرگ**
  - **شبکه‌های عصبی:**
    - در حل مسائل پیچیده با داده‌های بزرگ و چندبُعدی (مانند تصاویر، ویدئوها، و متن) بسیار قدرتمند هستند.
  - **سایر روش‌ها:**
    - در مسائل ساده یا داده‌های کم، اغلب کارایی بهتری دارند و سریع‌تر آموزش می‌بینند.
- **انعطاف‌پذیری**
  - **شبکه‌های عصبی:**
    - قابل استفاده در دامنه‌های مختلف از تشخیص تصویر و صدا گرفته تا پردازش زبان طبیعی و بازی‌ها.
  - **سایر روش‌ها:**
    - انعطاف‌پذیری کمتری دارند و معمولاً برای مسائل خاص بهینه‌سازی می‌شوند.

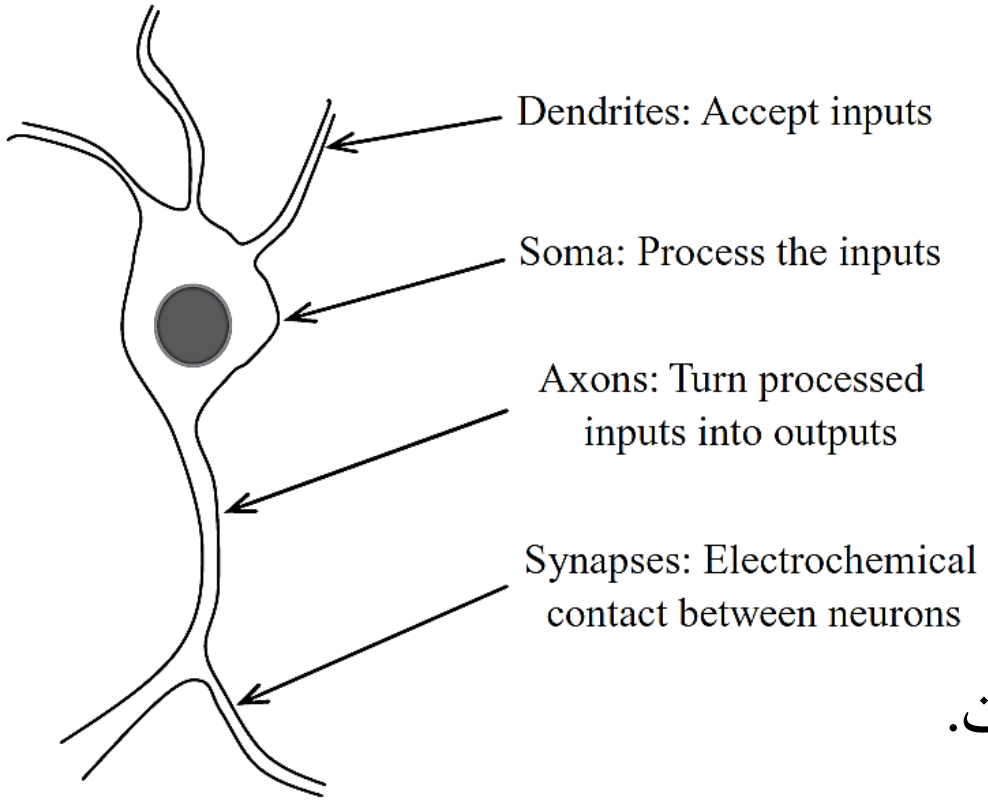


تفاوت شبکه‌های عصبی با سایر روش‌های یادگیری ماشین

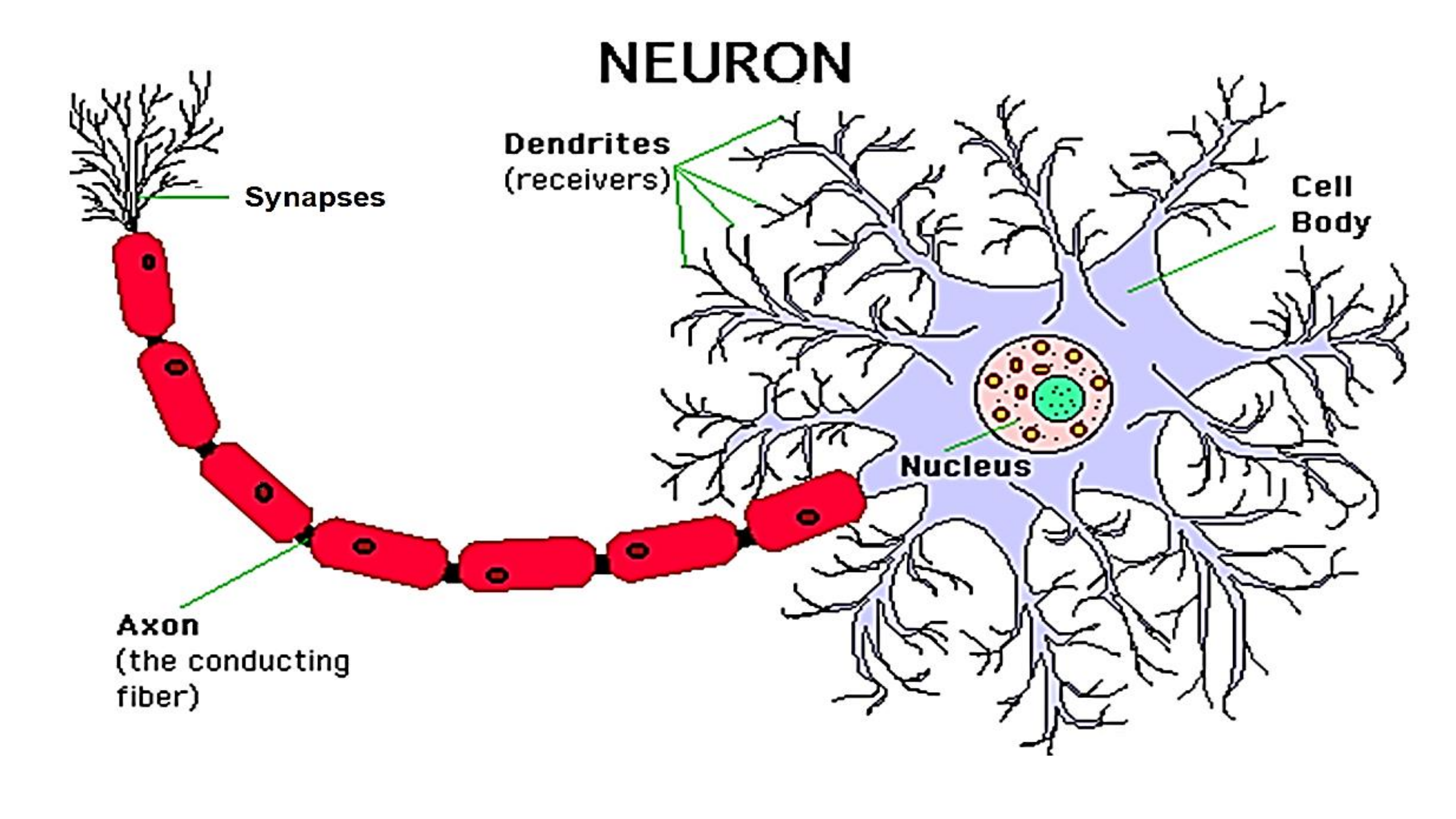
- **تفسیرپذیری**
  - **شبکه‌های عصبی:**
    - تفسیر فرایند یادگیری آن‌ها دشوار است، چرا که شامل هزاران یا میلیون‌ها پارامتر می‌شوند.
  - **سایر روش‌ها:**
    - روش‌هایی مانند درخت تصمیم یا رگرسیون خطی تفسیرپذیرتر بوده و فهم نتایج و روابط میان ویژگی‌ها در آن‌ها ساده‌تر است.
- **کاربردها**
  - **شبکه‌های عصبی:**
    - مناسب برای مسائل پیچیده که نیاز به درک عمیق از داده‌ها دارند، مانند ترجمه ماشینی
  - **سایر روش‌ها:**
    - بیشتر در مسائل ساده‌تر یا جایی که داده‌ها محدود هستند، استفاده می‌شوند، مانند خوشه‌بندی اولیه

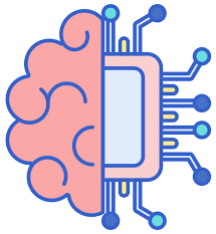
ساختار و ویژگی های سلول عصبی

- سیستم عصبی انسان از سلول هایی به نام نورون (Neuron) ساخته شده و اجزای آن:
  - سوما (Cell body(Soma
  - دندریت (Cell input(Dendrite
  - آکسون (Cell output(Axon
  - سیناپس (Neuron contact (Synapses



- مغز انسان بصورت موازی پردازش می کند.
  - تعداد $۱۰^{۱۱}$ نورون در مغز وجود دارد
  - هر نورون به $۱۰^{۴}$ نورون دیگر متصل است
- سرعت پاسخ نورون ها از مدارهای الکتریکی کندتر است.
  - نورون در $۱۰^{-۳}$ ثانیه و مدار الکتریکی در $۱۰^{-۹}$ ثانیه پاسخ می دهد


NEURON
Dendrites
(receivers)
Synapses
Cell
Body
Nucleus
Axon
(the conducting
fiber)

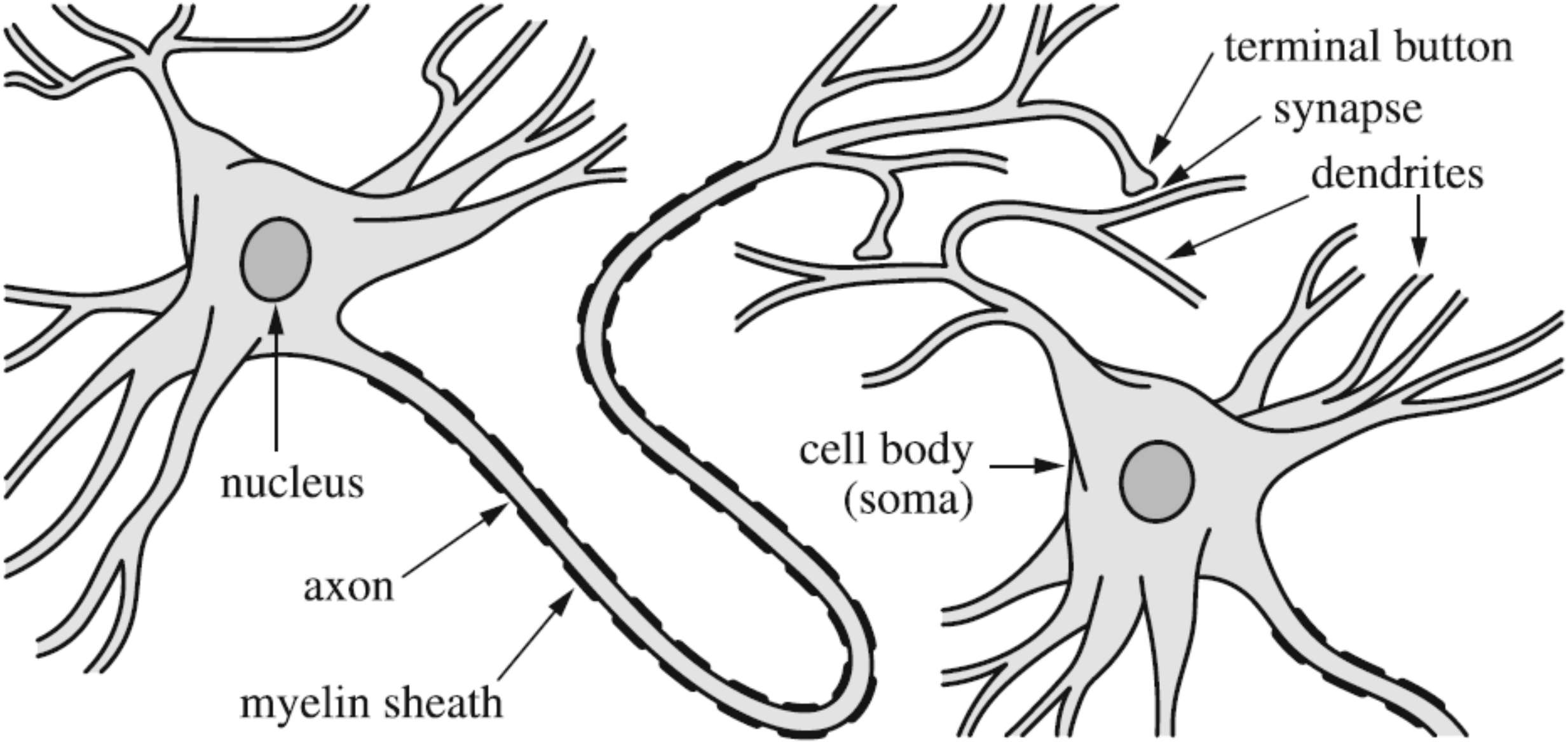


عملکرد سلول عصبی

- آکسون از یک نورون به دندریت های نورون های دیگر در نقطه سیناپس منتهی می شود.

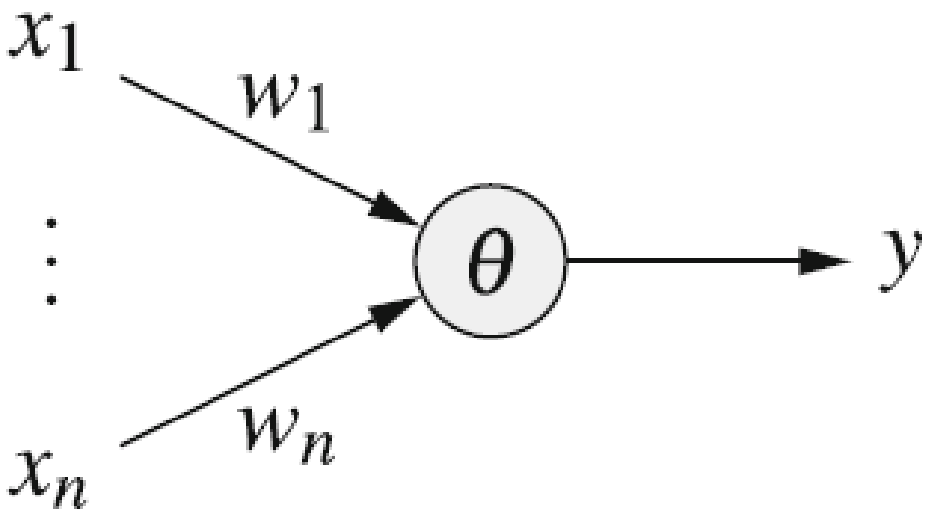
- انتقال دهنده عصبی نوروترنسمیتر (Neurotransmitter) از آکسون آزاد و روی غشای دندریت دریافت کننده سبب تغییر پتانسیل الکتریکی می شود.
  - * سیناپس کاهنده اختلاف پتانسیل، تحریک کننده (Excitatory) است.
  - * سیناپس افزاینده اختلاف پتانسیل، بازدارنده (Inhibitory) است.

- اگر یک نورون به اندازه کافی ورودی محرّک دریافت کند و مقدار ورودی به یک آستانه مشخص برسد، نورون فعال شده و به نورون های دیگر سیگنال (تکانه عصبی) ارسال می کند.
  - * تعداد تکانه های عصبی ارسالی یک نورون در هر ثانیه **نرخ شلیک (Firing Rate)** نامیده می شود.


terminal button
synapse
dendrites
cell body
(soma)
nucleus
axon
myelin sheath

مدل نورون پایه

- در نورون بیولوژیکی اگر مجموع ورودی ها نسبت به آستانه (Threshold):
  - * بیشتر باشد، نورون فعال است (یک)
  - * کمتر باشد، نورون غیرفعال است (صفر)



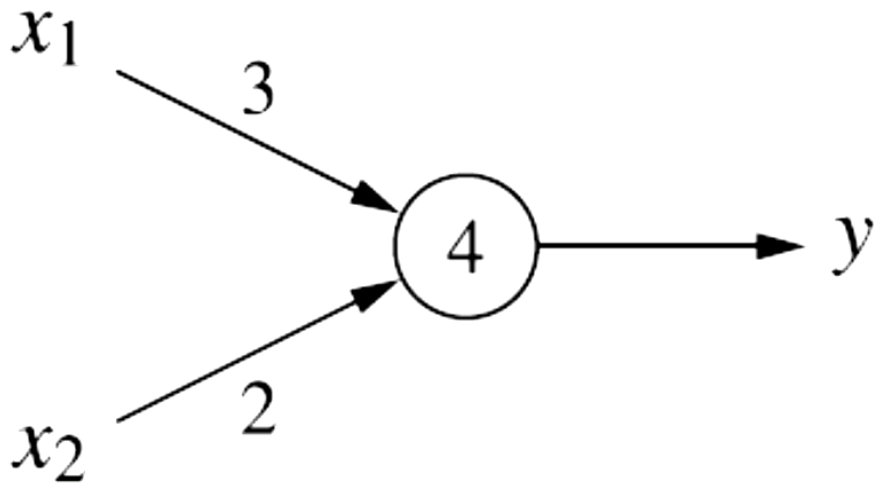
- مقادیر x_i ورودی ها
- مقادیر w_i وزن ها
- مقدار y خروجی
- مقدار θ آستانه

$$y = \begin{cases} 1 & \text{if} \quad \sum_{i=1}^{n} w_i x_i \geq \theta, \\ 0 & \text{otherwise.} \end{cases}$$

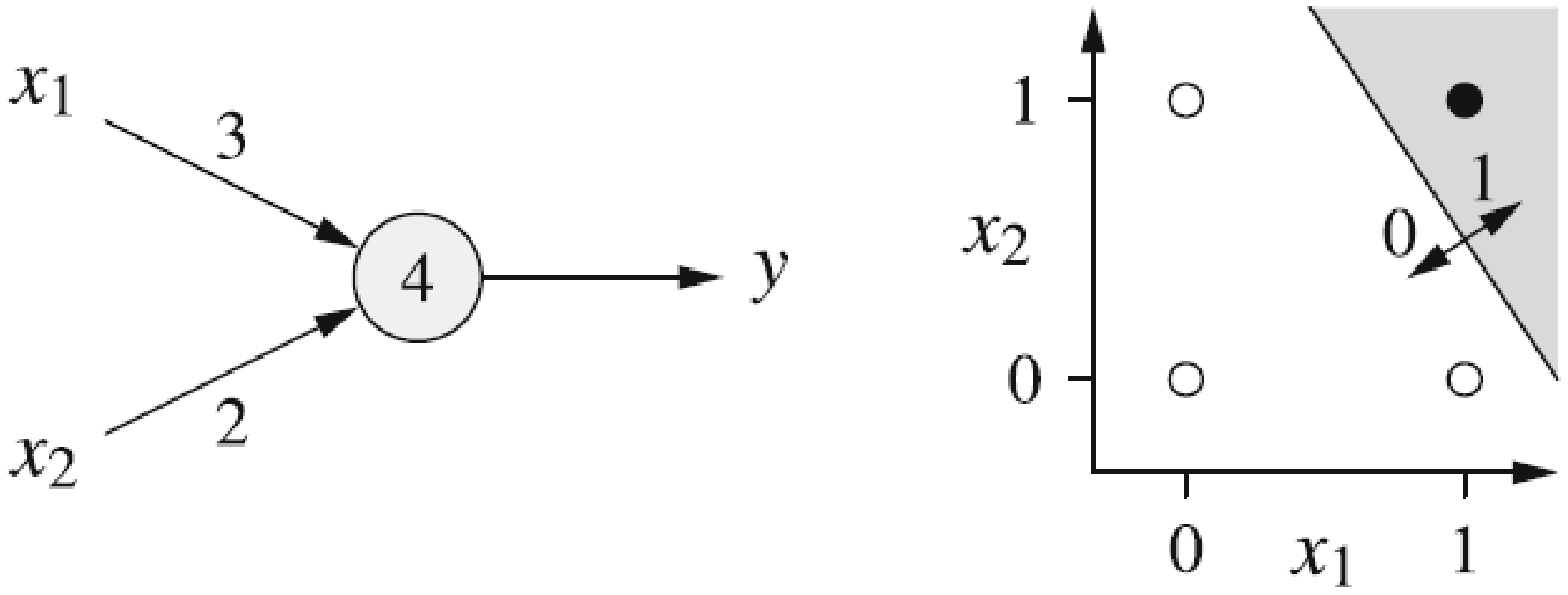
مثال نورون پایه برای ترکیب عطفی

- برای پیاده سازی ترکیب عطفی (AND) یک نورون پایه با دو ورودی x_1 و x_2 با وزن های $w_1=3$ و $w_2=2$ و آستانه $\theta=4$ در نظر می گیریم.

$$x_1 \wedge x_2 \Longrightarrow \sum_{i=1}^{2} w_i x_i \Longrightarrow 3x_1 + 2x_2 \Longrightarrow y = \begin{cases} 1, & 3x_1 + 2x_2 \geq 4 \\ 0, & otherwise \end{cases}$$



x_1	x_2	$3x_1 + 2x_2$	y
0	0	0	0
1	0	3	0
0	1	2	0
1	1	5	1

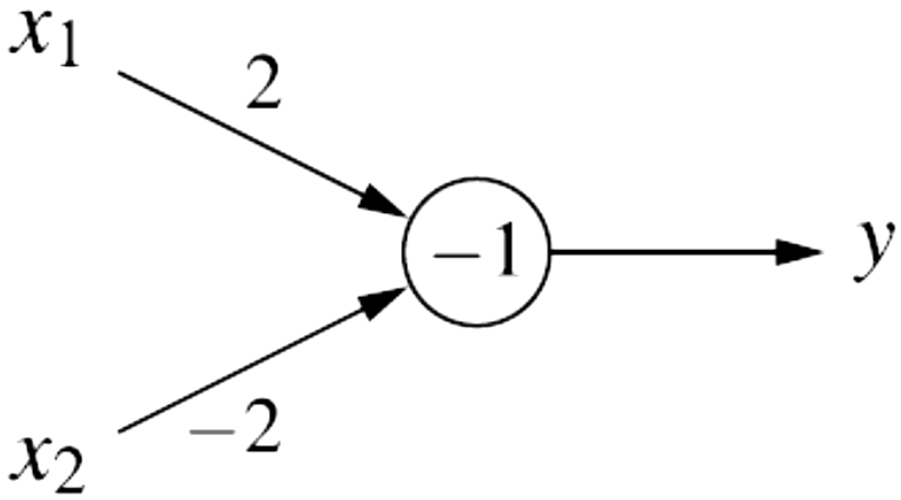


شکل هندسی پیاده سازی ترکیب عطفی (AND) برای $x_1 \Lambda x_2$ که خط درون شکل دارای معادله $3x_1 + 2x_2 = 4$ است و برای بخش خاکستری، معادله برابر است با $3x_1 + 2x_2 \geq 4$

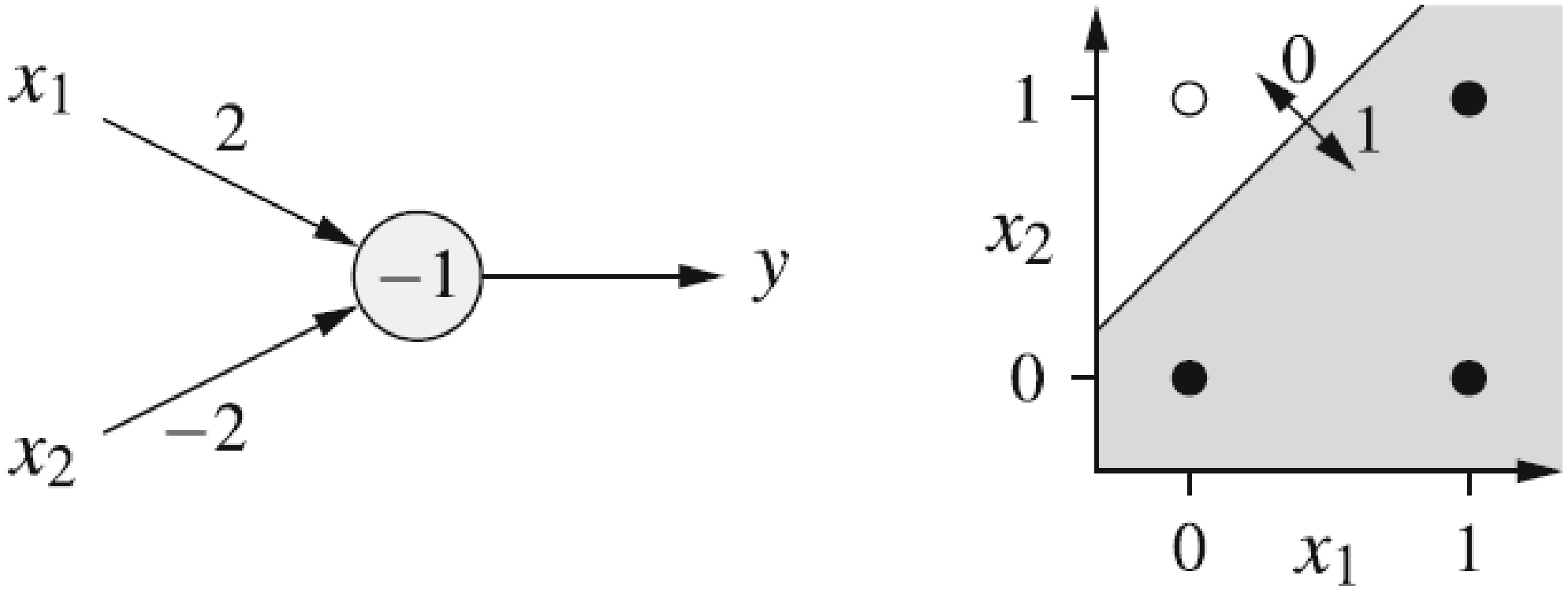
مثال نورون پایه برای ترکیب شرطی

- برای پیاده سازی ترکیب شرطی ($\rightarrow$) یک نورون پایه با دو ورودی x_1 و x_2 دارای وزن های $w_1=2$ و $w_2= -2$ و آستانه $\theta= -1$ در نظر می گیریم.

$$x_1 \longrightarrow x_2 \Longrightarrow \sum_{i=1}^{2} w_i x_i \Longrightarrow 2x_1 - 2x_2 \Longrightarrow y = \begin{cases} 1, & 2x_1 - 2x_2 \geq -1 \\ 0, & otherwise \end{cases}$$



x_1	x_2	$2x_1 - 2x_2$	y
0	0	0	1
1	0	2	1
0	1	-2	0
1	1	0	1



شکل هندسی پیاده سازی ترکیب شرطی $(\rightarrow)$ برای $x_2 \longrightarrow x_1$ که خط درون شکل دارای معادله $2x_1 - 2x_2 = -1$ است و برای بخش خاکستری، معادله برابر است با $2x_1 - 2x_2 \geq -1$

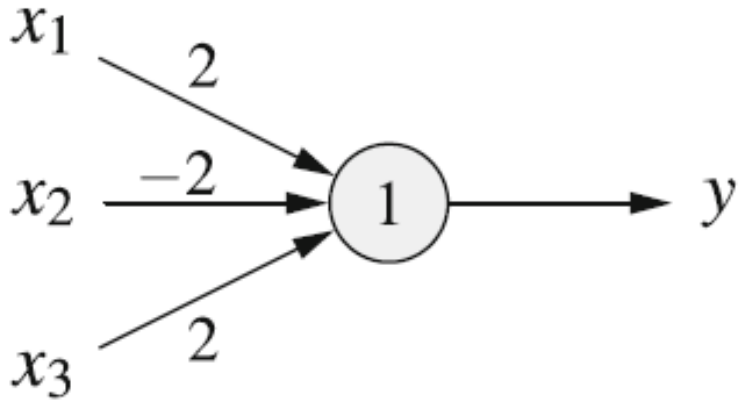
مثال نورون پایه برای یک عبارت پیچیده

- برای پیاده سازی یک عبارت پیچیده، یک نورون پایه با سه ورودی x_1 و x_2 و x_3 با وزن های $w_1=2$ و $w_2= -2$ و $w_3=2$ آستانه $\theta=1$ در نظر می گیریم.

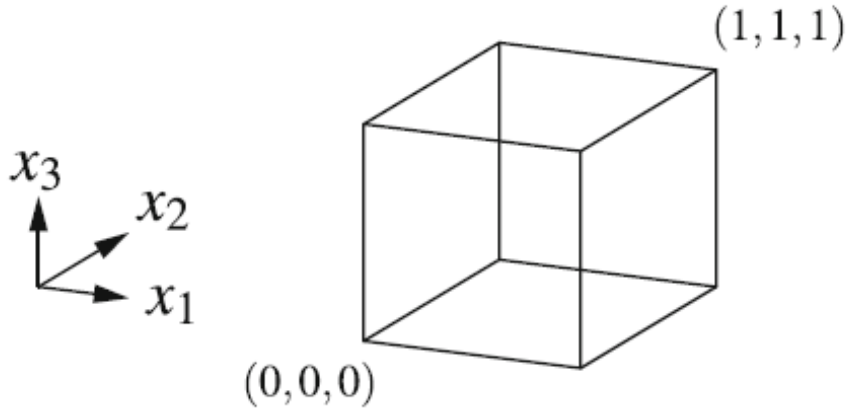
$$(x_1 \wedge \overline{x_2}) \vee (x_1 \wedge x_3) \vee (\overline{x_2} \wedge x_3) \Longrightarrow$$

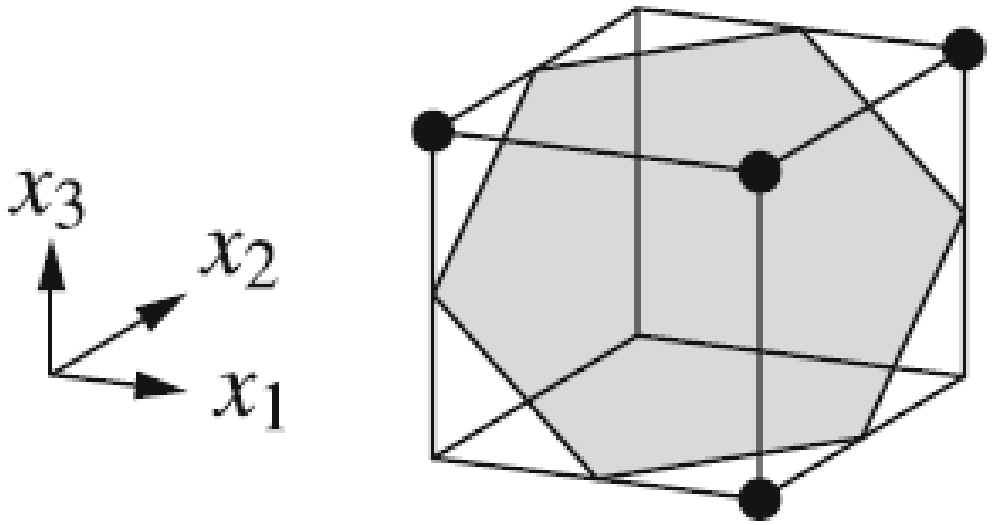
$$\sum_{i=1}^{3} w_i x_i \Longrightarrow 2x_1 - 2x_2 + 2x_3 \Longrightarrow$$

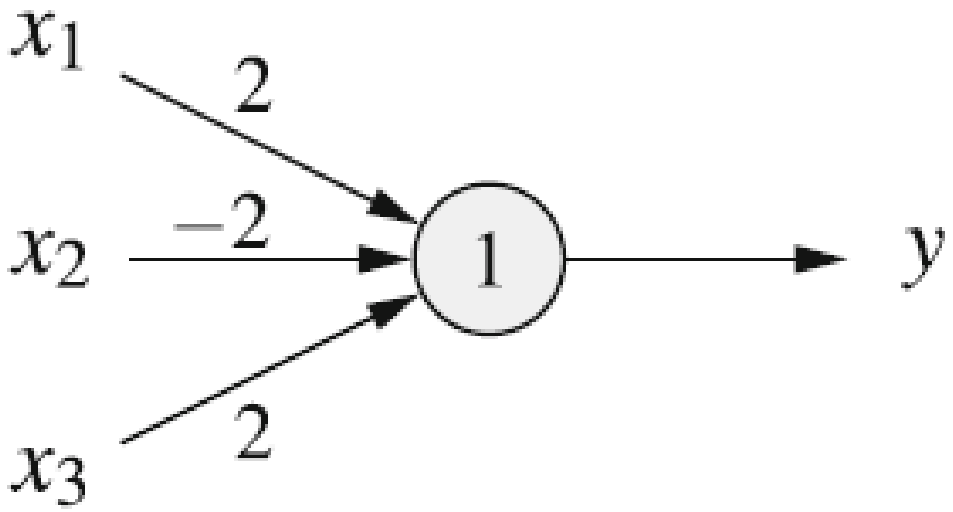
$$y = \begin{cases} 1, & 2x_1 - 2x_2 + 2x_3 \geq 1 \\ 0, & otherwise \end{cases}$$



x_1	x_2	x_3	$\sum_i w_i x_i$	y
0	0	0	0	0
1	0	0	2	1
0	1	0	−2	0
1	1	0	0	0
0	0	1	2	1
1	0	1	4	1
0	1	1	0	0
1	1	1	2	1





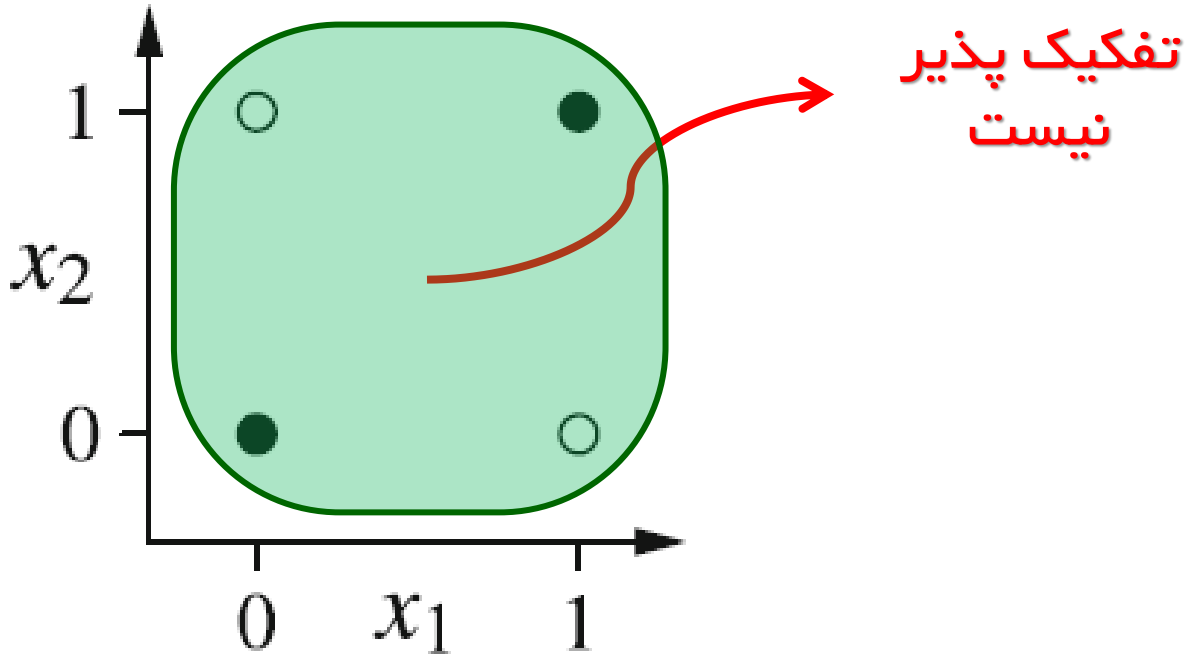


شکل هندسی پیاده سازی برای $(x_1 \Lambda \overline{x_2}) \vee (x_1 \Lambda x_3) \vee (\overline{x_2} \wedge x_3)$ که صفحه درون شکل دارای معادله $2x_1 - 2x_2 + 2x_3 = -1$ است.

محدودیت قدرت بیان و محاسبات نورون پایه

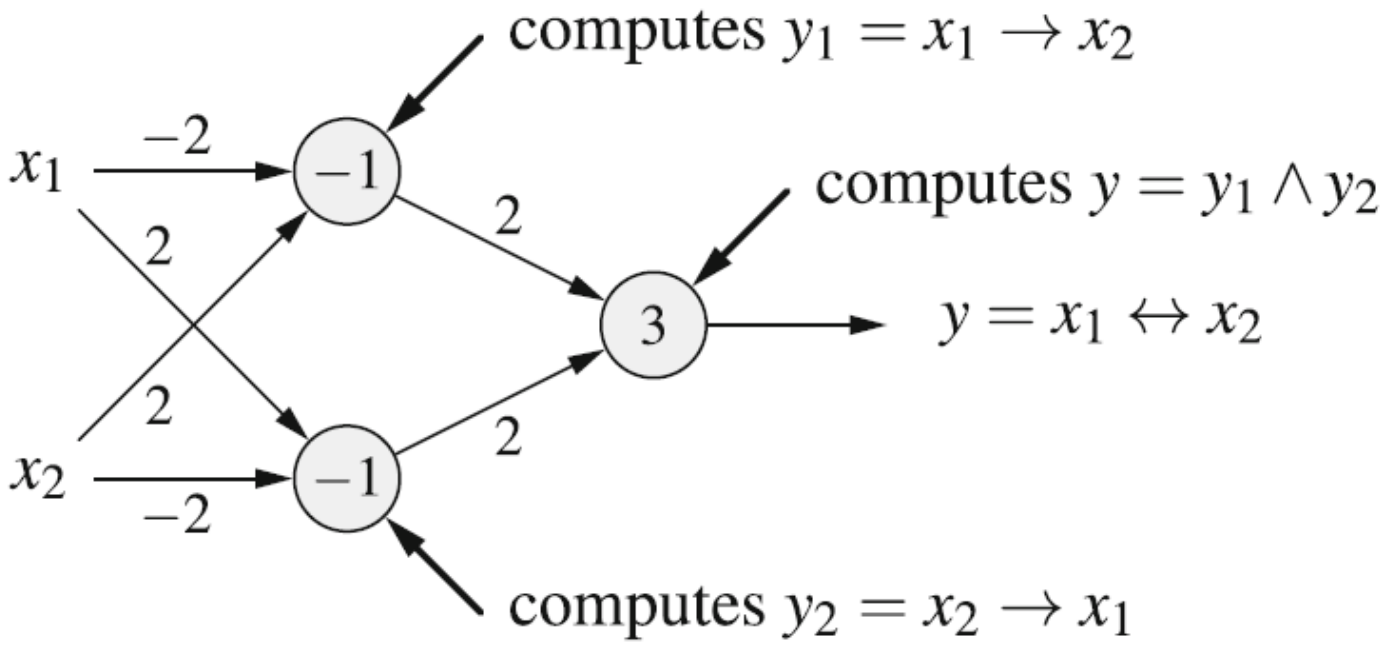
- نورون پایه تنها می تواند توابعی را نمایش دهد که تفکیک پذیر خطی هستند.
- ترکیب دو شرطی (↔) برای یک نورون پایه با دو ورودی x_1 و x_2 را در نظر می گیریم.

x_1	x_2	y
0	0	1
1	0	0
0	1	0
1	1	1



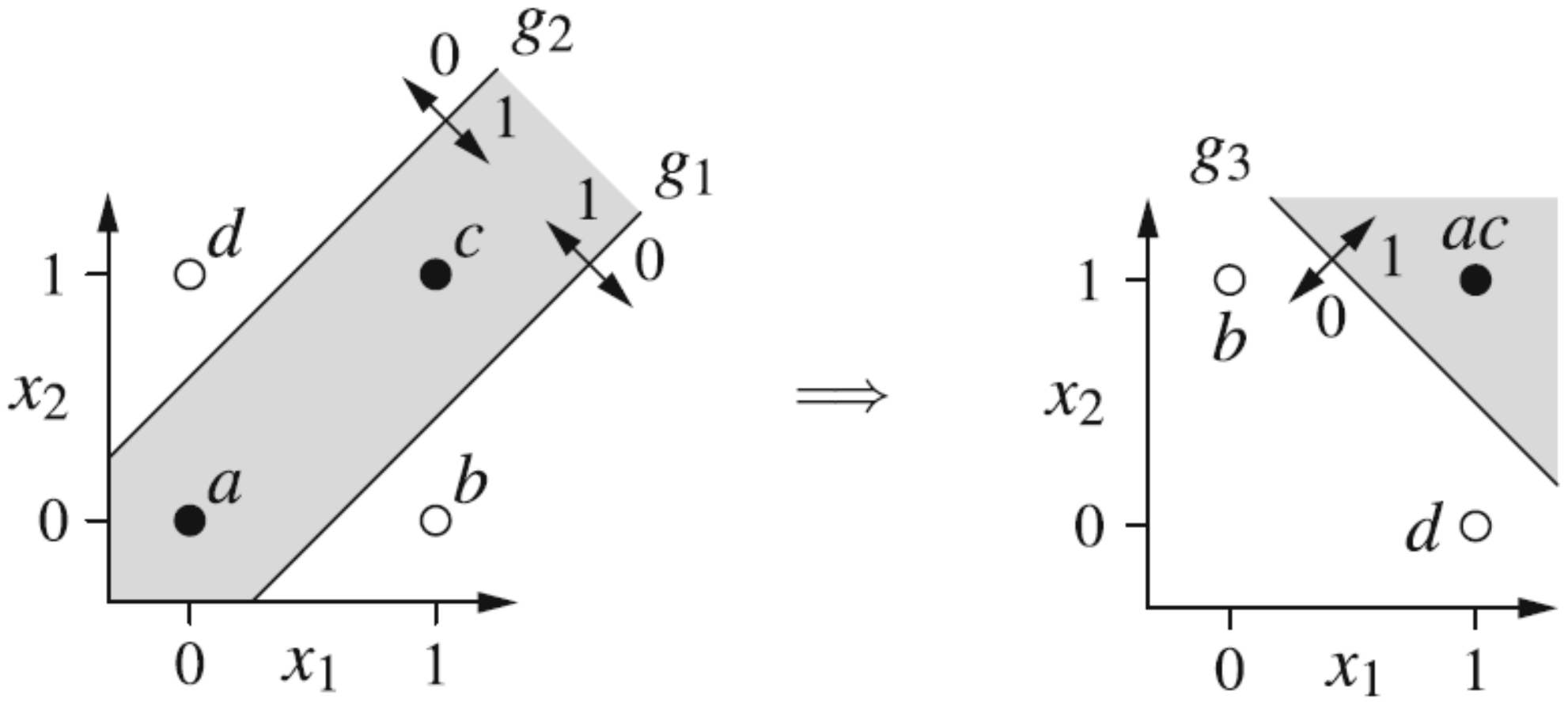
شبکه نورون های پایه

- تمام توابع بولی با هر تعداد ورودی دلخواه را می توان با شبکه نورون های پایه محاسبه کرد.
- شبکه دو لایه می تواند توابع بولی را پیاده سازی کند.
- برای پیاده سازی ترکیب دو شرطی (↔) شبکه نورون پایه با دو ورودی x_1 و x_2 دارای آستانه و وزن های زیر در نظر می گیریم.



$$x_1 \leftrightarrow x_2 \Longrightarrow$$

$$(x_1 \rightarrow x_2) \wedge (x_2 \rightarrow x_1)$$

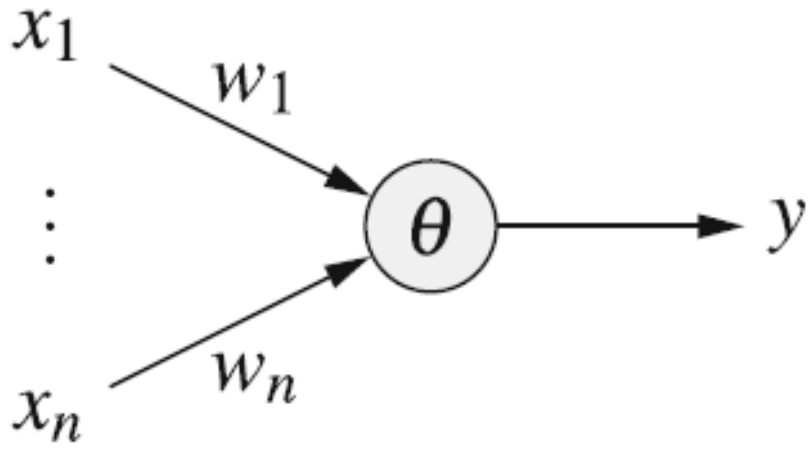


شکل هندسی پیاده سازی برای ترکیب دو شرطی بصورت یک شبکه از نورون های پایه

مفهوم وزن (Weights)

- وزن‌ها پارامترهایی هستند که قدرت و تأثیر هر اتصال بین نورون‌ها را تعیین می‌کنند.
- نقش وزن‌ها در شبکه:

* وزن‌ها نشان‌دهنده اهمیت هر ورودی خاص برای محاسبات نورون هستند.
* وزن بالاتر یعنی ورودی تأثیر بیشتری و وزن پایین‌تر یعنی ورودی تأثیر کمتر بر خروجی نورون دارد.
* در طول فرآیند آموزش، وزن‌ها به‌گونه‌ای به‌روزرسانی می‌شوند که خطای پیش‌بینی کاهش یابد.

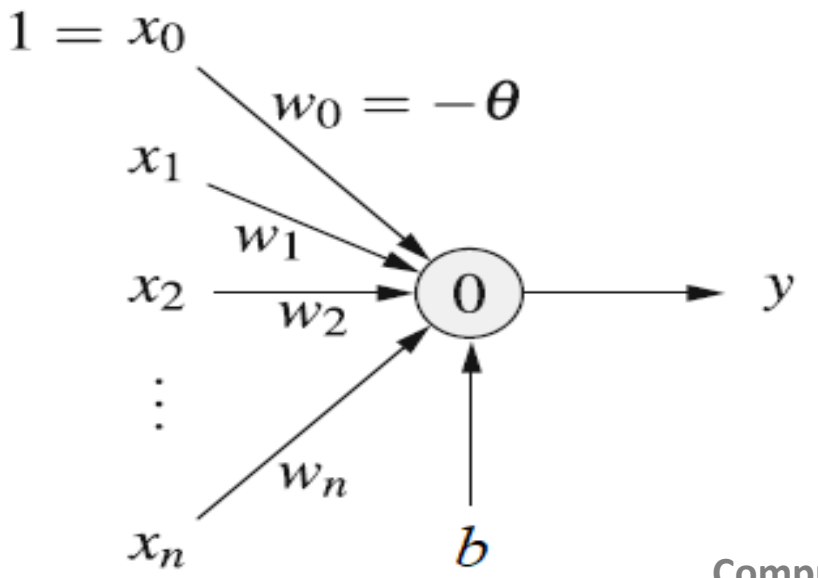


$$y = \begin{cases} 1 & \text{if} \quad \sum_{i=1}^{n} w_i x_i \geq \theta, \\ 0 & \text{otherwise.} \end{cases}$$

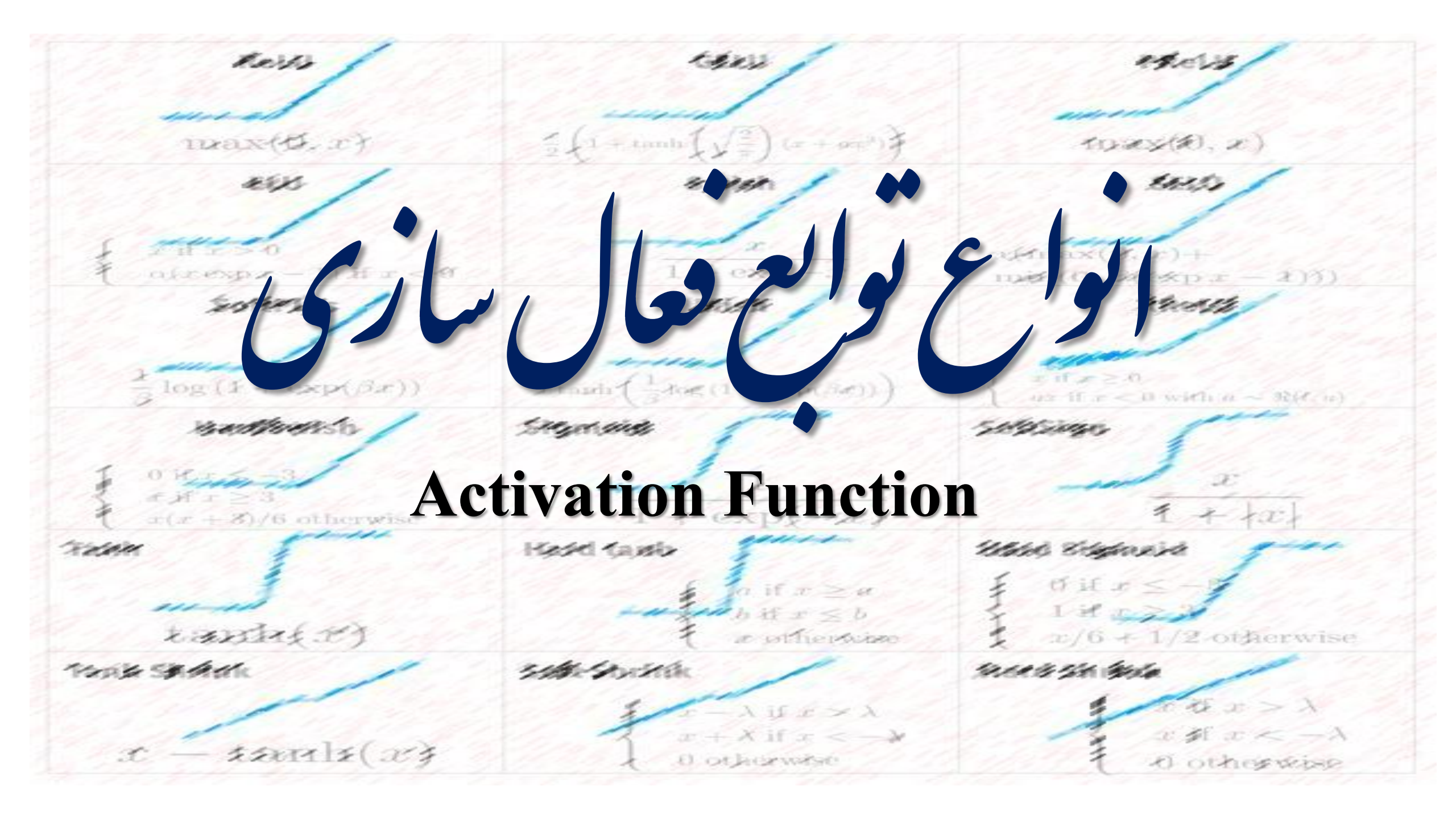
بایاس (Bias)

- بایاس یک مقدار ثابت است که به مقدار محاسبه‌شده توسط نورون اضافه می‌شود.
- نقش بایاس در شبکه:
  - بایاس به نورون اجازه می‌دهد خروجی را حتی زمانی که همه ورودی‌ها صفر هستند، تنظیم کند.
  - این مقدار کمک می‌کند تا شبکه بتواند به صورت انعطاف‌پذیرتر مرز تصمیم‌گیری خود را تغییر دهد.

برای مثال، بدون بایاس، مرز تصمیم همیشه از مبدا (صفر) عبور می‌کند، اما با وجود بایاس می‌تواند از هر نقطه‌ای در فضای ویژگی‌ها عبور کند.



$$y = \left[\sum_{i=0}^{n} w_i(t)x_i(t) \pm b\right]$$

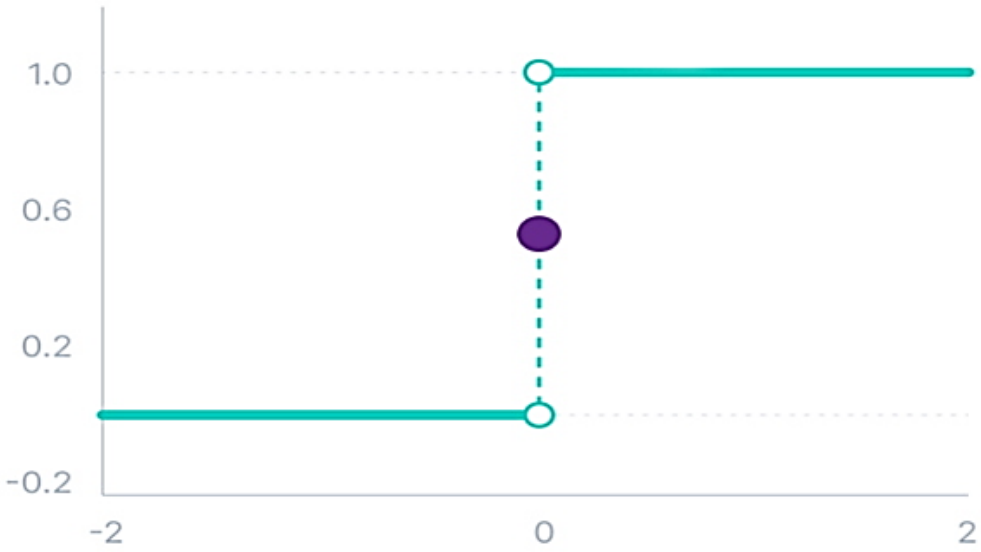


انواع توابع فعال سازی

Activation Function

تابع فعال سازی پله دودویی

- **تابع پله دودویی (Binary Step Function)**



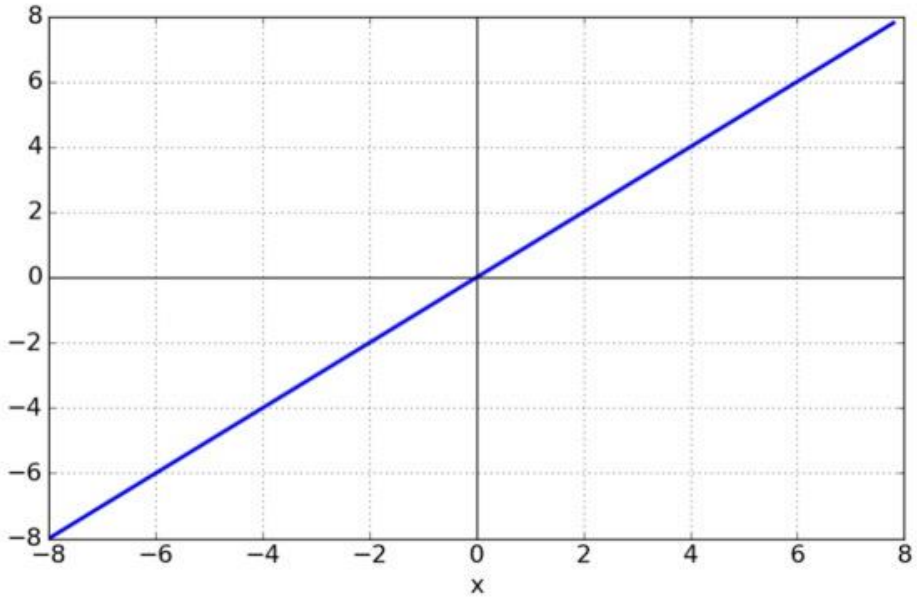
$$f(x): \begin{cases} 0 & for \quad x < 0 \\ 1 & for \quad x \geq 0 \end{cases}$$

* ورودی تابع پله دودویی با مقدار حد آستانه مقایسه می‌شود. اگر مقدار ورودی از مقدار حد آستانه بیشتر باشد، گره فعال خواهد شد در غیر این صورت غیرفعال باقی می‌ماند.

* محدودیت‌ها:
  - تابع پله دودویی نمی‌تواند خروجی‌هایی با مقادیر چندگانه تولید کند.
  - مشتق تابع پله دودویی برابر با صفر است که این ویژگی، مانعی برای روند انتشار رو به عقب محسوب می‌شود.

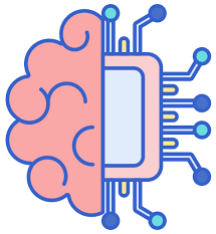
تابع فعال سازی خطی

تابع فعال سازی خطی (Linear Activation Function)



$$f(x) = x$$

* تابع همانی (Identify Function) مجموع وزنی ورودی را بدون هیچ تغییری به خروجی منتقل می‌کند.
* استفاده فقط در مدل‌های رگرسیون خطی
* محدودیت‌ها:
  - عدم استفاده در مرحله انتشار به عقب، زیرا مشتق آن یک عدد ثابت است و هیچ رابطه‌ای با ورودی x ندارد.
  - خروجی این تابع فعالسازی در لایه آخر شبکه با خروجی تابع فعالسازی در لایه اول برابر است.
  - وقتی شبکه دارای پارامترهای زیاد و پیچیده باشد، این نوع تابع فعالسازی عملکرد خوبی ندارد.



توابع فعال سازی غیرخطی

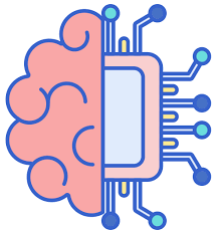
تابع فعالسازی غیرخطی (Non-linear Activation Function)

* این نوع توابع بیشترین استفاده را در شبکه های عصبی دارند. تعمیم‌پذیری و تطبیق‌پذیری مدل با انواع مختلف داده با استفاده از توابع فعالسازی غیرخطی آسان می‌شود.

* ویژگی‌های اصلی این توابع فعالسازی عبارت‌اند از:

- این توابع مشکل مربوط به مرحله انتشار به عقب را حل کردند. به عبارتی، توابع غیرخطی در مرحله انتشار رو به عقب می‌توانند مشخص کنند کدام وزن گره ورودی به تشخیص نهایی مدل می‌تواند کمک بهتری کند.

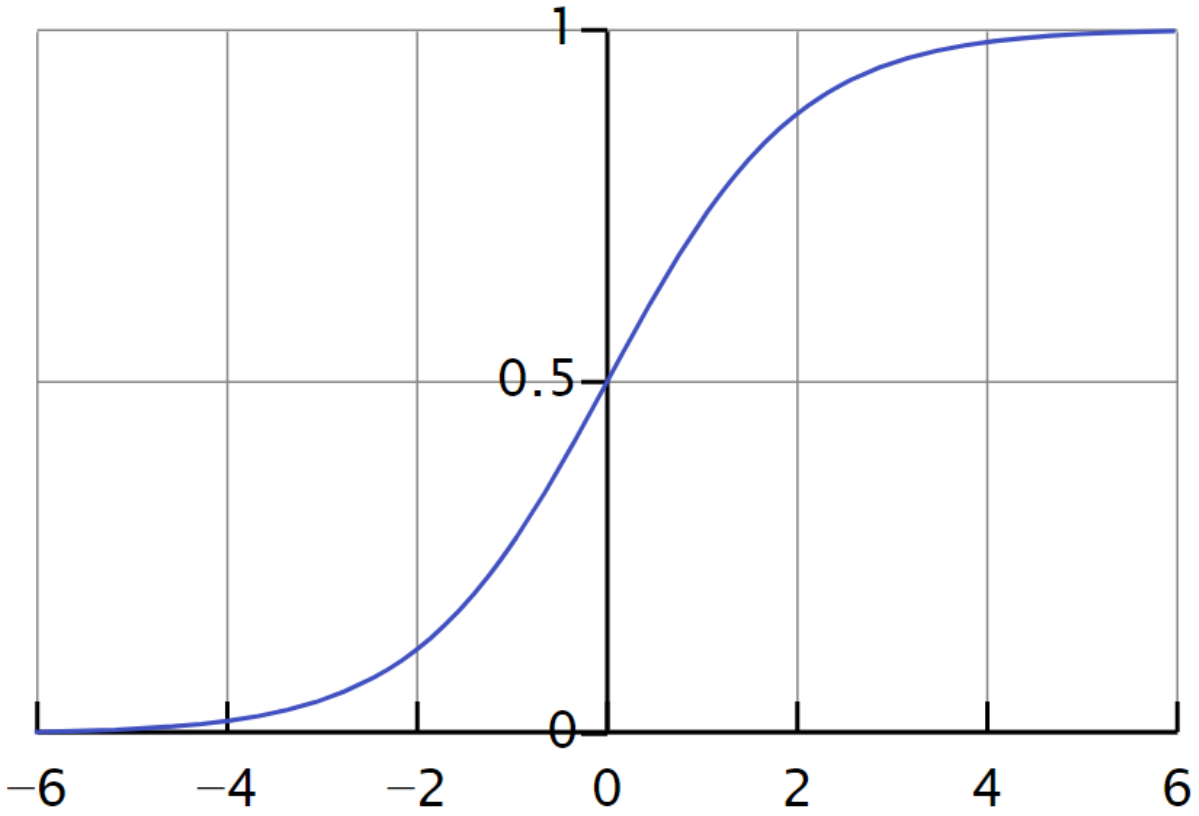
- با استفاده از این توابع می‌توان مسائل مربوط به خروجی‌های چندگانه را حل کرد. به عبارتی، خروجی توابع غیرخطی، ترکیب غیرخطی از ورودی‌ها است که از لایه‌های متعدد عبور کرده است.



انواع تابع فعال ساز غیر خطی

- تابع فعال سازی سیگموئید (Sigmoid | Logistic)
- تابع فعال سازی تابع تانژانت هذلولوی (Tanh | Hyperbolic Tangent)
- تابع فعال سازی یکسوساز (ReLU)
- تابع فعال سازی یکسوساز پارامتریک (Parametric ReLU)
- تابع فعال سازی واحدهای خطی نمایی (Exponential Linear Units | ELU)
- تابع فعال سازی (Softmax)
- تابع فعال سازی (Swish)

تابع فعالسازی غیرخطی سیگموئید

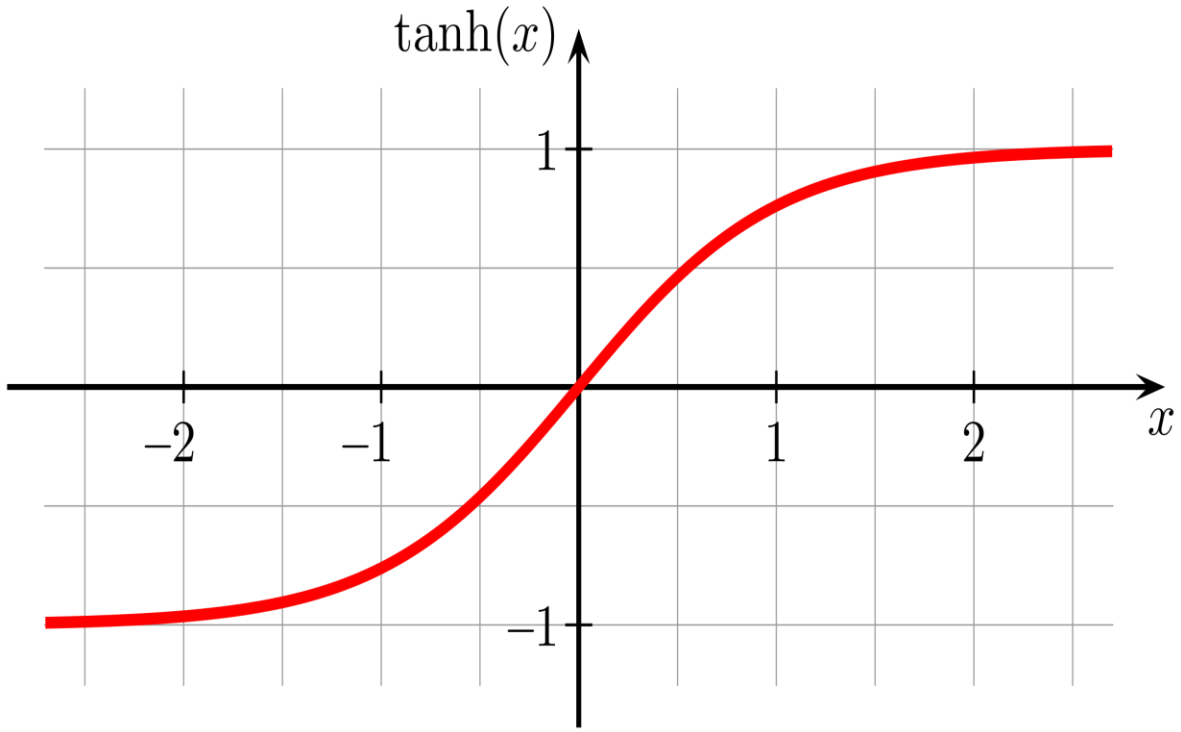


$$f(x) = \frac{1}{1 + e^{-x}}$$

- این تابع ورودی خود را به مقداری در بازه 0 تا 1+ تبدیل می‌کند. هرچه مقدار ورودی بزرگ‌تر باشد، مقدار خروجی این تابع به 1+ نزدیک‌تر شده و اگر مقدار ورودی کوچک باشد (عدد منفی)، مقدار خروجی این تابع به عدد 0 نزدیک‌تر می‌شود.

- تابع سیگموید، تابعی یکنوا (Monotonic) محسوب می‌شود اما مشتق این تابع، تابع یکنوا نیست.

تابع فعالسازی غیر خطی تانژانت هذلولوی

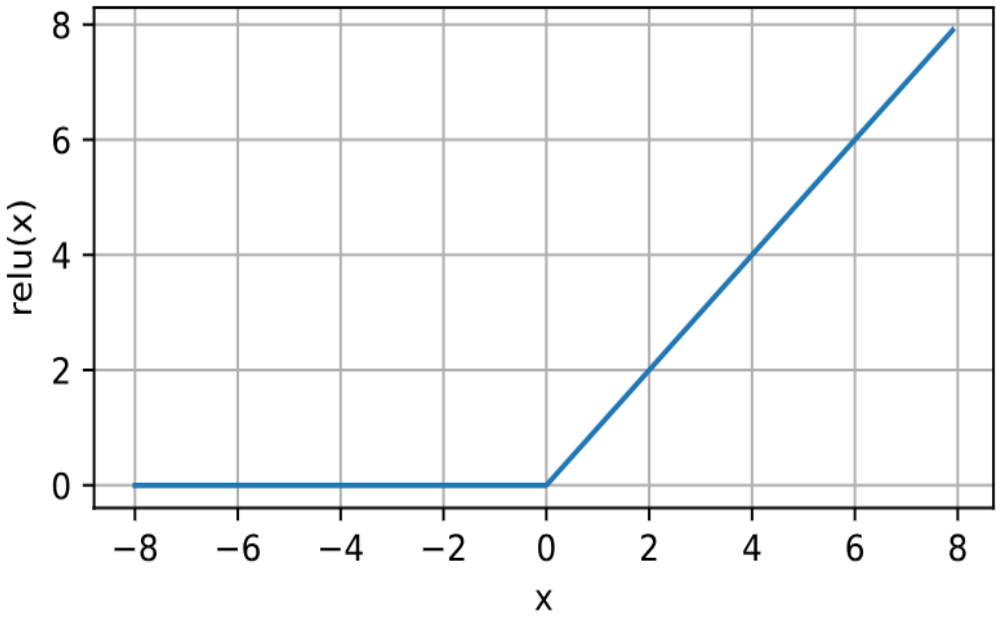


$$f(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

- تابع تانژانت هذلولوی (Hyperbolic Tangent) یا Tanh شبیه تابع سیگموئید است.
- تابع یکنواخت است اما مشتق آن یکنواخت نیست.
- تنها تفاوت این تابع با تابع سیگموئید، بازه خروجی این تابع است که به محدوده بین 1- تا 1+ نگاشت می‌کند.

تابع فعالسازی غیرخطی یکسوساز

- تابع واحد خطی اصلاح شده (**Re**ctified **L**inear **U**nit) یا ReLU یکی از رایج‌ترین توابع فعالسازی در:
  - شبکه های عصبی کانولوشنی (**C**onvolutional **N**eural **N**etworks)
  - یادگیری عمیق

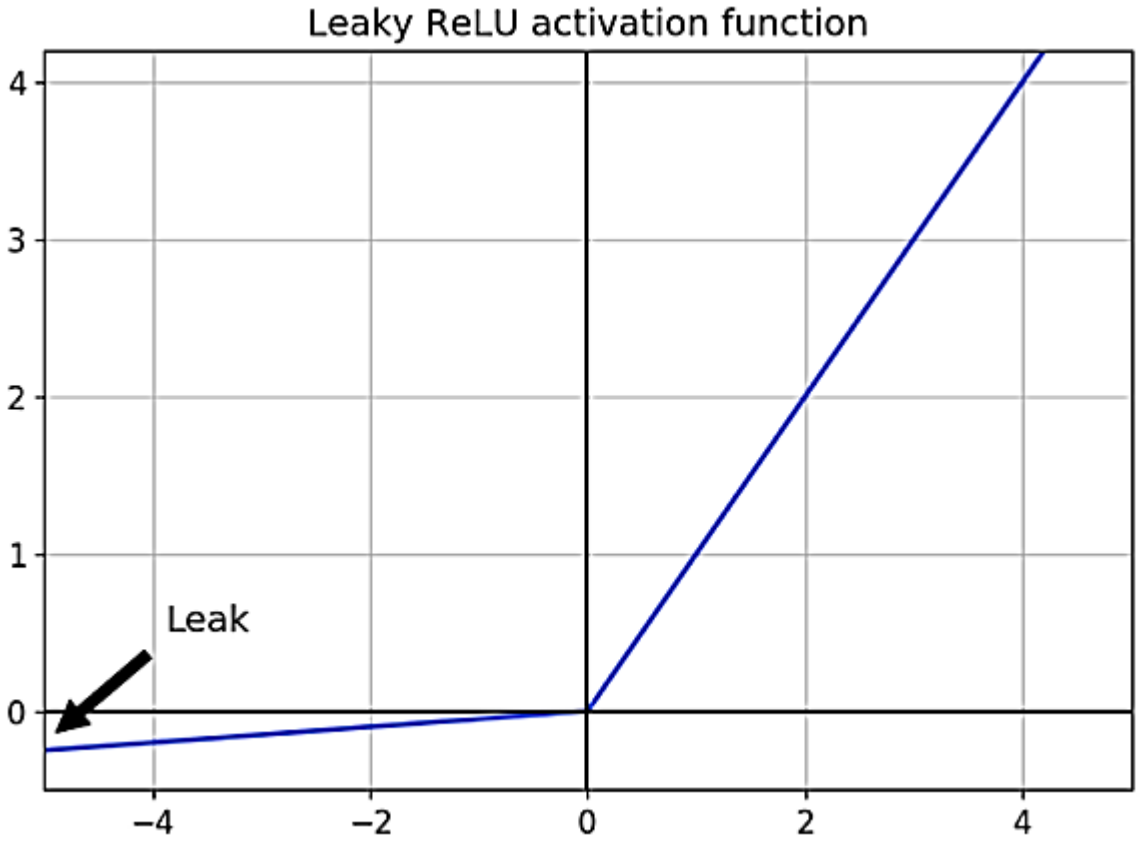


$$f(x) = \max(0, x)$$

- تابع مشتق‌پذیر است و می‌توان آنرا در مرحله انتشار رو به عقب استفاده کرد.

- این تابع تمامی گره‌ها را همزمان فعال نمی‌کند. گره‌ها زمانی غیرفعال هستند که مقدار ورودی این تابع کمتر از صفر باشد.

تابع فعالسازی غیرخطی یکسوساز پارامتریک



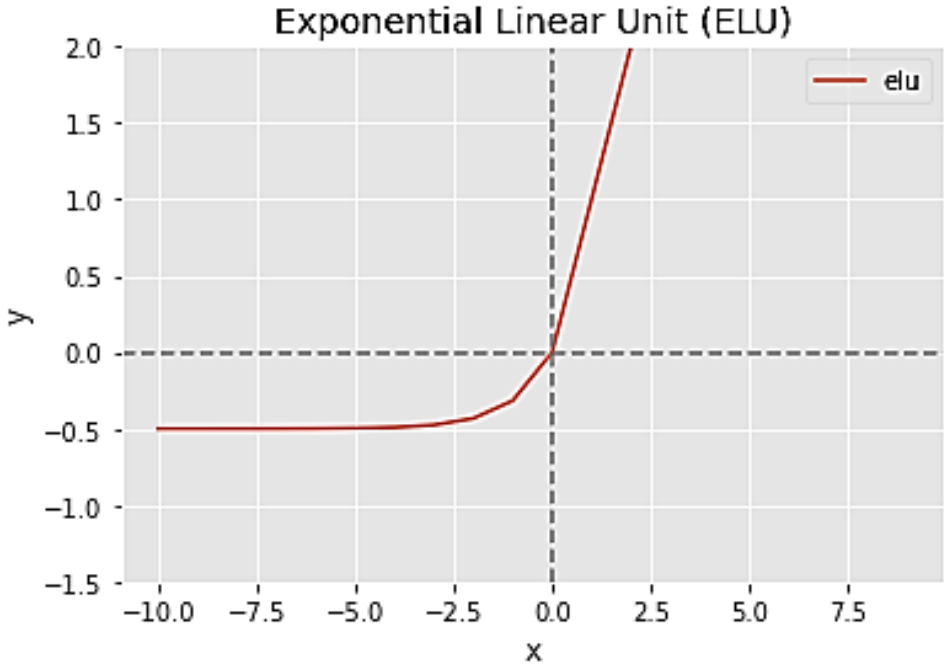
- این تابع فعالسازی، نسخه بهبودیافته از تابع فعالسازی یکسوساز ReLU است که مشکل زوال در تابع ReLU را حل می‌کند. در این تابع برای ناحیه منفی، شیب ملایم مثبتی به اندازه a درنظر گرفته می‌شود.

$$f(x) = max(ax, x)$$

تابع فعالسازی غیرخطی ELU

$$ELU(\alpha, x) = \begin{cases} x, & \text{if } x \geq 0 \\ \alpha(e^x - 1), & \text{otherwise} \end{cases}$$

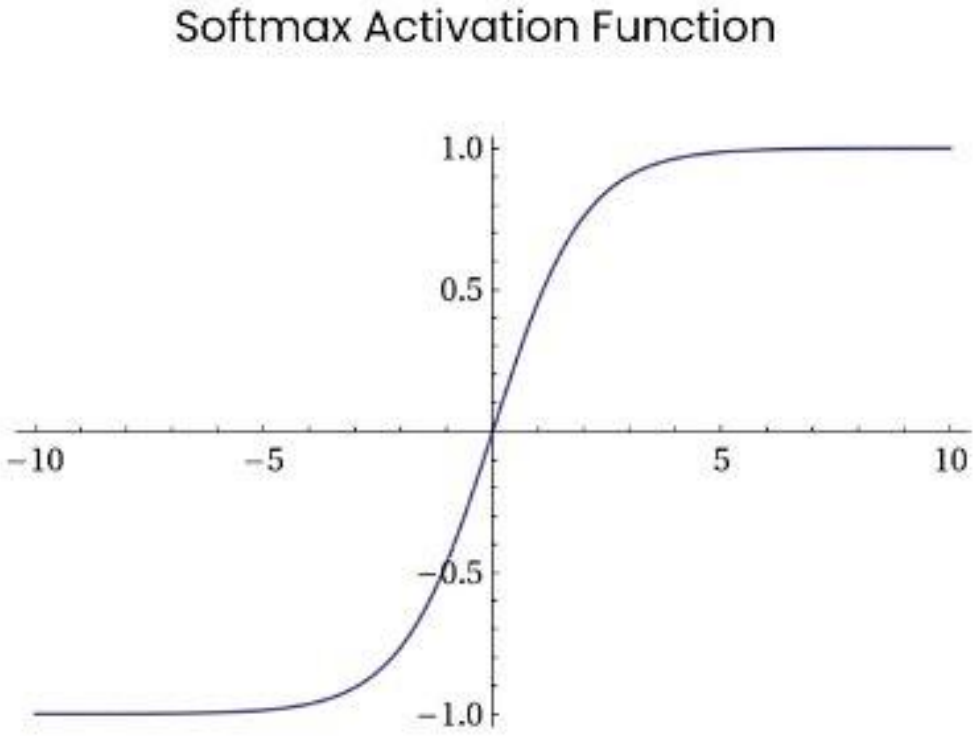
- تابع فعالسازی ELU یا تابع فعالسازی واحدهای خطی نمایی یکی از انواع تابع فعالسازی ReLU به حساب می‌آید.



- در تابع فعالسازی Parametric ReLU نمودار توابع در ناحیه منفی، به‌صورت خط مستقیم است. تابع فعالسازی ELU برای این ناحیه، منحنی لگاریتمی تعریف می‌کند.

تابع فعالسازی غیرخطی Softmax

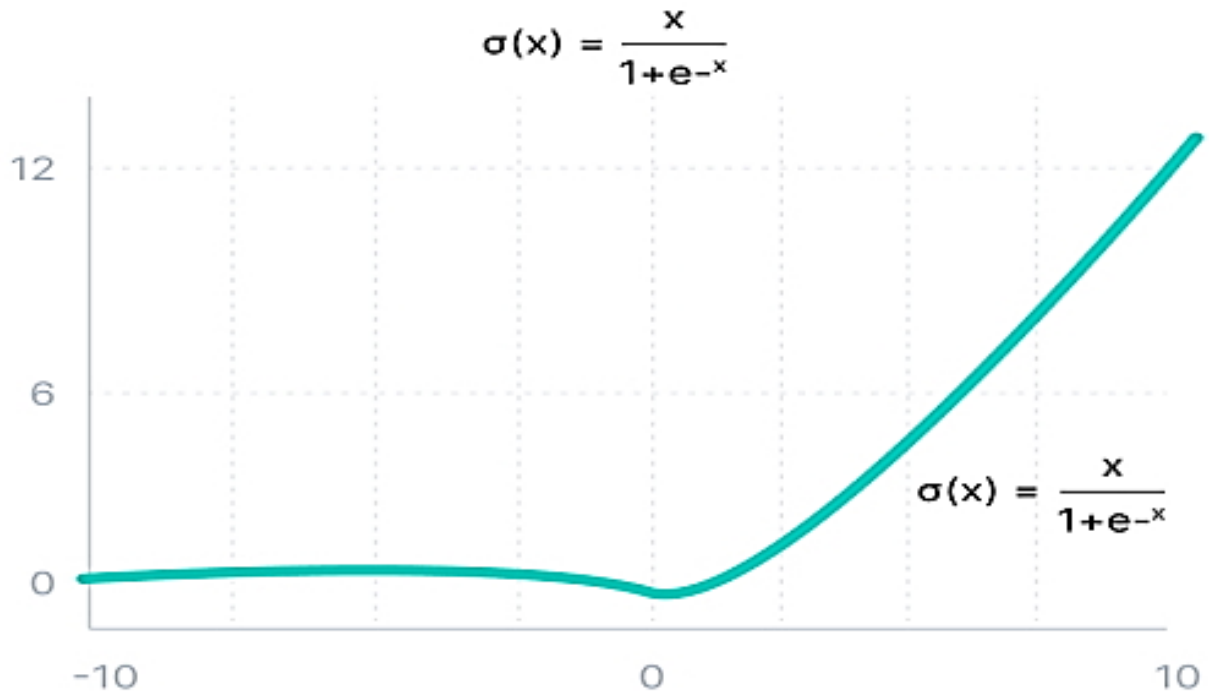
- تابع فعالسازی Softmax نوعی از تابع سیگموئید به حساب می‌آید. این تابع ترکیبی از چندین تابع سیگموئید است که احتمالات نسبی را محاسبه می‌کند. به عبارتی، تابع Softmax احتمالات چندین کلاس را مشخص می‌کند.



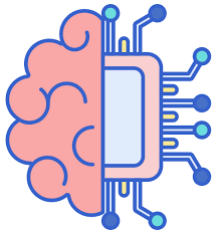
$$Softmax(z_i) = \frac{exp(z_i)}{\Sigma_j exp(z_j)}$$

تابع فعالسازی غیرخطی Swish

- این تابع را محققان گوگل ارائه کرده‌اند و به لحاظ کارآیی، عملکرد بهتری نسبت به تابع فعالسازی ReLU در شبکه های عمیق دارد. از تابع فعالسازی Swish در مسائلی نظیر دسته‌بندی تصاویر و ترجمه ماشین استفاده می‌شود.

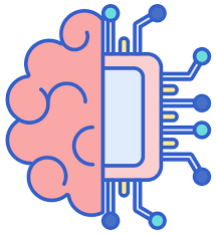


$$f(x) = x * sigmoid(x) = x * \frac{1}{1+e^{-x}}$$



شبکه عصبی تک‌لایه (Single Layer Neural Network)

- یک شبکه عصبی تک‌لایه شامل:
  - یک لایه ورودی برای دریافت داده‌ها.
  - یک لایه خروجی برای تولید خروجی‌ها. این شبکه‌ها هیچ لایه پنهانی (Hidden Layer) ندارند.
- ویژگی‌ها:
  - ساده‌ترین نوع شبکه‌های عصبی است.
  - فقط قادر به حل مسائل خطی جداناپذیر است.
  - معمولاً از یک یا چندین نورون (Perceptron) موازی در لایه خروجی تشکیل شده است.
- معایب:
  - توانایی مدل‌سازی مسائل غیرخطی مانند XOR را ندارد.
  - به دلیل محدودیت‌های ساختاری، برای داده‌های پیچیده مناسب نیست.



شبکه عصبی چندلایه (Multi-Layer Neural Network)

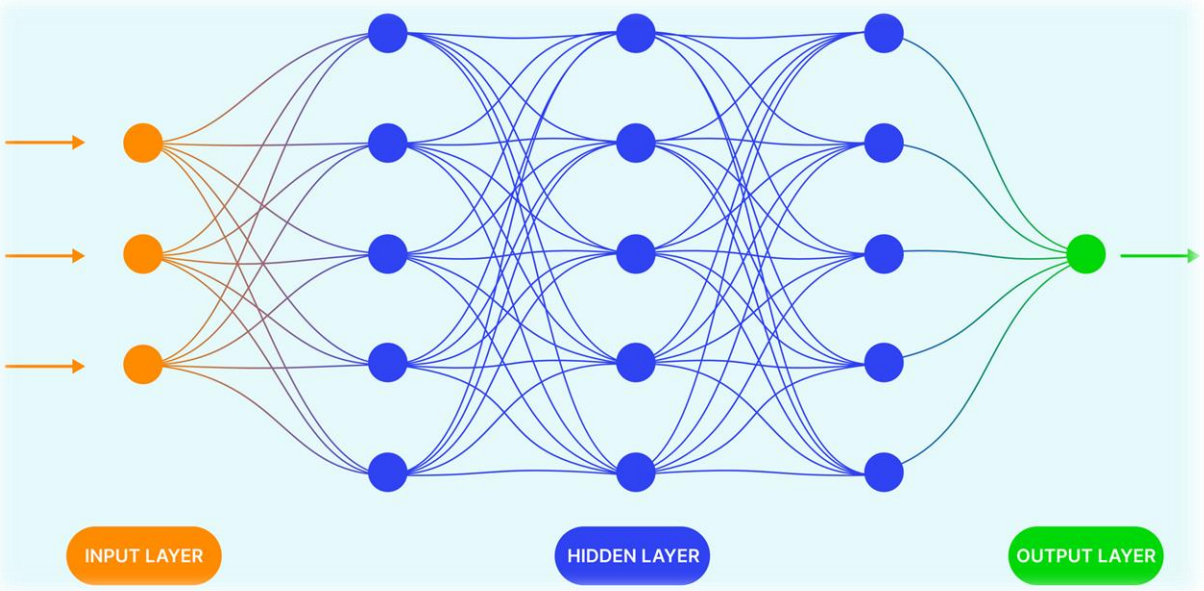
- یک شبکه عصبی چندلایه شامل:
  - یک لایه ورودی برای دریافت داده‌ها.
  - یک یا چند لایه پنهان (Hidden Layer) برای پردازش و استخراج ویژگی‌های پیچیده.
  - یک لایه خروجی: برای تولید نتایج نهایی.
- ویژگی‌ها:
  - مسائل خطی و غیرخطی مثل تشخیص تصاویر، پیش‌بینی سری زمانی و مسائل طبقه‌بندی را مدل‌سازی می‌کند.
  - با لایه‌های پنهان شبکه می‌تواند روابط پیچیده بین ورودی‌ها و خروجی‌ها را یاد بگیرد.
  - معمولاً از توابع فعال‌سازی غیرخطی استفاده می‌کند.
- معایب:
  - پیچیدگی محاسباتی بیشتر.
  - نیازمند داده‌های بیشتر و زمان آموزش طولانی‌تر.

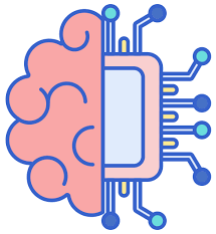
مقایسه بین شبکه عصبی تک‌لایه و چندلایه

ویژگی	شبکه تک‌لایه	شبکه چندلایه
لایه‌ها	فقط یک لایه خروجی	شامل لایه‌های ورودی، پنهان، و خروجی
توانایی مدل‌سازی	فقط مسائل خطی	مسائل خطی و غیرخطی
توابع فعال‌سازی	معمولاً خطی یا تابع پله‌ای	از توابع غیرخطی مانند سیگموید
کاربردها	ساده، مانند عملگرهای منطقی AND	پیچیده، مانند تشخیص الگو و پیش‌بینی
میزان محاسبات	کم	زیاد

لایه های شبکه عصبی (لایه ورودی)

- **لایه ورودی (Input Layer)**
  - اولین لایه شبکه عصبی که داده‌های خام (ویژگی‌های ورودی) را از محیط خارجی دریافت می‌کند.
  - تعداد نورون‌های این لایه برابر با تعداد ویژگی‌های ورودی داده‌ها است.
  - این لایه فقط داده‌ها را بدون هیچگونه پردازشی به لایه‌های بعدی منتقل می‌کند.

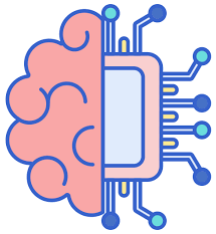




لایه های شبکه عصبی (لایه مخفی)

لایه‌های مخفی (Hidden Layers)

* میان لایه ورودی و لایه خروجی قرار دارند و پردازش و استخراج ویژگی‌های پیچیده را انجام می‌دهند.
* ممکن است یک یا چند لایه مخفی در شبکه وجود داشته باشد که به نوع مسئله و طراحی شبکه بستگی دارد و به صورت تجربی تعیین می‌شود.
* تعداد لایه‌های مخفی عمق شبکه عصبی را تعیین می‌کند. شبکه‌هایی با چندین لایه مخفی را شبکه‌های عصبی عمیق (Deep Neural Networks) می‌نامند.
* تعداد زیاد نورون‌ها در لایه‌های مخفی ممکن است قدرت مدل را افزایش دهد اما در عین حال، خطر بیش برازش (Overfitting) نیز وجود دارد.



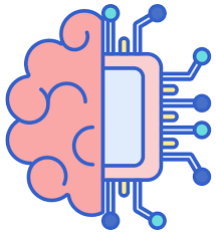
لایه های شبکه عصبی (لایه خروجی)

لایه خروجی (Output Layer)

* آخرین لایه در شبکه است که مقادیر خروجی لایه‌های قبلی را پردازش کرده و خروجی قابل تفسیر تولید می‌کند.
* تعداد نورون‌های لایه خروجی به نوع مسئله بستگی دارد:
  - در مسائل طبقه‌بندی دودویی (Binary Classification) و رگرسیون معمولاً ۱ نورون.
  - در مسائل طبقه‌بندی چند کلاسه: تعداد نورون‌ها برابر با تعداد کلاس‌ها.
* در مسائل رگرسیون، خروجی معمولاً یک مقدار عددی است.

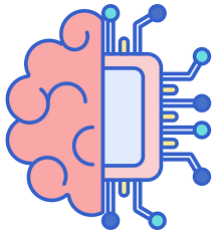
جمع بندی لایه های شبکه عصبی

لایه	نقش اصلی	ویژگی‌ها
ورودی (Input)	دریافت داده خام و ارسال آن به لایه‌های بعدی	تعداد نورون‌ها برابر با تعداد ویژگی‌های ورودی
مخفی (Hidden)	پردازش داده‌ها و استخراج روابط و ویژگی‌های پیچیده	تعداد و ساختار قابل تنظیم بر اساس مسئله
خروجی (Output)	ارائه نتایج پیش‌بینی شده یا کلاس بندی شده به صورت نهایی	تعداد نورون‌ها بسته به نوع خروجی (طبقه‌بندی یا رگرسیون)



پیش‌خور (Feedforward)

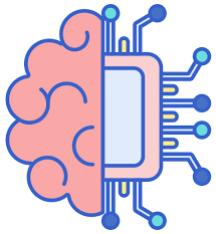
- فرآیندی که در آن داده‌ها از لایه ورودی به لایه‌های مخفی (در صورت وجود) و سپس به لایه خروجی، بدون هیچ بازگشت یا حلقه‌ای در جریان داده‌ها، هدایت می‌شوند. در این فرآیند، هر نورون در هر لایه به نورون‌های لایه بعدی متصل است و داده‌ها از ورودی به خروجی جریان پیدا می‌کنند.
- مراحل پیش‌خور:
  - دریافت ورودی‌ها
  - ضرب ورودی‌ها در وزن‌ها
  - جمع کردن نتایج
  - اعمال تابع فعال سازی
  - انتقال خروجی به نورون‌های لایه بعدی
  - خروجی نهایی



پیش‌خور (Feedforward)

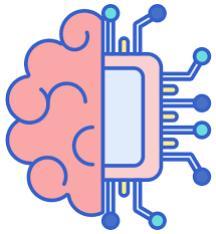
- اهمیت پیش‌خور در فرآیند آموزش
  - در طول آموزش، پارامترهای وزن و بایاس بروزرسانی می‌شوند تا خروجی شبکه با مقدار واقعی (هدف) مطابقت بیشتری پیدا کند.
  - پس از هر فاز پیش خور، خطا محاسبه شده و برای بروزرسانی وزن‌ها و بایاس‌ها در فاز پس‌انتشار (Backpropagation) استفاده می‌شود.

- پیش‌خور یکی از مراحل اصلی در یادگیری و پیش‌بینی در شبکه‌های عصبی است و بدون آن، شبکه نمی‌تواند به خوبی عمل کند.



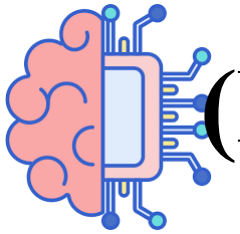
الگوریتم پس‌انتشار (Backpropagation)

- یکی از مهمترین روش‌ها برای آموزش شبکه‌های عصبی مصنوعی است.
- از مفاهیم گرادیان نزولی (Gradient Descent) و قاعده زنجیره‌ای (Chain Rule) برای محاسبه و بروزرسانی وزن‌ها و بایاس‌ها استفاده می‌کند.
- هدف اصلی پس‌انتشار، کاهش مقدار خطا یا تابع هزینه جهت پیش‌بینی‌های دقیق‌تر شبکه است.
- **مراحل:**
  - **پیش‌خور:** داده‌ها از ورودی به خروجی منتقل می‌شوند و خروجی پیش‌بینی شده $(\hat{y})$ تولید می‌شود.
  - **محاسبه خطا:** تفاوت بین خروجی پیش‌بینی شده $(\hat{y})$ و مقدار واقعی (y) محاسبه می‌شود.
  - **انتشار خطا به عقب:** گرادیان‌های خطا نسبت به وزن‌ها و بایاس‌ها محاسبه می‌شوند.
  - **بروزرسانی وزن‌ها:** وزن‌ها و بایاس‌ها با استفاده از گرادیان نزولی بروزرسانی می‌شوند.



نکات مهم الگوریتم پس‌انتشار (Backpropagation)

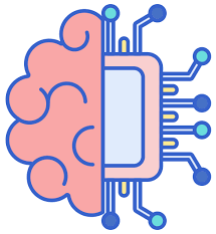
- نکات مهم در پس‌انتشار:
  - قاعده زنجیره‌ای (Chain Rule):
    - پس‌انتشار از قاعده زنجیره‌ای استفاده می‌کند تا گرادیان خطا را نسبت به وزن‌ها و بایاس‌ها در هر لایه محاسبه کند.
  - مقداردهی اولیه وزن‌ها:
    - وزن‌ها در ابتدای آموزش معمولاً به صورت تصادفی مقداردهی می‌شوند. انتخاب مقدار مناسب می‌تواند در همگرایی بهتر و سریع‌تر مؤثر باشد.
  - تابع فعال‌سازی:
    - انتخاب تابع فعال‌سازی مناسب مانند Sigmoid یا ReLU یا Tanh نقش کلیدی در محاسبه گرادیان‌ها دارد.
  - مشکل گرادیان ناپدید شونده (Vanishing Gradient):
    - در شبکه‌های عمیق، گرادیان ممکن است در حین انتقال به لایه‌های اولیه کاهش یابد. این مشکل می‌تواند یادگیری شبکه را کند یا متوقف کند.



مزایا و چالش‌های الگوریتم پس‌انتشار (Backpropagation)

- مزایا:
  - بهینه‌سازی مؤثر وزن‌ها و بایاس‌ها.
  - کاربرد گسترده در آموزش شبکه‌های عصبی عمیق.

- چالش‌ها:
  - **گرادیان ناپدید شونده یا انفجاری:** در شبکه‌های بسیار عمیق، گرادیان ممکن است خیلی کوچک یا بزرگ شود.
  - **نیاز به تنظیم دقیق نرخ یادگیری:** نرخ یادگیری باید به دقت انتخاب شود.
  - **تکیه بر داده‌های برچسب دار:** پس‌انتشار نیاز به داده‌های آموزشی با برچسب دارد.



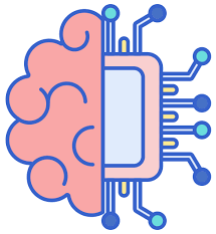
خطا یا تابع هزینه (Loss Function)

- معیاری است که نشان می‌دهد پیش‌بینی‌های شبکه تا چه اندازه با مقادیر واقعی (Labels) فاصله دارند. تابع هزینه معمولاً یک مقدار اِسکالر است و به صورت ریاضی بیانگر میزان خطا در پیش‌بینی‌ها است.
- نمونه‌هایی از توابع هزینه
  - MSE (Mean Squared Error)
    - برای مسائل رگرسیون

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - y_{\hat{i}})^2$$

  - Cross-Entropy Loss
    - برای مسائل طبقه‌بندی

$$Loss = -\frac{1}{n}\sum_{i=1}^{n} y_i \cdot log\big(p(y_i)\big) + (1 - y_i) \cdot log\big(1 - p(y_i)\big)$$



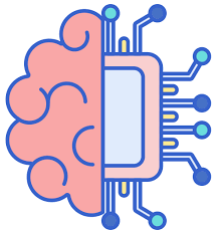
الگوریتم گرادیان نزولی (Gradient Descent)

- هدف یادگیری در شبکه‌های عصبی کاهش خطاست که از طریق روش‌های بهینه‌سازی مانند گرادیان نزولی (Gradient Descent) انجام می‌شود.
- گرادیان نزولی یک روش تکراری است که به شبکه کمک می‌کند طی فرآیند زیر، مقدار تابع هزینه را با بروزرسانی وزن‌ها و بایاس‌ها کاهش دهد.

* **محاسبه گرادیان (شیب):** گرادیان نشان دهنده میزان تغییر تابع هزینه نسبت به هر وزن است $\frac{\partial L}{\partial \omega}$

* **بروزرسانی وزن‌ها:** وزن‌ها به سمت عکس شیب حرکت می‌کنند $\omega = \omega - \eta \frac{\partial L}{\partial \omega}$
  - η نرخ یادگیری (Learning Rate) است.

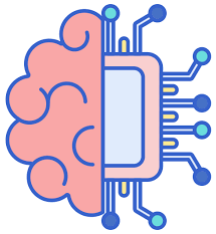
* **تکرار:** این فرآیند برای هر وزن در شبکه تکرار می‌شود تا زمانی که تابع هزینه به مقدار مطلوبی برسد.



نرخ یادگیری (Learning Rate)

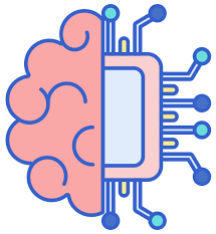
- یک مقدار عددی کوچک است که اندازه گام‌ها را در فرآیند گرادیان نزولی کنترل می‌کند.

- **اهمیت:**
  - اگر نرخ یادگیری خیلی بزرگ باشد، شبکه ممکن است از مینیمم محلی عبور کند و همگرا نشود.
  - اگر نرخ یادگیری خیلی کوچک باشد، فرآیند یادگیری کند می‌شود.



انواع گرادیان نزولی ساده

- Stochastic Gradient Descent (SGD)
  - * از یک نمونه (یک داده) برای محاسبه گرادیان استفاده می‌کند.
  - * **مزایا:** سریع‌تر است و حافظه کمتری نیاز دارد.
  - * **معایب:** نوسان زیادی دارد و ممکن است بهینه محلی پیدا شود.
- Batch Gradient Descent
  - * از کل داده‌های آموزشی برای محاسبه گرادیان استفاده می‌کند.
  - * **مزایا:** دقیق‌تر است.
  - * **معایب:** محاسباتی سنگین برای دیتاست های بزرگ.
- Mini-Batch Gradient Descent
  - * از گروه‌های کوچکی از داده‌ها استفاده می‌کند.
  - * **مزایا:** ترکیب مزایای دو روش قبلی.



انواع گرادیان نزولی بهبود یافته

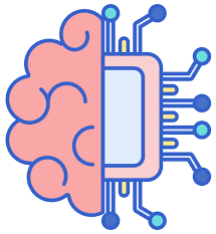
- Momentum
  - **مشکل:** در گرادیان نزولی ساده، ممکن است شبکه در مینیمم محلی گیر کند یا به کندی به هدف برسد.
  - **راه‌حل:** شتاب (Momentum) به گرادیان نزولی اضافه می‌شود تا شبکه بتواند از مینیمم محلی عبور کند و سریع‌تر همگرا شود.

- RMSProp (Root Mean Square Propagation)
  - **مشکل:** در مسائل پیچیده، اندازه گرادیان‌ها ممکن است در طول آموزش بسیار متفاوت باشد.
  - **راه‌حل:** اندازه گرادیان‌ها را با استفاده از مقادیر میانگین مربع گرادیان نرمال‌سازی می‌کند.

- Adam (Adaptive Moment Estimation)
  - یکی از قدرتمند ترین و پُرکاربرد ترین الگوریتم‌های بهینه‌سازی است که ترکیبی از دو روش قبلی است.

مقایسه روش های گرادیان نزولی

روش	مزایا	معایب
BGD	ساده و قابل فهم	محاسباتی سنگین برای داده‌های بزرگ
SGD	سریع و کم حافظه	نوسانات زیاد
Mini-Batch	تعادل بین دقت و سرعت	وابسته به اندازه دسته
Momentum	بهبود سرعت همگرایی و عبور از مینیمم محلی	ممکن است پیچیدگی محاسباتی افزایش یابد
RMSProp	تنظیم خودکار اندازه گام‌ها	ممکن است در برخی مسائل تنظیم بیش از حد رخ دهد
Adam	ترکیب مزایای Momentum و RMSProp	نیاز به تنظیم آَبَر پارامترها



انتخاب بهترین روش گرادیان نزولی

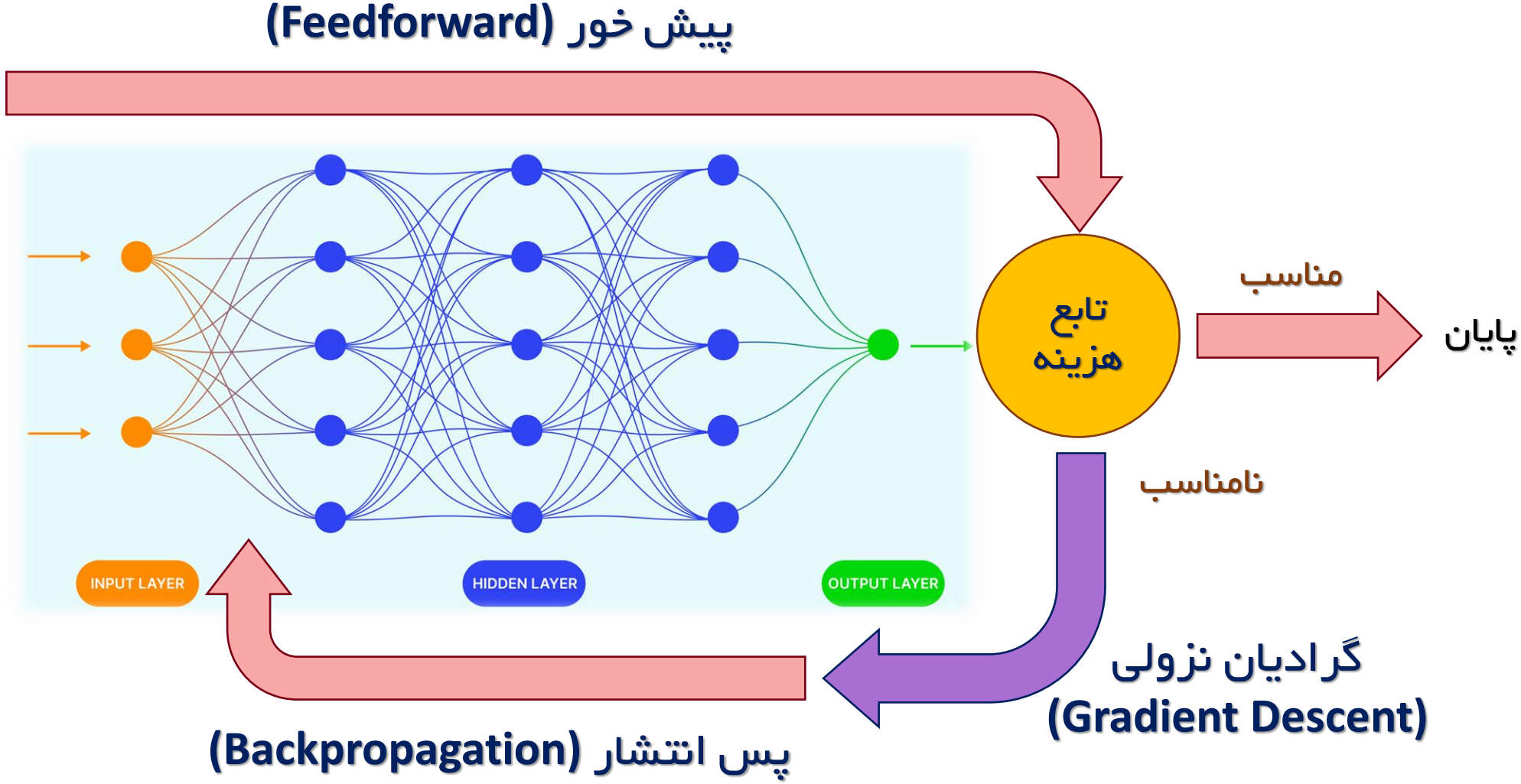
- انتخاب روش بهینه‌سازی بستگی به موارد زیر دارد:

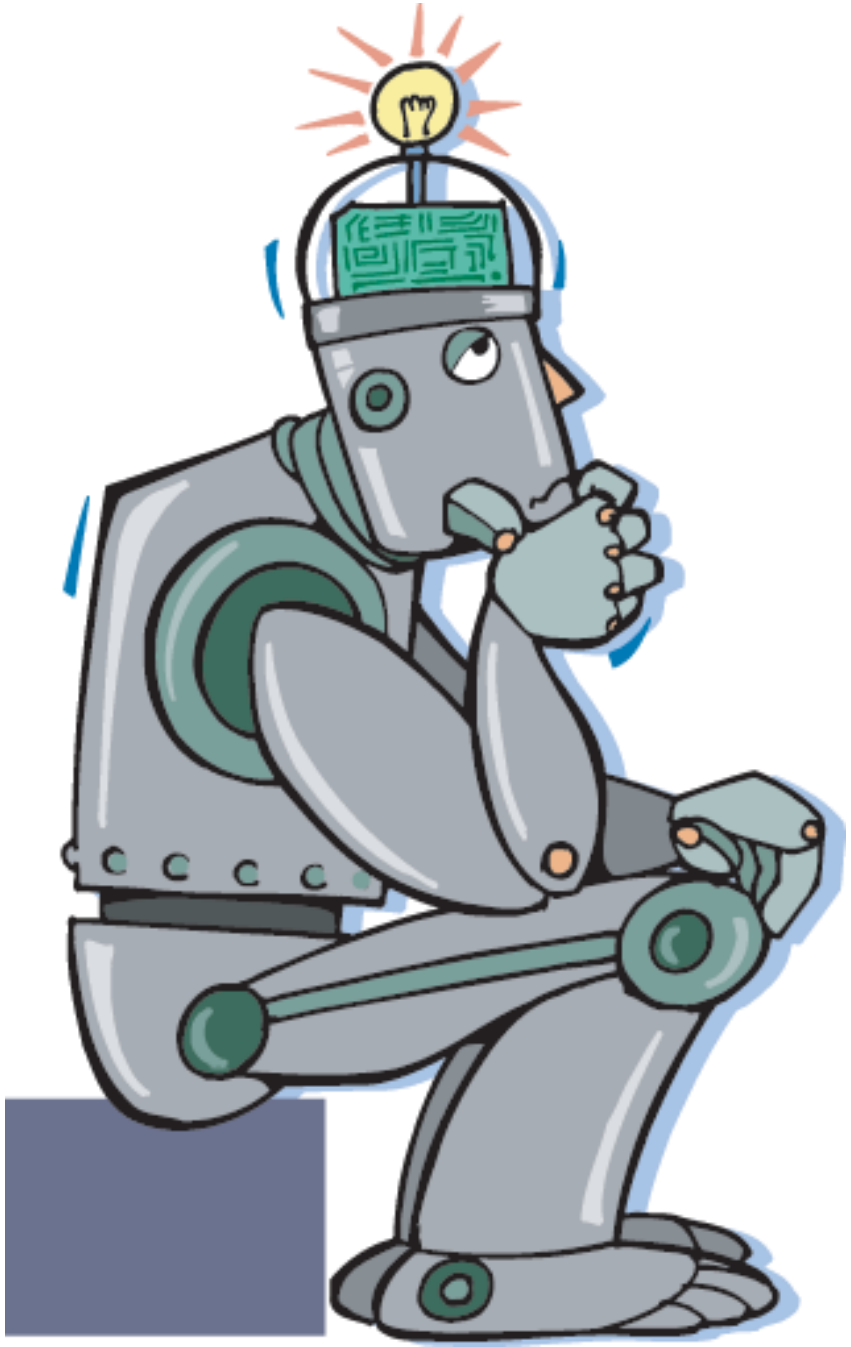
* اندازه داده:
  - برای دیتاست های کوچک، Gradient Descent مناسب است.
  - برای دیتاست های بزرگ، Mini-Batch یا Adam بهتر عمل می‌کنند.

* نوع مسئله:
  - مسائل طبقه‌بندی و رگرسیون پیچیده: Adam
  - مسائل با تغییرات گرادیان شدید: RMSProp

* محاسبات:
  - اگر سخت‌افزار محدود باشد، روش‌های سبک‌تر مانند SGD یا Mini-Batch پیشنهاد می‌شود.

فرآیند کلی آموزش شبکه عصبی





سؤال؟