**مجله کنترل**
ISSN (print) 2008-8345
ISSN (online) 2538-3752
**مقاله پژوهشی**
جلد۱۷، شماره ۲، تابستان ۱۴۰۲، صفحه ۷۹-۴۷

# مروری بر سیستم‌های هوشمند (شبکه‌های عصبی) از دیدگاه تئوری کلاسیک و کاربرد آن‌ها در مدل‌سازی و کنترل سیستم‌های پیچیده

محمد لطفی[۱]، محمدباقر منهاج[۲]

[۱] دانشجوی مقطع دکتری مهندسی برق، گروه کنترل، دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)، ایران lotfi.m@aut.ac.ir

[۲] استاد، دانشکدهٔ مهندسی برق، گروه کنترل، دانشگاه صنعتی امیرکبیر (پلی تکنیک تهران)، ایران menhaj@aut.ac.ir

دریافت: ۱۴۰۲/۰۵/۰۶ پذیرش: ۱۴۰۲/۰۶/۲۶

**چکیده:** امروزه با پیشرفت صنعت و فناوری، شاهد پیچیده‌تر شدن روزافزون سیستم‌ها هستیم. این پیچیدگی در صنعت، مستلزم پیشرفت موازی برای کنترل‌کننده‌ها نیز بوده است که نیازمندی به سیستم‌های کنترل یا کنترل‌کننده‌های پیشرفته و هوشمند را چندین برابر کرده است. یکی از مهم‌ترین معیارها در طراحی هر سیستم کنترلی، شناخت دقیق سیستم یا به بیان دقیق‌تر مدل‌سازی سیستم است. با در نظر گفتن این دو چالش، در این مقاله به بررسی سیستم‌های هوشمند و به طور خاص شبکه‌های عصبی از دیدگاه تئوری کلاسیک و کاربرد آن‌ها در مدل‌سازی و کنترل سیستم‌های پیچیده پرداخته می‌شود. در این راستا، ابتدا عنصر اصلی شبکه‌های عصبی یعنی نرون معرفی شده و انواع مدل آن (جمعی و شعاعی) ارائه می‌گردد. سپس انواع شبکه‌های عصبی از قبیل شبکه‌های عصبی پرسپترون چندلایه (MLP)، شبکه‌های عصبی شعاعی پایه (RBF) و شبکه‌های عصبی بازگشتی (RNN) تشریح می‌گردند و در مورد تعداد لایه‌ها و تعداد مناسب نرون‌های لایه‌های پنهان این شبکه‌های عصبی برای کاربردهای مختلف و به‌خصوص جهت تقریب توابع غیرخطی بحث می‌شود. در ادامه سعی می‌شود که پلی بین مفاهیم و اصطلاحات شبکه‌های عصبی (دنیای هوشمند) و دنیای کلاسیک زد و از دیدگاه تئوری کلاسیک آن‌ها را موردبررسی قرار داد. نشان داده می‌شود که از دیدگاه تئوری کلاسیک، یک شبکه عصبی را می‌توان به‌عنوان یک ساختار مدل و وزن‌ها و بایاس‌های آن را به‌عنوان پارامترهای مجهول این ساختار در نظر گرفت. در شبکه‌های عصبی، از الگوریتم‌های یادگیری برای تعیین پارامترهای مجهول شبکه (وزن‌ها و بایاس‌ها) استفاده می‌شود. در این راستا، الگوریتم‌های یادگیری شبکه‌های عصبی از دیدگاه تئوری کلاسیک و به‌طور خاص در ارتباط با بهینه‌سازی عددی موردمطالعه و بررسی قرار می‌گیرند و یک پل ارتباطی میان الگوریتم‌های یادگیری و روش‌های بهینه‌سازی عددی زده خواهد شد و در ادامه مهم‌ترین الگوریتم‌های یادگیری شبکه‌های عصبی به همراه مزایا و معایب آن‌ها معرفی می‌شوند. در پایان نیز به دو مورد از کاربردهای مهم شبکه‌های عصبی، مدل‌سازی و کنترل، پرداخته می‌شود و برای هر یک از کاربردهای مذکور، مثال‌های مختلفی ارائه می‌گردد تا کارایی شبکه‌های عصبی به‌وضوح مشاهده شود.

**کلمات کلیدی:** سیستم‌های هوشمند و شبکه‌های عصبی – تئوری کلاسیک - شناسایی و کنترل.

# An overview of intelligent systems (neural networks) from the perspective of classical theory and their application in modeling and control of complex systems

**Mohammad Lotfi, Mohammad Bagher Menhaj**

**Abstract:** In recent years, with the advancement of industry and technology, we see that systems are becoming more and more complex. This complexity in the industry has required a parallel progress in control systems (controllers), which has multiplied the need for advanced and intelligen

t controllers. One of the most important criteria in the design of any control system is the accurate knowledge of the system or, more precisely, the modeling of the system. Considering these two challenges, this paper examines intelligent systems and specifically neural networks from the perspective of classical theory and their application in modeling and control of complex systems. At first, the basic element of neural networks, i.e. neuron, is introduced and its types of models (collective and radial) are also presented. Then the types of neural networks such as multilayer perceptron neural networks (MLP-NN), radial basis neural networks (RBF-NN) and recurrent neural networks (RNN) are described and the appropriate number of layers and also the appropriate number of neurons in the hidden layers in neural networks for different applications and especially for the approximation of nonlinear functions (modeling) are discussed. Then, it is tried to build a bridge between the concepts of neural networks (intelligent world) and the concepts of classical world and analyze neural networks from the perspective of classical theory. It is shown that from the point of view of classical theory, a neural network can be considered as a model structure and its weights and biases as unknown parameters of this structure. In neural networks, learning algorithms are used to determine the unknown parameters of the network (weights and biases). Therefore, learning algorithms of neural networks are studied from the point of view of classical theory and specifically in relation to numerical optimization and a bridge will be built between learning algorithms and numerical optimization methods. Also, the most important neural network learning algorithms are introduced along with their advantages and disadvantages. In the end, two important applications of neural networks, modeling and control, are discussed and for each of these applications, several illustrative examples are presented and simulated to show the effectiveness of neural networks.

**Keywords:** Intelligent systems and neural networks, Classical theory, Identification and control.

## ۱- مقدمه

امروزه مزایای متعددی را می‌توان برای استفاده هوشمندانه از مهندسی کنترل برشمرد که از مهم‌ترین آن‌ها می‌توان به افزایش کیفیت زندگی یا محصول تولیدی، کاهش مواد زائد، کاهش آلودگی، افزایش ایمنی، کاهش مصرف انرژی و ... اشاره نمود. همچنین، به‌وضوح می‌توان مشاهده کرد که مفاهیمی از قبیل "فیدبک یا بازخورد [۱]" و "کنترل [۲]" از جنبه‌های متعدد اجتماعی – تکنولوژی، از نقش بسیار مهمی برخوردار هستند. باور عموم بر این است که مفهوم کنترل بیشتر از آن‌که دارای ماهیت علمی باشد، با مقوله مهندسی در ارتباط است، لیکن برای اینکه بتوان این مفهوم را با موفقیت پروژه‌های چالشی‌تر به کار برد، تئوری قوی‌تری در این زمینه موردنیاز است و همین توجه تئوری در محافل علمی و دانشگاهی، در طول سالیان متمادی به مباحث گسترده‌ای در خصوص شکاف تئوری و عمل انجامیده است [۷-۱]. نقطه اوج آن را می‌توان در یکی از شماره‌های "مجله سیستم‌های کنترل IEEE" (جلد ۱۹، شماره۶، ۱۹۹۹) [۸] مشاهده نمود.

در تمام علوم مهندسی، بیشتر کارها با تکیه بر مدل ریاضی انجام می‌شود؛ به‌طوری‌که در تمام رشته‌ها و علوم، برای شناخت و تحلیل آنچه در واقعیت اتفاق می‌افتد، نیاز به استفاده از کامپیوتر به‌عنوان ابزار قدرتمندی برای شبیه‌سازی رفتار سیستم و تجزیه‌وتحلیل آن وجود دارد. مدل ریاضی یک سیستم طبق تعریف یک مجموعه معادله است که رفتار سیستم را دقیقاً، یا حداقل به خوبی، توصیف می‌کند. تعیین مدل برای یک سیستم فیزیکی و مطالعه خواص آن دلیلی است که علم مدل‌سازی به خاطر آن به وجود آمده است. مدل‌سازی ریاضی نه تنها در علوم طبیعی مانند فیزیک، شیمی، اخترشناسی، زیست‌شناسی، هواشناسی و علوم مهندسی مانند علوم رایانه، کنترل و ... کاربرد دارد، بلکه در علوم اجتماعی مانند علم اقتصاد، روان‌شناسی و جامعه‌شناسی نیز کاربرد گسترده‌ای دارد. مدل‌سازی به پژوهشگران کمک می‌کند تا یک سیستم را تحلیل کرده و رفتار آن را پیش‌بینی کنند. محور اصلی مدل‌سازی یک سیستم فیزیکی می‌تواند قوانین فیزیکی حاکم بر سیستم، استفاده از نتایج یک آزمایش روی سیستم فیزیکی و یا ترکیبی از هر دو باشد که به ترتیب به روش‌های مدل‌سازی تحلیلی (جعبه سفید)، آزمایشی (جعبه سیاه) و ترکیبی (جعبه خاکستری) معروف هستند. برای توصیف دقیق یک سیستم عملاً استفاده از روابط ریاضی دقیق کار معمول و عرفی است، ولی همیشه ساختن یک مدل دقیق برای سیستم به کمک روابط ریاضی مقدور نیست. ممکن است دسته‌ای از پارامترهای یک سیستم در دسترس یا قبال اندازه‌گیری نباشند و یا اینکه دستگاه‌های اندازه‌گیری (سنسورها)، دقت لازم برای این اندازه‌گیری را نداشته باشند. از این رو تخمین، بخش لاینفک فرآیند مدل‌سازی آزمایشی یا شناسایی یک سیستم است (شناسایی عنوانی کلی برای دانشی است که مهندسان کنترل برای استخراج مدل ریاضی یک سیستم با استفاده از نتایج یک آزمایش، داده‌های ورودی و خروجی، استفاده می‌کنند). با توجه به اینکه همیشه روابط ریاضی برای توصیف یک

1. Feedback
2. Control

سیستم امکان‌پذیر نیست، نیاز به روش‌های غیرپارامتریک برای شناسایی سیستم‌ها، نیاز روزافزاونی در صنعت به شمار می‌آید. از مهم‌ترین روش‌های غیرپارامتریک می‌توان به الگوریتم‌های هوشمند، شبکه‌های عصبی و منطقی اشاره کرد. هر کدام از این روش‌ها معایب و مزایایی نسبت به یکدیگر در شناسایی سیستم‌ها دارند. دقت در شناسایی و سرعت، از مواردی است که سبب برتری یکی بر دیگری می‌شود. شبکه‌های عصبی (مصنوعی) در واقع یک مدل الگو گرفته شده از مغز انسان هستند که به خاطر خواص خوبی که در تقریب تابع از خود نشان داده‌اند، موردتوجه قرار گرفته‌اند.

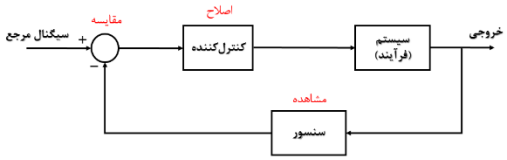
از طرفی، پیشرفت صنعت و فناوری و پیچیده‌تر شدن روزافزون سیستم‌ها نیازمندی به سیستم‌های کنترل هوشمند را چندین برابر کرده است. اهمیت بالای کنترل اتوماتیک در صنعت و زندگی امروزی بر کسی پوشیده نیست. با وجود اینکه توسعه روزافزون اتوماسیون، منافع زیادی را برای بشر به ارمغان آورده است ولی همچنان مشکل اتوماسیون فرایندهای پیچیده، باقی است. بیشتر فرآیندهای صنعتی رفتار غیرخطی از خود نشان می‌دهند و کنترل فرآیندهای با تأخیرهای غیرقابل چشم‌پوشی، چالش مهم دیگری است. نیاز به استفاده از روش‌های هوشمند برای حل این دشواری‌ها به حدی حائز اهمیت است که در قالب انقلاب صنعتی چهارم از آن یاد شده است. بنابراین، سیستم‌های کنترل هوشمند زمینه‌ای به‌روز، بسیار جالب، الهام گرفته از طبیعت و ابزاری کارآمد در کنترل فرایندهای پیچیده و در راستای نیازهای صنعت مدرن محسوب می‌شود [۷-۵].

در این مقاله به مروری بر تئوری کنترل از دیدگاه کلاسیک و همچنین شناسایی و کنترل هوشمند (با استفاده از شبکه‌های عصبی) و کاربردهای وسیعی که در سیستم‌های پیچیده و صنعت دارند، پرداخته شده است. این مطالعه و بررسی از آن جهت حائز اهمیت است که می‌تواند برای پژوهش‌های آتی محققان به‌عنوان یک مرجع و راهنما از کارهای قبلی انجام شده و همچنین برای کاربردهای صنعتی مورداستفاده قرار بگیرد.

ساختار این مقاله به این صورت است که در بخش ۲، مفاهیم پایه تئوری کنترل، انواع کنترل‌کننده‌های کلاسیک و کارهای پیشین انجام شده در این زمینه ارائه شده است. در بخش ۳ به‌عنوان آشنایی با مقدمات و مفاهیم لازم از شبکه‌های عصبی مصنوعی، ابتدا نرون‌های مصنوعی و مدل آن‌ها معرفی خواهند شد. سپس شبکه‌های عصبی مصنوعی را معرفی خواهیم کرد که از اتصال چندین نرون مصنوعی تشکیل می‌شود و نحوه آموزش این شبکه‌ها موردبررسی قرار می‌گیرند. پس از تشریح و آشنایی با این مفاهیم، در بخش ۴ به نحوه مدل‌سازی سیستم‌های پیچیده با استفاده از شبکه‌های عصبی و در بخش ۵ به کنترل سیستم‌های پیچیده با استفاده از شبکه‌های عصبی پرداخته خواهد شد. در بخش ۶ نیز به جمع‌بندی مطالب پرداخته شده است.

## ۲- مروری بر تئوری کنترل از دیدگاه کلاسیک

کنترل و فیدبک یکی از مفاهیم اساسی دنیای امروز است که در بسیاری از حوزه‌ها کاربرد دارد. واژه پرمعنای کنترل که از دیدگاه نویسنده دوم مقاله به صورت "مشاهده – مقایسه – اصلاح" تعریف می‌شود، در بسیاری از ابعاد زندگی ما وارد شده است. کنترل رژیم غذایی، کنترل مالی، کنترل آفات، کنترل موتور، کنترل ربات، و صدها مورد دیگر مثال‌های بارزی از این کاربردها هستند. همچنین، می‌توان گفت: "قدرت بدون کنترل فاقد ارزش است". این جمله در بسیاری از زمینه‌ها از قبیل تکنولوژی و مسائل اجتماعی صحیح است. به‌عنوان مثال، وقتی یک شخص احساس سرما (مرحله حس کردن) دارد، لباس گرم‌تر می‌پوشد (مرحله تصمیم‌گیری و سپس اقدام کنترلی) و در نتیجه خود را گرم‌تر و شرایط راحت‌تری را بر خود فراهم می‌کند (هدف). این مثالی از فیدبک بیولوژیکی است که به علت تغییر در محیط پیرامون صورت می‌گیرد. در سیستم‌های صنعتی نیز به منظور اصلاح رفتار سیستم، حلقه "مشاهده – مقایسه – اصلاح" برای رسیدن به رفتار مطلوب به کار گرفته می‌شود. شکل ۱ ساختار عمومی یک سیستم کنترل با فیدبک را نشان می‌دهد که متشکل از سه بلوک اصلی به نام‌های سیستم (فرآیند)، کنترل‌کننده و سنسور (حس‌گر) است. مطابق این شکل، در ساختار عمومی کنترل با فیدبک، بلوک‌های کنترل‌کننده و سیستم (فرآیند) در مسیر مستقیم و در مسیر فیدبک، سنسورها و در برخی موارد تطبیق‌دهنده سیگنال[1] قرار دارند. این ساختار عمومی، معمولاً در کنترل انواع سیستم‌ها به صورت گسترده مشاهده می‌گردد.




شکل ۱. ساختار عمومی یک سیستم کنترل فیدبک

توجه داشته باشید که در سیستم‌های کنترل، مفهوم فیدبک بسیار حائز اهمیت است. اگر فرض کنیم که مسیر فیدبکی وجود ندارد، سیستم منحصراً به وسیله سیگنال ورودی (مرجع) راه‌اندازی می‌شود و بعد از عبور سیگنال از بلوک کنترل‌کننده، سیگنال خروجی سیستم تولید می‌شود. این نوع ساختار سیستم، "ساختار کنترل حلقه باز" نامیده می‌شود. توجه داشته باشید در شرایط ایده‌آل، استراتژی کنترل حلقه باز کارآمد خواهد بود؛ لیکن این امر منوط به داشتن مدلی دقیق از سیستم تحت مطالعه است که در عمل به دلیل وجود خطای مدل‌سازی و عدم‌قطعیت‌هایی از قبیل اغتشاشات وارده به سیستم امکان‌پذیر نیست. بنابراین برای کنترل موفق بایستی از یک ساختار "سیستم حلقه" بسته به جای "ساختار حلقه باز" استفاده نمود. از سیستم‌های حلقه بسته اغلب تحت عنوان "سیستم‌های کنترل با فیدبک" یاد می‌شود [۱، ۴، ۹].

### ۲-۱ کنترل کلاسیک و تاریخچه آن

1. Signal Conditioning

احتمالاً ساده‌ترین و اولین نمونه‌های یک سیستم کنترل اتوماتیک خودکار به ساعت آبی باستانی بندر اسکندریه در مصر یا قطب نمای قدیمی ساخته شده در بیش از دو هزار سال پیش در چین باز می‌گردد. سیستم گاورنر کنترل موتور بخار که در سال ۱۷۸۸ میلادی توسط جیمز وات[1] اختراع گردید [۱۰] نیز به عنوان اولین سیستم کنترل با فیدبک با دامنه کاربرد فراوان شناخته می‌شود. تحقیقات نظری در زمینه سیستم‌های کنترل با مطالعه مسائل پایداری سیستم‌ها شامل بررسی معادلات دیفرانسیل و با پیشگامی ماکسول[2] در ۱۸۶۸، روث[3] در سال ۱۸۷۴ و هورویس[4] در سال ۱۸۹۵ آغاز گردید. مسائل طراحی استراتژی کنترل ابتدا به وسیله مینوراسکی در سال ۱۹۲۲ پیشنهاد گردید [۱۱] که به واسطه آن، کنترل‌کننده سه ترمی معروف PID (تناسبی، انتگرال‌گیر، مشتق‌گیر) برای نخستین بار فرموله شد. الگوریتم‌های عملی برای تنظیم کنترل‌کننده PID به وسیله زیگلر[5] و نیکولز[6] در سال ۱۹۴۲ [۱۲] ارائه گردید که هنوز هم در مسائل مربوط به مهندسی کنترل کاربرد خود را حفظ نموده‌اند. چارچوب تحلیل حوزه فرکانسی سیستم‌های کنترل با فیدبک در سال ۱۹۳۲ توسط نایکوئیست[7] بنا نهاده شد [۱۳]. این روش به وسیله بود[8] در سال ۱۹۴۵ و نیکولز در سال ۱۹۴۷ توسعه یافت [۱۵-۱۴]. تحلیل مکان هندسی ریشه‌ها توسط ایوانس[9] در سال ۱۹۴۸ پیشنهاد گردید [۱۶] که مرحله مهم دیگری در طراحی و تحلیل سیستم‌های فیدبک و کنترل کلاسیک خطی بود.

معرفی روش اصل بیشینه در کنترل بهینه توسط پونترایگین[10] در سال ۱۹۵۶ [۱۷]، برنامه‌نویسی دینامیکی به وسیله بلمن[11] در سال ۱۹۵۷ [۱۸] و نمایش فضای حالت به وسیله کالمن[12] در سال ۱۹۵۹ [۱۹] افق جدیدی را به سوی دنیای سیستم کنترل کلاسیک گشود که بعدها تحت عنوان "تئوری کنترل مدرن" شناخته شد. از موفقیت‌های مهمی که توسط کالمن رقم خورد، می‌توان به رگولاتور بهینه خطی مربعی[13] (LQR) در سال ۱۹۵۹، رویت‌گر[14] حالت بهینه در سال ۱۹۶۰ و کنترل‌کننده بهینه گوسی خطی مربعی[15] (LQG) [۲۱-۲۰] اشاره نمود. در ادامه، دویل به این واقعیت دست یافت که احتمال دارد LQG حاشیه پایداری سیستم را کاهش دهد و به‌این‌ترتیب، تحقیق در مورد طراحی سیستم بهبود انتقال حلقه (LTR) آغاز گردید [۲۲]. همچنین، از مهم‌ترین مراجع و مقالات چاپ شده در زمینه کنترل بهینه می‌توان به موارد زیر اشاره نمود: کنترل بهینه کاربردی: بهینه‌سازی، تخمین و کنترل [۲۳]، تئوری کنترل بهینه [۲۴]، کنترل بهینه: مقدمه‌ای بر تئوری به همراه کاربردها [۲۵]، پایدارسازی و کنترل بهینه سیستم¬های غیرخطی با مدل نامعین و تضعیف اغتشاش [۲۶]، ایجاد الگوریتم¬های موازی در مقیاس مستقل زمان کند و سریع برای حل مسائل کنترل و فیلترینگ بهینه خطی در سیستم‌های استاندارد و غیر استاندارد آشفته تکین، پیوسته و گسسته، تصادفی و...[۲۷]، ارتباط تئوری کنترل هندسی با تئوری کنترل بهینه H2 و H-inf [۲۸]، ایجاد پلی میان بهینه¬سازی متداول با استفاده از حساب تغییرات و آنچه کنترل بهینه مدرن خوانده می¬شود [۲۹]، ارائه فرم تحلیلی کنترل پیش¬بین بهینه بدون نیاز به بهینه¬¬سازی آنلاین [۳۰]، معرفی سه وجه کنترل بهینه: برنامه¬ریزی پویا، اصل حداقل¬یابی پنتریاگین، روش¬های عددی بهینه¬سازی مسیر [۳۱]، ارائه پیش¬زمینه ریاضی، با فرمول‌بندی منسجم مسئله کنترل و بحث راجع به شرایط لازم بهینگی بر مبنای اصل حداکثر پنتریاگین، بیان کاربردها و تئوری حداقل زمان، حداقل انرژی و مسائل معیار مربعی [۳۲]، پیشرفت‌های جدید در تئوری تخمین بهینه و تکنیک‌های طراحی[۳۳]، بیان مفاهیم سیستم¬های خطی، کنترل بهینه و کنترل مقاوم به همراه مثال‌ها و مسائل ملموس [۳۴]، پوشش سه حوزه اصلی مهندسی کنترل: کنترل PID، مقاوم و بهینه [۳۵]، بررسی کنترل بهینه برای گروه بزرگی از مهندسان و دانشمندان که ریاضیدان نیستند[۳۶]، بیان کلیاتی از تئوری ریاضیاتی کنترل بهینه پروسه‌هایی که توسط معادلات دیفرانسیل یا نوع مشخصی از آن‌ها که معادلات دیفرانسیل حافظه¬دار هستند توصیف می¬شوند [۳۷]، بررسی کنترل بهینه سیستم¬های پیوسته و گسسته، برنامه¬ریزی پویا، بازی‌های دیفرانسیلی، کنترل تطبیقی بهینه و یادگیری تقویتی و مطالب دیگر [۳۸]، بررسی مسائل مهم کنترل بهینه غیرخطی ابعاد محدود و تئوری کنترل¬پذیری [۳۹]، پوشش روش¬های طراحی الگوریتم-های کنترل بهینه (یا شبه بهینه) به شکل سنتز سیستم¬های دینامیکی قطعی و تصادفی، با کاربرد در هوافضا، رباتیک و تکنولوژی سرومکانیک [۴۰]، بررسی روش¬های کنترل خودکار، با توجه ویژه به قوام، قابلیت تضمین پایداری در حضور نامعینی، روش¬های پیشرفته مواجهه با نامعینی و بهینه¬سازی حلقه کنترل، LQR، H-inf و....[۴۱]، بررسی کنترل بهینه معادلات دیفرانسیل تصادفی [۴۲]، بیان کنترل بهینه و تخمین بهینه برای سیستم¬های خطی و غیرخطی [۴۳]، مسئله کنترل بهینه برای سیستم¬های قطعی و تصادفی با پارامتر پیوسته یا گسسته [۴۴].

یک از شاخه‌های کنترل بهینه، کنترل بهینه سیستم‌های تصادفی یا به اختصار کنترل بهینه تصادفی است. از مهم‌ترین مراجع و مقالات چاپ شده در زمینه کنترل بهینه تصادفی می‌توان به موارد زیر اشاره نمود: کاربرد تئوری کنترل بهینه تصادفی [۴۵]، بررسی کاربرد تئوری کنترل تصادفی در کنترل رفتار دینامیکی پروسه¬ها تحت تأثیر اغتشاشات تصادفی [۴۶]، کنترل تصادفی خطی [۴۷]، بررسی تضعیف اغتشاش قطعی، کنترل تصادفی و کنترل تطبیقی سیستم¬های غیرخطی [۴۸]، حل مسئله تضعیف اغتشاش تصادفی (غیرخطی) هنگامی‌که باید حل سیستم با یک تابع مونوتون

1. James Watt
2. Maxwell
3. Routh
4. Hurwitz
5. Ziegler
6. Nichols
7. Nyquist
8. Bode
9 . Evans
10. Pontryagin
11. Bellman
12. Kalman
13. Linear Quadratic Regulator (LQR)
14. Observer
15. Linear Quadratic Gaussian (LQG)

سوپریموم کوواریانس نویز کراندار شود [۴۹]، بیان تئوری کنترل تصادفی به صورت تحلیل، بهینه¬سازی پارامتری و کنترل تصادفی بهینه [۵۰]، مطرح کردن کلیاتی در مورد کنترل تصادفی [۳۳]، بررسی مسائل کنترل پیش-بین مدل دارای قیود اکید یا احتمالاتی برای مواردی که نامعینی مدل تصادفی و ضربی وجود دارد [۵۱]، بررسی کنترل غیرخطی تصادفی H-inf/H2 [۵۲] ، بررسی بهینه‌سازی تصادفی، معادلات دیفرانسیل تصادفی و انتگرال¬گیری تصادفی و مسائل تصادفی در فیزیک و بیولوژی [۵۳]، بررسی مجموعه مقالات ارائه شده در دوازدهمین مدرسه زمستانی پروسه¬های تصادفی و کاربردهای آن‌ها [۵۴]، بررسی دسته¬ای از روش‌های فیلترینگ و کنترل برای سیستم¬های تصادفی دارای پیچیدگی¬های متداول و مهندسی[۵۵]، پوشش روش¬های طراحی الگوریتم¬های کنترل بهینه (یا شبه بهینه) به شکل سنتز سیستم¬های دینامیکی قطعی و تصادفی، با کاربرد در هوافضا، رباتیک و تکنولوژی سرومکانیک [۴۰]، پوشش مهم‌ترین دسته‌های پروسه¬های تصادفی به کار رفته در مدل‌سازی سیستم¬های مختلف [۵۶]، شناسایی پارامتری و شرایط مرزی/ اولیه در معادلات دیفرانسیل جزئی قطعی و تصادفی و بررسی کنترل بهینه معادلات دیفرانسیل تصادفی [۵۷]، مسئله کنترل بهینه برای سیستم¬های قطعی و تصادفی با پارامتر پیوسته یا گسسته [۴۴].

نظریه کنترل رایج و متعارف به بشر این امکان را داده است تا قرن‌ها محیط خود را کنترل و خودکار کند. با گذشت زمان، تکنیک‌های کنترل مدرن به مهندسان این توانایی را داده است که سیستم‌های کنترلی را که از نظر هزینه و کارایی بهینه کنند. با این حال، الگوریتم‌های کنترل بهینه همیشه در برابر تغییرات سیستم کنترل یا محیط کارایی لازم را ندارند. از این رو، شاخه جدیدی تحت عنوان کنترل مقاوم[1] مطرح گردید که یک حوزه بسیار جذاب در طراحی سیستم‌های کنترل است. نظریه کنترل مقاوم راهکاری برای اندازه‌گیری تغییرات عملکرد یک سیستم کنترل نسبت به تغییر پارامترهای سیستم و کنترل آن است.

هدف کنترل مقاوم این است که امکان کاوش در فضای طراحی برای گزینه‌های غیرحساس نسبت به تغییرات در سیستم فراهم شود و بتواند پایداری و عملکرد آن‌ها را حفظ کند. یکی از نتایج مطلوب این روش، مربوط به سیستم‌هایی است که در صورت وجود تغییرات یا خطاهای سیستم، افت عملکرد شدیدی را نشان می‌دهند. تحقیق در زمینه کنترل مقاوم مدرن به وسیله زامس[2] در سال ۱۹۸۱ آغاز گردید [۵۸] و مسائل کنترل بهینه برای کمینه کردن نرم‌ها[3] در فضای هاردی[4] فرمول‌بندی شدند. حل فضای حالت چنین مسائلی به وسیله دویل در سال ۱۹۸۹ ارائه گردید [۹]. از مهم‌ترین مراجع و مقالات چاپ شده در زمینه کنترل مقاوم می‌توان به موارد زیر اشاره نمود: مجموعه مقالات کنترل مقاوم [۵۹]، کنترل مقاوم سیستم¬های غیرخطی با نامعینی پارامتری متغیر با زمان [۶۰]، پایدارسازی و کنترل بهینه سیستم¬های غیرخطی با مدل نامعین و تضعیف اغتشاش [۲۶]، کنترل حالت لغزشی [۶۱]، فیلتر کالمن مقاوم، فیلتر H-inf و فیلتر H-inf برای سیستم¬های گسسته[۳۳]، روش‌های کنترل مقاوم، تئوری کنترل H-inf و کنترل حالت لغزشی سیستم¬های خطی [۳۴]، کنترل تطبیقی مقاوم برای سیستم¬های غیرخطی قابل خطی¬سازی فیدبک چند متغیره دارای غیرخطی¬های نامعین قادر به تضمین عملکرد تعیین شده [۶۲]، ارزیابی و بهینه‌سازی تحمل عدم قطعیت یا حاشیه پایداری [۶۳]، بررسی پایداری مقاوم و عملکرد سیستم¬ها تحت تأثیر نامعینی پارامتری و عدم قطعیت غیر ساختاری [۳۵]، کنترل تطبیقی مقاوم سیستم¬های نامعین تک ورودی در حضور اشباع ورودی و اغتشاشات خارجی نامعین [۶۴]، بررسی اساس مسائل تحلیل و طراحی سیستم-های خطی گسسته و پیوسته بر اساس LMI [۶۵]، مواجهه با کنترل کلاسیک پیش¬بین با استفاده از روش‌های بروز تصادفی و مقاوم [۵۱]، کنترل H2/H-inf تصادفی غیرخطی [۵۲]، بررسی چالش‌های تحلیل پایداری و طراحی کنترل مثل غیرخطی بودن، ابعاد، نامعینی و قیود اطلاعات علاوه بر رفتار ناشی از کوانتیزه ¬سازی، نمونه‌برداری داده و ضربه¬ها [۶۶]، بررسی ریاضیات و کاربردهای کنترل خطی، غیرخطی، بهینه، پیش¬بین مدل، مقاوم، دیجیتال و تطبیقی [۶۷]، موضوعات پیشرفته در کنترل خودکار و کنترل بهینه و مقاوم [۴۱]، بررسی ویژگی مقاوم کنترل فیدبک حالت با توجه به اصل جدایی از رویت‌گر [۶۸].

کنترل تطبیقی[5] دیگر حوزه جذاب در طراحی سیستم‌های کنترل است. هیچ تعریف کاملاً مشخصی که موردقبول همگان باشد برای سیستم‌های کنترل تطبیقی موجود نیست. با توجه به این موضوع که همیشه آرزوی بشر این بوده است که جهت کنترل نمودن و تنظیم بهتر منابع و سیستم‌های موجود از تمامی ابزارهای موجود استفاده نماید تا بتواند به زندگی خود ادامه دهد، کلمه "تطبیقی" مثل کلماتی دیگر از دنیای بیولوژی به دنیای مهندسی آورده شده است. چراکه می‌دانیم دینامیک سیستم‌ها و محیط اطرافشان در حال تغییر می‌باشد و انسان قادر به وفق دادن خود آن‌هم در سطح عالی می‌باشد. (به‌عنوان‌مثال تنظیم قند خون را در نظر بگیرید). به همین خاطر شاید بتوان بدون اغراق انسان را به عنوان کامل‌ترین سیستم دینامیکی، چند متغیره، غیرخطی، متغیر با زمان تطبیقی، با ابعاد بزرگی که از تعداد زیادی سیستم‌های تطبیقی کوچک‌تر تشکیل شده است، در نظر گرفت. البته توجه داریم که شأن انسان خیلی بالاتر از این حرف‌ها است. ایده‌های اولیه کنترل تطبیقی قبل از ۱۹۴۰ به شکل ضعیفی مطرح شدند و در خلال دهه ۱۹۵۰ نخستین بار تنظیم خودکار پارامترهای کنترل‌کننده عملی و نخستین تعریف سیستم‌های کنترل تطبیقی ارائه گردید: توانایی سیستم کنترل در تنظیم و تطبیق پارامترهای خود با دینامیک و متغیرهای حالت سیستم تحت کنترل. شروع کار با روش حساسیت[6] و قانون MIT جهت طراحی قوانین تطبیقی برای روندهای مختلف کنترل‌های تطبیقی به وجود آمد. کالمن در سال ۱۹۵۸ طراحی

1. Robust Control
2. Zames
3. Norm
4. Hardy Space
5 Adaptive Control
6. sensitivity Method

کنترل‌کننده‌های خود ـ بهینه[1] (که همانا روند جایابی تطبیقی قطب[2]، با استفاده از کنترل بهینه خطی است) را ارائه داد. در این زمان طراحی کنترل‌کننده تطبیقی برای اتو پایلوت[3] هواپیماهای با قدرت مانور و سرعت پاسخ بالا به انجام رسید. چرا که به هنگام تغییر نقطه نامی پرواز تغییرات وسیعی در دینامیک سیستم پروازی رخ می‌دهد و کنترل‌کننده‌های کلاسیک غیر تطبیقی قابل طرح نیستند. قطعاً ایده کنترل‌کننده‌های پیچیده‌تر که بتوانند از محیط بیاموزند و دینامیک خود را با تغییرات سیستم تحت بررسی (هواپیما) وفق دهند، ضرورتاً نیاز بود. در نتیجه کنترل‌کننده تطبیقی مدل مرجع توسط آزبورن[4] و همکارانش در ۱۹۶۱ برای حل مسئله کنترل تطبیقی اتو پایلوت ارائه شد. نظر به اینکه در این دوران دانش تئوریک در مورد سیستم‌های تطبیقی به طور کامل تدوین نشده بود و پیاده‌سازی سخت‌افزاری سیستم‌های اتوماتیک کنترل به طور آنالوگ صورت می‌گرفت، این طراحی کنترل‌کننده تطبیقی موفقیت‌آمیز نشد و در عمل جواب‌های تجربی حاصل از پرواز، منفی از آب درآمد [۴].

هرچند تحقیقات در مورد تئوری کنترل تطبیقی اساساً از ۱۹۷۵ شروع شده است، لیکن در دهه ۱۹۶۰ بسیاری از شاخه‌های مهم تئوری کنترل از قبیل آنالیز متغیرهای حالت، آنالیز پایداری و شناسایی سیستم‌ها، توسعه یافتند: تئوری کنترل دوگانه[5] توسط فلد بام[6] در سال ۱۹۶۶، مسئله یادگیری در سیستم‌های کنترلی اتوماتیک در سال ۱۹۷۱ توسط سیپکین[7]، تئوری تخمین و شناسایی پارامترها توسط آستروم[8] در سال ۱۹۷۱. پیشرفت‌های حاصله در تئوری پایداری و تئوری کنترل در همین دهه موجبات فهم بیشتر کنترل تطبیقی در دهه ۷۰ قرن بیستم را فراهم آورد، به‌طوری که در دهه ۷۰ شاهد تحولات شگرف در فن‌آوری و تجهیزات کامپیوتری بوده‌ایم که این امر به نوبه خود موجب سهولت پیاده‌سازی کنترل‌کننده‌های پیشرفته از جمله کنترل‌کننده‌های تطبیقی شد. در این دهه (۱۹۷۰) نشان داده شد که کارهای زیادی در سیستم‌های کنترلی بر اساس تئوری‌های یادگیری و تطبیقی می‌توانند در یک لوای مشترک تحت معادلات تکراری یا بازگشتی[9] قرار گیرند و همچنین توسعه‌های زیادی در شناسایی و تخمین پارامترها انجام شد. در دهه ۱۹۷۰ هنگامی که روش‌های مختلف شناسایی با روش‌های مختلف طراحی ترکیب گردیدند، تحول وسیعی در کنترل تطبیقی صورت پذیرفت. در دهه ۸۰ و در اواخر دهه ۷۰ تئوری پایداری سیستم‌های تطبیقی با ارائه دلیل مطرح شدند. تکنیک‌های مختلف کنترل تطبیقی مدل مرجع با استفاده از روند طراحی مبتنی بر تئوری پایداری لیاپانوف ارائه و مورد تجزیه‌وتحلیل قرار گرفت. آگارت[10] در ۱۹۷۹، لاندو[11] در ۱۹۷۹، نارندرا[12] در ۱۹۸۰، مورس[13] در ۱۹۸۳ و همچنین بسیاری کاربردهای موفقیت‌آمیز کنترل تطبیقی ارائه شد. هریس[14] ۱۹۸۱، نارندرا ۱۹۸۰، اونبهاین[15] ۱۹۸۰. در دهه ۱۹۸۰ تئوری سیستم‌های کنترل تطبیقی دیجیتال مدون گشت و نخستین کنترل‌کننده تطبیقی دیجیتال در سال ۱۹۸۱ وارد بازار شد [۴]. از افراد شاخص این دهه می‌توان از آستروم[16] (۱۹۸۳) و آیزرمن (۱۹۸۶) نام برد. اوایل دهه ۸۰ و اواخر دهه ۷۰ قرن بیستم نشان داده شد که وقتی نویز حتی با مقدار کم روی سیستم عمل کند، روش‌های تطبیقی ارائه‌شده در این دهه می‌توانند به‌راحتی ناپایدار گردند. و این یعنی این‌که کنترل‌های تطبیقی در برابر دینامیک‌های مدل نشده و اغتشاشات مقاوم نیستند، ایونا [17] ۱۹۸۳، روهرز[18] ۱۹۸۵. در اواسط دهه ۸۰ اصلاحات و طراحی‌های جدید پیشنهاد شدند و مبحث جدیدی تحت عنوان کنترل تطبیقی مقاوم اعلام وجود نمود. یک کنترل‌کننده تطبیقی را مقاوم گویند اگر تضمین کند که پاسخ سیستم در حضور طیف وسیعی از دینامیک‌های مدل نشده و اغتشاشات با دامنه محدود، محدود بماند. روند تکامل در تئوری و تکنولوژی سیستم‌های تطبیقی و تطبیقی مقاوم به طور همزمان ادامه یافت که می‌توان از کارهای آستروم در سال ۱۹۸۹، نارندرا در ۱۹۸۶، هریس در سال ۱۹۸۳، آیزرمن در سال ۱۹۹۲ و ایونا در ۱۹۹۱ یاد نمود [۴]. از مهم‌ترین مراجع و مقالات چاپ شده در زمینه کنترل تطبیقی می‌توان به موارد زیر اشاره نمود: مفاهیم کنترل تطبیقی و بررسی روش‌ها و تفاوت کنترل تطبیقی و مقاوم [۴]، طراحی کنترل ردیاب فیدبک خروجی تطبیقی فراگیر برای سیستم¬های تک ورودی-تک خروجی [۶۹]، طراحی کنترل¬کننده فیدبک خروجی تطبیقی شبه فراگیر با تضمین ردیابی هر سیگنال مرجع کراندار مفروض توسط خروجی [۷۰]، تضعیف اغتشاش، کنترل تصادفی و کنترل تطبیقی سیستم‌های غیرخطی [۲۶]، ارائه طرح پایدارسازی تطبیقی (بر اساس توابع تنظیم) که نیازمند دانش قبلی از کران کوواریانس نیست [۴۹]، ارائه روش جدید برای ایجاد طرح¬های کنترل غیرخطی تطبیقی به‌سادگی تنظیم شونده [۷۱]، کنترل‌کننده تطبیقی مقاوم جدید برای سیستم¬های غیرخطی قابل خطی¬سازی فیدبک چند ورودی-چندخروجی دارای غیرخطی‌های نامعین قادر به تضمین عملکرد تعیین شده [۶۲]، معرفی مکانیسم‌های تطبیقی بر اساس تخمین آنلاین عیب برای کنترل/ فیلترینگ قابل اطمینان [۷۲]، کنترل ردیاب تطبیقی سیستم-های غیرخطی چند متغیره نامعین با قیود ورودی غیرمتقارن [۷۳]، کنترل تطبیقی مقاوم سیستم¬های غیرخطی نامعین با استفاده از روش پسگام در حضور اشباع ورودی و اغتشاشات خارجی با استفاده از تابع نوسام [۶۴]،

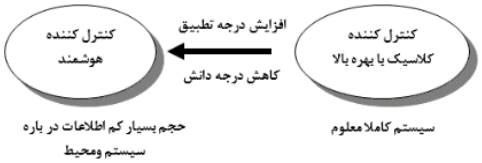
1. Self - optimization
2. Adaptive Pole Positioning
3. Auto - Pilot
4. Osburn
5. Dual control
6. Feldbaum
7. Tsypkin
8. Astrom
9. Recursive
10. Egardt

andou. L11.
Narendra12
13 Morse
Harise14
Onbehauen15
Austrom16-
Ioannou17
Rohrs18

مجموعه مقالات ریاضیات تئوری کنترل برای کنترل پروسه¬های تطبیقی [۷۴]، بررسی استراتژی¬های کنترل تطبیقی مدل مرجع، رگولاتورهای خودتنظیم، سیستم¬های ساختار متغیر، طرح‌های کاهش مرتبه، کنترل پیش¬بین، منطق فازی و...[۷۵]، طراحی کنترل PI/PD/PID تطبیقی بر اساس سیستم‌های غیرخطی [۷۶]، بررسی ریاضیات و کاربردهای کنترل خطی، غیرخطی، بهینه، پیش‌بین مدل، مقاوم، دیجیتال و تطبیقی [۶۷].

آنچه مسلم است بایستی سیستم‌های تطبیقی را در ارتباط با داده‌ها (دانش از سیستم و سیگنال‌های محیطی) و تنظیم (تغییرات در رفتار خود) مورد ملاحظه قرارداد. به شکل ۱ توجه نمایید. یک سیستم تطبیقی مثلاً کنترل‌کننده تطبیقی، بایستی اطلاعاتی در مورد سیستم تحت بررسی را در خلال اجرای سیستم (که در مسیر حلقه بسته خود با سیستم صورت می‌پذیرد) جمع‌آوری نماید و رفتار خود را بتواند مطابق با قاعده یا قانونی تنظیم نماید آن‌گونه که توافقی حاصل آید. واضح است که هرچقدر میزان یا حجم اطلاعات کمتر باشد، میزان تطبیقی بودن و وفق یافتن بیشتر می‌گردد و هرچقدر نحوه تنظیم یا قاعده تطبیقی بهتر باشد، سرعت وفق بیشتر خواهد بود. اگر این دو موضوع را به‌عنوان دو شاخص اصلی بپذیریم، می‌توانیم بگوییم که سیستم‌های کنترلی کلاسیک با گین[۱] یا بهره بالا، منتهی الیه سمت راست (ساده‌ترین شکل) سیستم‌های تطبیقی را نمایندگی می‌کنند؛ چراکه در اینجا همه اطلاعات در مورد سیستم تحت بررسی، محیط اطراف در اختیار می‌باشد و سیستم کنترل‌کننده هم با در اختیار داشتن کل اطلاعات با بهره بالا طراحی می‌شود، طوری که هم پایداری تضمین گردد و هم کل سیستم در برابر تغییرات محدود پارامترها و نقاط کاری، کمتر حساس می‌باشد [۴].

در مقابل، سیستم‌های هوشمند مثلاً سیستم‌های کنترلی مبتنی بر شبکه‌های عصبی و محاسبات فازی، منتهی‌الیه سمت چپ (پیچیده‌ترین و یا به عبارتی کامل‌ترین شکل) سیستم‌های تطبیقی را معرفی خواهند نمود (شکل۲)؛ چراکه سیستم‌های هوشمند، مدل آزاد[۲] بوده و احتیاجی به دانش مدل خاصی از سیستم تحت بررسی ندارند و فقط به داده‌های تجربی که در مسیر زمان به دست می‌آورند، نیاز دارند [۴].




شکل۲. حدود سیستم‌های تطبیقی [۴]

در بخش ۱، کنترل کلاسیک و تاریخچه آن مورد بررسی قرار گرفت. در ادامه و در بخش۳، ابتدا به‌عنوان آشنایی با مقدمات و مفاهیم لازم از شبکه‌های عصبی مصنوعی، نرون‌های مصنوعی و مدل آن‌ها معرفی خواهند شد. سپس شبکه‌های عصبی مصنوعی را معرفی خواهیم کرد که از اتصال چندین نرون مصنوعی تشکیل می‌شود و نحوه آموزش این شبکه‌ها موردبررسی قرار می‌گیرند. در بخش ۳ تا حد امکان سعی شده است که از دیدگاه تئوری کنترل کلاسیک به مسئله و فرمول‌بندی آن نگاه شود.

## ۳- مقدمه‌ای بر شبکه‌های عصبی از دیدگاه تئوری کلاسیک

عبارت هوش مصنوعی[۳] برای نخستین بار توسط جان مک‌کارتی[۴] در سال ۱۹۵۶ مطرح شد. بر اساس تعریف مک‌کارتی، هوش مصنوعی به علم و روش‌های مهندسی هوشمند کردن ماشین‌ها گفته می‌شود. حوزه مطالعاتی هوش مصنوعی گسترده است و موضوعات مختلفی را شامل می‌شود. طیف وسیع پژوهش‌ها و کاربردهای هوش مصنوعی را می‌توان در چندین شاخه از قبیل: شبکه‌های عصبی[۵]، منطق فازی[۶]، یادگیری ماشین[۷]، و ... تقسیم‌بندی کرد که در این مقاله به شاخه "شبکه‌های عصبی" پرداخته شده است.

اصولاً یک شبکه عصبی مصنوعی[۸] یا به‌اختصار شبکه عصبی (NN)، یک ابزار محاسباتی است که در شاخه هوش محاسباتی[۹] قرار می‌گیرد. منشأ الهام این ابزار محاسباتی، مشابه بسیاری از ابزارهای محاسباتی دیگر در شاخه هوش محاسباتی، سیستم‌های بیولوژیکی و طبیعی هستند که دانشمندان این سیستم‌ها را موردمطالعه قرار داده‌اند و برای آن‌ها توصیف یا مدل‌های ریاضی به دست آورده‌اند و این توصیف‌های ریاضی، به یک سری ابزارها یا بلوک‌های محاسباتی تبدیل شده‌اند، از قبیل: شبکه‌های عصبی مصنوعی (الهام گرفته‌شده از شبکه‌های عصبی بیولوژیکی)، الگوریتم ژنتیک[۱۰] یا GA (الهام گرفته‌شده از فرایند تکامل[۱۱] در طبیعت)، الگوریتم ازدحام ذرات[۱۲] یا PSO (الهام گرفته‌شده از رفتار اجتماعی دسته‌های پرندگان) و غیره. در این بخش، به یکی از این ابزارهای محاسباتی مهم، یعنی شبکه‌های عصبی (مصنوعی) پرداخته می‌شود.

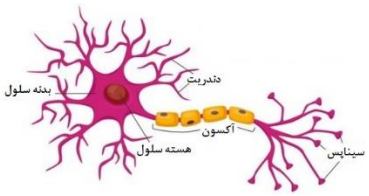
همان‌طور که گفته شد، شبکه عصبی یا مناسب‌تر این است که بگوییم شبکه عصبی (مصنوعی) را می‌توان یک مدل/ابزار محاسباتی در نظر گرفت که عملکرد آن الهام گرفته از شبکه‌های عصبی بیولوژیکی موجود در مغز انسان است که وظیفه پردازش اطلاعات را بر عهده دارند. شبکه‌های عصبی بیولوژیکی، شبکه‌هایی از سلول‌های عصبی (که نرون[۱۳] نام دارند) مغز هستند. مغز انسان، به اذعان بسیاری از دانشمندان، پیچیده‌ترین سیستمی است که تاکنون در کل گیتی مشاهده‌شده و موردمطالعه قرار گرفته است. اما این سیستم پیچیده نه ابعادی در حد کهکشان دارد و نه تعداد اجزای

۱. Model Free
3. Artificial Intelligence
4. John McCarthy
5. Neural Networks (NN)
6. Fuzzy Logic
7. Machine Learning
8. Artificial Neural Network (ANN)
9. Computational Intelligence (CI)
10. Genetic Algorithms (GA)
11. Evolution
12. Particle Swarm Optimization (PSO)
13. Neuron

سازنده‌اش بیشتر از پردازنده‌های ابررایانه‌های امروزی است. پیچیدگی رازآلود این سیستم (مغز) بی‌نظیر، به اتصال‌های فراوان موجود میان اجزای آن (نرون‌ها) بازمی‌گردد (جالب است بدانید که مغز انسان تقریباً ۱۰۰ میلیارد سلول عصبی یا نرون دارد که هر کدام از این سلول‌ها به‌طور متوسط به ۱۰ هزار سلول یا نرون دیگر به‌نوعی متصل شده است). لذا تجسم کنید که چه شبکه پیچیده‌ای از سلول‌ها بر مغز انسان حاکم است. به این دلیل، مغز انسان پیچیده‌ترین سیستمی است که ما در کل هستی شناخته‌ایم. در سال ۱۹۰۹، «سانتیاگو رامون کاخال[1]» کشف کرد که مغز از تعداد زیادی نرون متصل به‌هم تشکیل‌شده که پیام‌های بسیار ساده تحریکی[2] و مهاری[3] را برای یکدیگر ارسال می‌کنند و تهییج[4] آن‌ها با همین پیام‌های ساده به‌روز می‌شود. قبل از ادامه بحث اجازه بدهید که با عنصر سازنده شبکه عصبی یعنی نرون بیشتر آشنا شویم [۵، ۷۹-۷۷].

نرون‌ها ساده‌ترین واحد ساختاری سیستم‌های عصبی هستند. بافت‌هایی که عصب نامیده می‌شوند، اجتماعی از نرون‌ها هستند که اطلاعات و پیام‌ها را از یک قسمت بدن به قسمت دیگر منتقل می‌کنند. این پیام‌ها از نوع ضربه‌های[5] الکتروشیمیایی هستند. میلیون‌ها نرون در بدن انسان وجود دارند، حتی ساده‌ترین کارهای روزمره انسان از قبیل پلک زدن، تنها از طریق همکاری همه‌جانبه این نرون‌ها میسر است. بیشترین تعداد نرون‌ها در مغز و باقی در نخاع و سیستم‌های عصبی جانبی تمرکز یافته‌اند. گرچه همه نرون‌ها کارکرد یکسانی دارند، ولی اندازه و شکل آن‌ها بستگی به محل استقرارشان در سیستم عصبی دارد. با وجود این همه تنوع، بیشتر نرون‌ها از سه بخش اصلی تشکیل شده‌اند. مطابق شکل۳، یک نرون بیولوژیکی از سه بخش اصلی تشکیل شده است [۵]:

۱. بدنه سلول[6]
۲. دندریت[7]
۳. اکسون[8]



شکل۳. قسمت‌های اصلی یک نرون بیولوژیکی

عملکرد یک نرون بیولوژیکی بدین صورت است که بدنه سلول (عصبی) شامل هسته و سایر قسمت‌های حفاظتی بوده و دندریت و اکسون عناصر ارتباطی نرون را تشکیل می‌دهند. دندریت‌ها مناطق دریافت سیگنال‌های الکتریکی هستند که دارای شکل شاخه‌ای پیچیده‌ای می‌باشند. دندریت‌ها، سیگنال‌های الکتریکی را به هسته سلول منتقل می‌کنند. بدنه سلول، انرژی لازم را برای فعالیت نرون فراهم نموده و بر روی سیگنال‌های دریافتی عمل می‌کند که با یک عمل ساده جمع و مقایسه با یک سطح آستانه مدل می‌گردد. اکسون، سیگنال‌های الکتروشیمیایی دریافتی از هسته سلول را به نرون‌های دیگر منتقل می‌کند. محل اتصال اکسون یک سلول به دندریت سلول دیگر سیناپس[9] نام دارد. سیناپس‌ها واحدهای ساختاری کوچکی هستند که ارتباط بین نرون‌ها را برقرار می‌سازند [۵].

بنابراین، به‌طور خلاصه یک نرون بیولوژیکی، اطلاعاتی را از سلول‌ها، بافت‌ها و یا سایر نرون‌های بدن دریافت می‌کند (به‌عنوان ورودی‌های نرون) و یک پردازش روی اطلاعات دریافت‌شده انجام می‌دهد و بعد از پردازش، نتیجه را به نحوی به سایر سلول‌ها، نرون‌ها و یا بافت‌ها ارسال می‌کند (به‌عنوان خروجی نرون). در ادامه، وقتی‌که مدل ریاضی نرون را ارائه کردیم، شرح خواهیم داد که این پردازش‌های جزئی چیست؟

جالب است بدانید که زمان پاسخ یک نرون، به‌عنوان واحد سازنده مغز انسان، تقریباً برابر یک میلی‌ثانیه $(10^{-3}\,\text{sec})$ است. این در حالی است که زمان پاسخ یک ترانزیستور سیلیکونی، به‌عنوان واحد سازنده کامپیوتر، تقریباً یک نانوثانیه $(10^{-9}\,\text{sec})$ است. به عبارتی، یک ترانزیستور سیلیکونی یک میلیون بار زمان پاسخ سریع‌تری نسبت به یک نرون دارد. بنابراین، می‌بینیم که واحدهای سازنده مغز به دلیل فرایندهای شیمیایی و زیستی که دارند، واحدهای کندی هستند. ولی چرا ما (درواقع مغز ما) می‌توانیم مثلاً فردی را که یک ماه پیش دیده‌ایم و ظاهر او را می‌شناسیم، بعد از یک ماه اگر دوباره او را ببینیم، خیلی سریع می‌توانیم اطلاعات را بازیابی کنیم و آن فرد را بشناسیم؛ ولی یک کامپیوتر برای انجام این کار، ساعت‌ها زمان نیاز دارد؟ [۷۷].

همان‌طور که قبلاً گفته شد، قدرت پردازش سریع اطلاعات توسط مغز اولاً به خاطر وجود شبکه بسیار بزرگی از سلول‌هایی با ساختار ساده است. دوماً، در سال ۱۹۸۲، فلدمن[10] و بالارد[11] در مقاله [80] نشان دادند که مغز انسان برای انجام هر عملی از قانونی تحت عنوان "قانون صد گام[12]" استفاده می‌کند. این قانون به شرح زیر است:"هر عملی در مغز انسان، مستقل از اینکه چقدر ساده یا پیچیده باشد، حداکثر از ۱۰۰ سلول عصبی یا نرون استفاده می‌کند". طبق این قانون، برای انجام هر عملیاتی در مغز انسان، اعم از حفظ تعادل، حرف زدن، شناسایی یک چهره، نواختن یک نت و ...، حداکثر ۱۰۰ نرون دخیل می‌شود. لذا، در بدترین حالت که ۱۰۰ نرون در انجام یک عملیات مفروض دخیل باشند و این نرون‌ها هم به‌صورت پشت سرهم پاسخ بدهند، انجام آن عملیات توسط مغز ۱۰۰ میلی‌ثانیه طول می‌کشد $(100\times 1m\,\text{s}=100ms)$. درنتیجه، هر عملیاتی در مغز انسان، در کُندترین حالت، ۱۰۰ میلی‌ثانیه طول می‌کشد. توجه داشته

1. Santiago Ramony Cajal
2. Excitatory
3. Inhibitory
4. Excitation
5. Impulse
6. Cell Body
7. Dendrite
8. Axon
9. Synapse
10. Feldman
11. Ballard
12. 100-step rule

باشید که ۱۰۰ نرون حداکثر تعداد نرون دخیل در هر عملیاتی است و برای هر عملیاتی، تعداد نرون لازم به‌صورت بهینه تعیین می‌گردد و حتی میزان اطلاعاتی که بین نرون‌ها ردوبدل می‌شود نیز کافی است (نه زیاد است و نه کم). مثلاً برای عملیات ساده، تعداد این نرون‌ها قطعاً کمتر از ۱۰۰ است.

با توجه به توضیحات فوق، اگر بتوان شبکه‌های عصبی مصنوعی که واحدهای پردازنده آن‌ها نرون‌های مصنوعی هستند (مدل نرون مصنوعی را در ادامه معرفی خواهیم کرد)، به‌گونه‌ای مثلاً در کامپیوتر طراحی نمود که اولاً شامل حدود ۱۰۰ میلیارد نرون مصنوعی باشد و هر یک از این نرون‌ها به‌طور متوسط به حدود ۱۰هزار نرون دیگر متصل شده باشد و ثانیاً، از کمترین تعداد نرون‌ها برای انجام هر عملیات استفاده شود و کمترین اطلاعات لازم نیز بین نرون‌های دخیل منتقل شود، آنگاه ما به یک شبکه عصبی مصنوعی بهینه‌ای دست‌یافته‌ایم. البته بایستی اذعان کرد که ما هیچ‌وقت نمی‌توانیم ۱۰۰ میلیارد نرونی که در مغز انسان است را با کامپیوتر مدل‌سازی و حتی شبیه‌سازی کنیم و مقیاس کاری ما خیلی پایین‌تر از این تعداد است.

در ادامه، ابتدا به‌عنوان آشنایی با مقدمات و مفاهیم لازم از شبکه‌های عصبی مصنوعی، نرون‌های مصنوعی و مدل آن‌ها معرفی خواهند شد. سپس شبکه‌های عصبی مصنوعی را معرفی خواهیم کرد که از اتصال چندین نرون مصنوعی تشکیل می‌شود و نحوه آموزش این شبکه‌ها موردبررسی قرار می‌گیرند و پس از تشریح و آشنایی با این مفاهیم، به نحوه مدل‌سازی و کنترل با استفاده از شبکه‌های عصبی خواهیم پرداخت.

## ۱-۳ مدل یک نرون مصنوعی

اولین گام در معرفی شبکه‌های عصبی مصنوعی، تعریف عنصر سازنده آن یعنی نرون است. در بخش قبل، توصیفی اجمالی از نرون‌های واقعی (بیولوژیکی) و شبکه‌های عصبی ارائه شد. در این بخش قصد داریم که یک مدل ساده و درعین‌حال بسیار کاربردی از نرون واقعی ارائه کنیم و سپس نشان دهیم که چگونه می‌توان ساختارهای مختلف شبکه‌های عصبی (مصنوعی) را از ترکیب و کنار هم قرار دادن تعداد زیادی از این نرون‌ها ایجاد کنیم.

همان‌طور که در بخش قبل گفته شد، به‌طور خلاصه یک نرون که واحد سازنده دستگاه عصبی و مغز انسان است، یک بلوک ساده پردازنده است که اطلاعاتی را از ورودی‌های خود دریافت می‌کند و سپس بر روی این اطلاعات دریافتی پردازش جزئی انجام می‌دهد و نتیجه را به سایر نرون‌ها تحویل می‌دهد.

در سال ۱۹۴۳، آقای مک‌کلاچ[1] (عصب‌شناس) با همکاری آقای پیتز[2] (منطق‌دان) با ایده‌برداری از ساختار و عملکرد کلی نرون بیولوژیکی یک مدل ریاضی برای نرون معرفی کردند که به مدل نرون "مک‌کلاچ – پیتز[3]" معروف است. این مدل ریاضی در حالت تک ورودی به صورت گرافیکی در شکل۴ نشان داده‌شده است و به‌صورت معادله ریاضی زیر بیان می‌شود[۵]:

$$a = f\,(wp + b) \qquad (۱)$$

البته در برخی مراجع، به این مدل، مدل نرون جمعی نیز می‌گویند. در این مدل

$p$ : ورودی نرون

$w$: وزن[4]

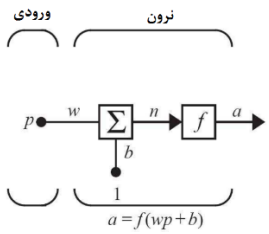
$b$: بایاس[5] یا آفست[6]

$f$: تابع فعال‌ساز[7] یا تابع تبدیل[8]

$n$: ورودی خالص[9]

$a$: خروجی نرون (اسکالر)

هستند.




شکل۴. مدل مک‌کلاچ-پیتز نرون در حالت تک ورودیج

تابع فعال‌ساز $f$ یک تابع خطی یا غیرخطی از $n$ است و برای تعیین خصوصیات نرون در راستای حل مسائل مختلف استفاده می‌شود.

بدنه هر سلول عصبی (نرون) از دو بخش تشکیل می‌شود: بخش اول را تابع ترکیب (جمع‌کننده) می‌گویند. وظیفه تابع ترکیب این است که تمام ورودی‌ها را ترکیب و یک عدد تولید می‌کند. در بخش دوم سلول، تابع انتقال قرار دارد که به آن تابع تحریک نیز می‌گویند. درواقع همان‌گونه که یک سلول بیولوژیک باید به سطح آستانه تحریک خاصی برسد تا یک سیگنال تولید کند، تابع تحریک نیز تا زمانی که ورودی‌های ترکیب‌شده و وزن‌دار شده به یک حد آستانه خاصی نرسند، مقدار خروجی بسیار کوچکی تولید می‌کنند. وقتی ورودی‌های ترکیب‌شده به حد آستانه خاصی برسند، سلول عصبی تحریک‌شده و سیگنال خروجی تولید می‌کند. طراحی یک نرون مصنوعی با مدل مک‌کلاچ-پیتز، درواقع انتخاب یک تابع فعال‌ساز $f$ مناسب و تعیین مقادیر پارامترهای w و b است. به فرایند تعیین پارامترهای w و b نرون‌ها در یک شبکه عصبی، یادگیری[10] شبکه عصبی می‌گویند [۵].

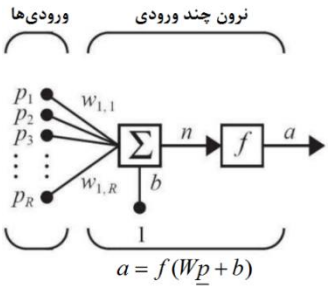
1. Warren MuCulloch
2. Walter Pitts
3. McCulloch-Pitts
4. Weight
5. Bias
6. Offset
7. Activation Function
8. Transfer Function
9. Net Input
10. Learning

در مدل نرون تک ورودی مک‌کلاچ-پیتز ارائه‌شده در معادله (۱)، ورودی خارجی p وارد نرون می‌گردد و با ضرب در مشخصه ذاتی نرون، یعنی وزن w، وزن‌دار می‌شود. ورودی دیگر نرون که مقدار آن ثابت و برابر یک است، در مشخصه ذاتی دیگر نرون، یعنی بایاس b، ضرب شده و حاصل با wp جمع می‌شود. این حاصل جمع، ورودی خالص n برای تابع فعال‌ساز (یا تابع تبدیل) $f$ خواهد بود. تابع فعال‌ساز $f$ بر روی ورودی خالص n اثر کرده و خروجی a نرون را که یک عدد اسکالر است، تولید می‌کند. لذا می‌توان نتیجه گرفت که اگر تابع $f$ خطی باشد $(f(n)=n)$، یک نرون با معادله خط $a=f(wp+b)\cong y=mx+b$ هم‌ارز است.

در مقایسه مدل تک ورودی مک‌کلاچ-پیتز با یک نرون بیولوژیکی (واقعی)، w معادل شدت سیناپس، مجموعه جمع‌کننده و تابع فعال‌ساز معادل هسته سلول و a معادل سیگنال گذرنده از اکسون خواهد بود.

نکته‌ای که باید به آن توجه شود، اهمیت و تأثیر جمله بایاس b است. این جمله را می‌توان مانند وزن w در نظر گرفت، با این تصور که میزان تأثیر ورودی ثابت یک را روی نرون منعکس می‌سازد. وجود جمله b باعث افزودن یک متغیر اضافی به شبکه (عصبی) می‌شود. شبکه‌ای که ترم بایاس دارد، قوی‌تر از شبکه فاقد بایاس عمل می‌کند. همچنین، یک نرون بدون بایاس همیشه در اثر ورودی صفر، خروجی صفر می‌دهد و این حالت مطلوب نیست و با اضافه کردن بایاس به مدل، این نقیصه برطرف می‌شود.

باید توجه داشت که پارامترهای $w$ و $b$ قابل تنظیم هستند و تابع فعال‌ساز $f$ نیز توسط طراح انتخاب می‌شود. به عبارتی، بر اساس انتخاب $f$ و نوع الگوریتم یادگیری، پارامترهای $w$ و $b$ تنظیم می‌شوند. یادگیری بدین معنی است که $w$ و $b$ طوری تغییر می‌کنند که رابطه ورودی و خروجی نرون با هدف خاصی مطابقت نماید. همچنین، باید اشاره نمود که یک نرون لزوماً یک خروجی دارد، اما برای تعداد ورودی‌های آن محدودیتی وجود ندارد. شکل۵ یک نرون با چند ورودی را نشان می‌دهد.



شکل۵. مدل یک نرون چند ورودی

برای یک نرون چند ورودی، رابطه بین ورودی و خروجی طبق رابطه زیر تعریف می‌شود:

$$a=f(w_{1,1}p_1+w_{1,2}p_2+\ldots+w_{1,R}p_R+b) \qquad (۲)$$

برای سهولت نمایش، اگر بردار ورودی به‌صورت $\underline{p}=[p_1\cdots p_R]^T$ تعریف شود و وزن‌های نرون با ماتریس W زیر خلاصه شوند:

$$W=[w_{1,1}\ w_{1,2}\ \cdots\ w_{1,R}]$$

آنگاه خروجی نرون چند ورودی را می‌توان به فرم زیر نیز توصیف نمود:

$$\begin{aligned} a&=f(w_{1,1}p_1+w_{1,2}p_2+\ldots+w_{1,R}p_R+b)\\ &=f(W\underline{p}+b) \end{aligned} \qquad (۳)$$

نمایش ماتریسی فوق یک نمایش استاندارد در توصیف شبکه‌های عصبی است. در انتخاب اندیس‌ها، قرارداد خاصی به‌کاررفته است که باید درباره آن توضیح داده شود. دومین اندیس در نمایش ماتریسی شبکه‌های عصبی یا نرون‌ها، مبدأ سیگنال ورودی نرون را نشان می‌دهد و اندیس اول به شماره خود نرون اشاره خواهد داشت. برای نرون با ساختار شکل۵، چون فقط یک نرون داریم، ماتریس وزن W به یک بردار سطری تبدیل می‌شود. مثلاً عنصر $w_{1,2}$ بیانگر وزن دومین عنصر ورودی به نرون اول است. در ادامه که شبکه‌های عصبی را معرفی خواهیم کرد، خواهیم دید که در حالت کلی، هر سطر ماتریس W متناظر با یک نرون است.

به‌عنوان یک نکته در نظر داشته باشید که تعداد ورودی‌ها (R) از صورت‌مسئله موردبررسی مشخص می‌شود. به‌عبارت‌دیگر، R تحت انتخاب طراح نیست، بلکه بستگی به روش حل مسئله موردبررسی دارد. به‌عنوان‌مثال، اگر بخواهیم یک شبکه عصبی طراحی کنیم که تابع دو ورودی $y=x_1+x_2^2$ را تقریب بزند، R باید برابر با ۲ باشد.

## ۲-۳ تابع فعال‌ساز نرون

تابع فعال‌ساز نرون می‌تواند در حالت کلی یک تابع خطی یا غیرخطی از ورودی خالص شبکه یعنی $n$ باشد که بسته به نوع کاربرد، نحوه عمل آن متفاوت است. توجه داشته باشید که اگر از توابع فعال‌ساز در مدل نرون و شبکه‌های عصبی استفاده نکنیم، وزن‌ها و مقدار بایاس فقط یک معادله خطی را ایجاد می‌کنند. درست است که معادله خطی خیلی راحت‌تر حل‌شدنی است، اما برای حل مسائل پیچیده از قبیل مدل‌سازی سیستم‌های غیرخطی نمی‌تواند کمکی به ما کند؛ درواقع معادلات خطی در یادگیری الگوهای پیچیده داده خیلی محدود هستند و یک شبکه عصبی بدون تابع فعال‌ساز فقط یک مدل رگرسیون خطی است. به‌طورکلی، شبکه‌های عصبی از توابع فعال‌ساز استفاده می‌کنند تا بتوانند به شبکه در یادگیری داده‌های پیچیده کمک و پیش‌بینی قابل‌قبولی را در خروجی ارائه کنند [۵].

تابع فعال‌ساز در شبکه عصبی بعد از بلوک جمع در نرون مصنوعی (نرون جمعی) قرار می‌گیرد. یعنی پس از جمع ورودی‌ها، نتیجه از یک تابع فعال‌ساز عبور می‌کند. اما این تابع فعال‌ساز چه تابعی هست؟ برای نرون‌ها، معمولاً از تعداد محدودی تابع فعال‌ساز در محاسبات استفاده می‌شود که در جدول۱ لیست شده است [۵].

جدول ۱. توابع فعال‌ساز پرکاربرد در شبکه‌های عصبی

| نام تابع فعال‌ساز | رابطه ریاضی $a=f(n)$ | شکل |
|---|---|---|
| محدودیت سخت | $a=\begin{cases}0, n<0\\1, n\geq 0\end{cases}$ | |
| محدودیت سخت متقارن[1] | $a=\begin{cases}-1, n<0\\1, n\geq 0\end{cases}$ | |
| خطی | $a=n$ | |
| خطی اشباع‌شده[2] | $a=\begin{cases}0, n<0\\n, 0\leq n<1\\1, n\geq 1\end{cases}$ | |
| خطی اشباع شده متقارن | $a=\begin{cases}-1, n<-1\\n, -1\leq n<1\\1, n\geq 1\end{cases}$ | |
| حلقوی لگاریتمی | $a=\frac{1}{1+e^{-n}}$ | |
| حلقوی تانژانت هیپربولیک | $a=\frac{e^{n}-e^{-n}}{e^{n}+e^{-n}}$ | |
| خطی مثبت[3] | $a=\begin{cases}0, n<0\\n, n\geq 0\end{cases}$ | |

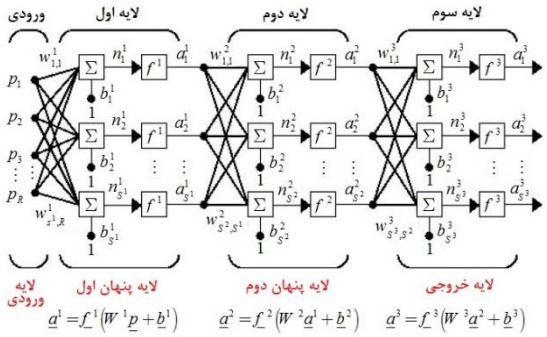
با انتخاب تابع فعال‌ساز، عملاً نوع مدلِ نرون نیز مشخص می‌گردد. مثلاً به یک نرون با تابع فعال‌ساز محدودیت سخت[4]، نرون پرسپترون[5] گفته می‌شود. بنابراین، خروجی یک نرون پرسپترون همواره صفر یا یک است. از نرون پرسپترون عمدتاً برای طبقه‌بندی کردن داده‌ها (ورودی‌ها) به دو کلاس محدود استفاده می‌شود. نرون‌های خطی (نرون‌هایی که تابع فعال‌ساز آن‌ها از نوع خطی هستند) برای تخمین توابع و سیستم‌های خطی و حتی سیستم‌های غیرخطی حول نقطه کار استفاده می‌شوند. همچنین، در شبکه‌های عصبی چند لایه، استفاده از این تابع فعال‌ساز در لایه آخر باعث می‌شود که لایه آخر یک تابع خطی از خروجی نرون‌های لایه قبل باشد.

از پراستفاده‌ترین توابع فعال‌ساز غیرخطی می‌توان به تابع حلقوی لگاریتمی[6] و حلقوی تانژانت هیپربولیک[7] اشاره نمود. تابع فعال‌ساز نوع حلقوی لگاریتمی، هر عدد حقیقی را به‌عنوان ورودی دریافت می‌کند و خروجی آن عددی بین صفر و یک است. هرچقدر مقدار ورودی بزرگ‌تر (مثبت‌تر) باشد، خروجی این تابع به یک نزدیک‌تر است و هرچقدر مقدار ورودی کوچک‌تر باشد، خروجی بیشتر به صفر نزدیک‌تر خواهد بود.

تابع حلقوی تانژانت هیپربولیک ازلحاظ شکل مختصاتی بسیار شبیه به تابع حلقوی لگاریتمی است؛ با این تفاوت که خروجی تابع (بُرد تابع) در بازه ۱- تا ۱+ است. در تابع حلقوی تانژانت هیپربولیک، هرچقدر ورودی بزرگ‌تر (مثبت‌تر) باشد، خروجی آن بیشتر به ۱ نزدیک می‌شود و هرچقدر ورودی کوچک‌تر (منفی‌تر) باشد، خروجی تابع به ۱- نزدیک خواهد بود.

## ۳-۳ شبکه‌های عصبی پیشرو

چنانچه چندلایه نرون کنار یکدیگر قرار بگیرند، نرون‌های چندلایه یا همان شبکه‌های عصبی[8] را تشکیل می‌دهند. هر چیدمانی از لایه‌ها در کنار یکدیگر، یک نوع شبکه عصبی با کاربرد خاص ارائه می‌دهد. مثلاً چنانچه چندین لایه به‌گونه‌ای در کنار یکدیگر چیده شوند که ورودی ابتدا وارد لایه اول گردد و خروجی لایه اول به‌عنوان ورودی لایه دوم محسوب شود و دوباره ورودی لایه سوم، همان خروجی لایه دوم باشد و این روند ادامه پیدا کند، شبکه عصبی حاصل را یک شبکه عصبی چندلایه پیشرو[9] می‌گویند [۵]. در شکل ۶، نمونه‌ای از یک شبکه عصبی سه لایهِ پیشرو نشان داده‌شده است. مطابق شکل ۶، در این نوع از شبکه‌های عصبی، هر لایه با شماره آن لایه تعیین می‌گردد؛ امّا در منابع علمی، به لایه‌ای از این شبکه که خروجی از آن حاصل می‌شود، لایه خروجی[10] گفته می‌شود و به ورودی‌ها که در تعریف لایه نمی‌گنجند، لایه ورودی[11] گفته‌شده و به لایه‌های دیگر، لایه‌های پنهان[12] می‌گویند. توجه داشته باشید که بیشتر شبکه‌های عصبی عملی به دلیل کثرت محاسبات، دو تا حداکثر سه لایه دارند. شبکه‌هایی با بیش از چهار لایه به‌جز در مباحث پردازش تصویر به‌ندرت یافت می‌گردند.




شکل ۶. یک شبکه عصبی سه لایه پیشرو

شبکه عصبی نشان داده‌شده در شکل ۶ دارای تعداد $R$ ورودی، $S^1$ نرون در لایه اول، $S^2$ نرون در لایه دوم و $S^3$ نرون در لایه سوم است (اندیس

1. Symmetrical Hard Limit
2. Saturating Linear
3. Positive Linear
4. Hard Limit
5. Perceptron
6. Log-Sigmoid
7. Hyperbolic Tangent Sigmoid
8. Neural Networks
9. FeedForward Multi-Layer
10. Output Layer
11. Input Layer
12. Hidden Layer

بالانویس بیانگر شماره لایه است). همان‌طور که مشخص است، یک شبکه عصبی می‌تواند دارای تعداد متفاوت نرون در لایه‌های مختلف باشد.

توجه شود که خروجی هر یک از لایه‌های میانی (پنهان) به‌عنوان ورودی لایه بعدی مورداستفاده قرار می‌گیرند. بنابراین، لایه دوم از این شبکه را به‌تنهایی می‌توان یک شبکه تک‌لایه با $S^1$ ورودی (تعداد نرون‌های لایه اول) و $S^2$ خروجی (تعداد نرون‌های لایه دوم) در نظر گرفت که دارای ماتریس وزن $W^2$ با ابعاد $S^2 \times S^1$ است. ورودی لایه دوم بردار $\underline{a}^1$ (بردار خروجی لایه اول) و بردار خروجی آن $\underline{a}^2$ است. این پارامترها را می‌توان به طریق مشابه برای لایه سوم نیز تعریف نمود.

با استفاده از رابطه (۱)، معادلات حاکم بر شبکه عصبی شکل۶ در لایه اول با $S^1$ نرون و $R$ ورودی عبارت است از:

$$\begin{cases} a_1^1 = f^1(w_{1,1}^1 p_1 + w_{1,2}^1 p_2 + \ldots + w_{1,R}^1 p_R + b_1^1) \\ a_2^1 = f^1(w_{2,1}^1 p_1 + w_{2,2}^1 p_2 + \ldots + w_{2,R}^1 p_R + b_2^1) \\ \quad \vdots \\ a_{S^1}^1 = f^1(w_{S^1,1}^1 p_1 + w_{S^1,2}^1 p_2 + \ldots + w_{S^1,R}^1 p_R + b_{S^1}^1) \end{cases} \quad (۴)$$

و یا

$$\underline{a}^1 = f^1(W^1 \underline{p} + \underline{b}^1) \quad (۵)$$

به‌طوری‌که ماتریس وزن $W^1$ و بردار بایاس $\underline{b}^1$ لایه اول برابر هستند با:

$$W^1 = \begin{bmatrix} w_{1,1}^1 & w_{1,2}^1 & \cdots & w_{1,R}^1 \\ w_{2,1}^1 & w_{2,2}^1 & \cdots & w_{2,R}^1 \\ \vdots & \vdots & \ddots & \vdots \\ w_{S^1,1}^1 & w_{S^1,2}^1 & \cdots & w_{S^1,R}^1 \end{bmatrix}, \quad \underline{b}^1 = \begin{bmatrix} b_1^1 \\ b_2^1 \\ \vdots \\ b_{S^1}^1 \end{bmatrix}$$

به‌طور مشابه، برای لایه دوم با $S^2$ نرون و $S^1$ ورودی نیز می‌توان نوشت:

$$\begin{cases} a_1^2 = f^2(w_{1,1}^2 a_1^1 + w_{1,2}^2 a_2^1 + \ldots + w_{1,S^1}^2 a_{S^1}^1 + b_1^2) \\ a_2^2 = f^2(w_{2,1}^2 a_1^1 + w_{2,2}^2 a_2^1 + \ldots + w_{2,S^1}^2 a_{S^1}^1 + b_2^2) \\ \quad \vdots \\ a_{S^2}^2 = f^2(w_{S^2,1}^2 a_1^1 + w_{S^2,2}^2 a_2^1 + \ldots + w_{S^2,S^1}^2 a_{S^1}^1 + b_{S^2}^2) \end{cases} \quad (۶)$$

و یا

$$\underline{a}^2 = f^2(W^2 \underline{a}^1 + \underline{b}^2) \quad (۷)$$

به‌طوری‌که داریم:

$$\begin{aligned} \underline{a}^3 &= f^3(W^3 \underline{a}^2 + \underline{b}^3) \\ &= f^3(W^3(f^2(W^2 \underline{a}^1 + \underline{b}^2)) + \underline{b}^3) \\ &= f^3(W^3(f^2(W^2(f^1(W^1 \underline{p} + \underline{b}^1)) + \underline{b}^2)) + \underline{b}^3) \\ &= f_{NN}(\underline{p}, \underline{\theta}) \end{aligned} \quad (۸)$$

همان‌طور که در رابطه (۸) مشاهده می‌شود، یک شبکه عصبی در حالت کلی یک تابع خطی یا غیرخطی (بسته به نوع توابع فعال‌ساز تعریف شده) با پارامترهای مجهول (وزن‌ها و بایاس‌ها) است. به بیان دقیق‌تر، یک شبکه عصبی، یک نگاشتی را با اجزای ساده (نرون‌ها) انجام می‌دهد. پارامترهای مجهول این شبکه عصبی، یعنی وزن‌ها و بایاس‌ها، بایستی به‌گونه‌ای تعیین شوند که خروجی‌های شبکه عصبی، مقادیر مطلوبی را تولید کنند. مثلاً در بحث مدل‌سازی آزمایشی یک سیستم با شبکه عصبی، وزن‌ها و بایاس‌های شبکه باید به‌گونه‌ای تعیین شوند که خروجی شبکه عصبی تا حد ممکن به خروجی واقعی/اندازه‌گیری شده سیستم نزدیک باشد. در ادامه خواهیم دید که مسئله طراحی شبکه عصبی و تعیین وزن‌ها و بایاس‌ها می‌تواند به یک مسئله بهینه‌سازی تبدیل شود که نشان‌دهنده ارتباط بین دنیای هوشمند و دنیای کلاسیک است.

شبکه‌های عصبی چندلایه بسیار قدرتمند هستند و هر یک از لایه‌های آن وظایف متفاوتی دارند. به‌عنوان‌مثال، یک شبکه دولایه‌ای را در نظر بگیرید که لایه اول آن از نرون‌های حلقوی تانژانت هیپربولیک یا حلقوی لگاریتمی و لایه دوم آن از نرون‌های خطی تشکیل‌شده باشند. این شبکه عصبی می‌تواند هر تابع دلخواهی را با تعداد محدود نقاط ناپیوستگی تقریب بزند [۵].

با زیاد شدن پارامترهای طراحی (تعداد لایه‌های شبکه، تعداد نرون‌های هر لایه، نوع توابع فعال‌ساز هر لایه، مقادیر وزن‌ها و بایاس‌ها)، شاید طراحی یک شبکه عصبی کار بسیار دشواری به نظر برسد؛ اما این مشکل چندان هم غیرقابل‌حل نیست. زیرا اولاً باید به یاد داشته باشید که برای هدف مدل‌سازی آزمایشی یک سیستم، تعداد لایه‌های شبکه عموماً دو است و تعداد ورودی‌ها و خروجی‌های آن نیز با توجه به خصوصیات مسئله تعیین می‌شوند. بنابراین، اگر مسئله به‌عنوان‌مثال دارای ۴ ورودی باشد، آنگاه تعداد ورودی‌های شبکه عصبی نیز برابر ۴ خواهد بود. به همین ترتیب، اگر تعداد خروجی‌های مسئله مثلاً ۷ باشد، شبکه عصبی در لایه آخر خود بایستی دارای ۷ نرون باشد. درنهایت می‌توان گفت خصوصیات مطلوب سیگنال خروجی، تعیین‌کننده نوع تابع فعال‌ساز لایه خروجی می‌باشد. بنابراین، طراحی یک شبکه تک لایه وابسته به پارامترهای مسئله است. اما آیا این مطلب در مورد شبکه‌های عصبی چندلایه نیز صادق است؟ باید بگوییم خیر. چراکه نمی‌توان با توجه به پارامترهای مسئله، تعداد نرون‌ها را تعیین نمود. این مسئله برای شبکه‌هایی با بیش از دولایه یک مشکل بزرگ و موضوع بسیاری از تحقیقات علمی می‌باشد؛ ولی در بخش‌های آتی، در بحث مدل‌سازی سیستم‌ها با شبکه‌های عصبی، حداکثر و حداقل تعداد نرون‌های مجاز برای لایه‌های یک شبکه عصبی دولایه تعیین خواهد شد.

در حالت کلی، در مورد تعداد لایه‌های یک شبکه عصبی می‌توان گفت: اغلب شبکه‌های عصبی مورداستفاده شامل ۲ یا حداکثر ۳ لایه هستند و به‌ندرت از شبکه‌هایی با ۴ و یا بیشتر لایه در بحث مدل‌سازی آزمایشی (شناسایی) و کنترل سیستم‌ها استفاده می‌شود. از طرفی، استفاده از بایاس در شبکه دلخواه است. اما باید گفت که استفاده از آن با فراهم آوردن یک پارامتر بیشتر، منجر به قوی‌تر شدن شبکه عصبی می‌شود. به‌طورمعمول، از تابع فعال‌ساز مشابهی برای تمامی لایه‌های پنهان شبکه عصبی استفاده می‌شود. بااین‌حال، بر اساس هدف و کاربرد شبکه عصبی، تابع فعال‌ساز لایه خروجی شبکه معمولاً با سایر توابع فعال‌ساز لایه‌های پنهان مغایرت دارد. عموماً برای مدل‌سازی آزمایشی و کنترل با شبکه عصبی، تابع

فعال‌ساز لایه(های) پنهان از نوع حلقوی لگاریتمی یا حلقوی تانژانت هیپربولیک بوده و برای لایه خروجی از تابع فعال‌ساز خطی استفاده می‌شود. به عنوان یک قاعده کلی، تعداد نرون‌های لایه‌های پنهان یک شبکه عصبی با یکدیگر برابر هستند. برای تشخیص تعداد نرون‌های مناسب (نه بهینه) در لایه پنهان می‌توان از فرمول زیر استفاده نمود [۵، ۸۱] :

$$\text{تعداد نرون‌های لایه‌های پنهان} = \frac{N}{\lambda(R+M)}$$

که $N$ تعداد داده‌ها، $R$ تعداد ورودی شبکه (برابر تعداد ورودی مسئله)، $M$ تعداد خروجی شبکه (برابر تعداد خروجی مسئله) و $\lambda$ ضریبی است که مقدار آن در فرمول فوق، عددی بین ۲ و ۱۰ می‌باشد و برای جلوگیری از بیش‌برازش[1] شدن شبکه از آن استفاده می‌شود.

## ٤- کاربرد شبکه‌های عصبی در مدل‌سازی آزمایشی سیستم‌ها

ورود به دنیای هر علمی بدون آشنایی با مفاهیم پایه و ادبیات مورد گفتگو آن امکان‌پذیر نیست؛ به‌خصوص که آن موضوع در مورد شبکه‌های عصبی باشد. شاید یکی از دلایل وجود این‌همه اصطلاحات در این حوزه، کاربران زیاد و حیطه‌های کاربردی متنوع برای شبکه‌های عصبی از قبیل: مدل‌سازی، پیش‌بینی سری‌های زمانی، طبقه‌بندی، خوشه‌بندی، تشخیص الگو، شناسایی چهره، پردازش تصویر و... باشد. در جدول۲ عبارت‌های معادل شبکه‌های عصبی با اصطلاحات کلاسیک آن‌ها آورده شده است.

جدول۲. عبارت‌های معادل شبکه‌های عصبی با اصطلاحات کلاسیک

| اصطلاح شبکه‌های عصبی | اصطلاح کلاسیک |
|---|---|
| شبکه | مدل (تابع) |
| یادگیری (آموزش) | تخمین |
| یادگیری با نظارت[2] | رگرسیون |
| تعمیم[3] | درون‌یابی[4] |
| مجموعه داده‌های آموزشی[5] | مشاهدات[6] |
| وزن‌ها و بایاس‌ها | پارامترها |
| ورودی‌ها | متغیرهای مستقل |
| خروجی‌ها | متغیرهای وابسته |

یک شبکه عصبی چندلایه با $\underline{p}$ بردار ورودی و $a$ خروجی (اسکالر) در نظر بگیرید. مطابق رابطه (۸)، در حالت کلی این شبکه را می‌توان مطابق زیر توصیف نمود:

$$a = f_{NN}(\underline{p} \mid \underline{\theta}) \qquad (۹)$$

به‌طوری‌که $f_{NN}$ در حالت کلی یک تابع غیرخطی بوده و ترکیبی از توابع فعال‌ساز تعریف‌شده برای نرون‌های هر لایه می‌باشد. در بحث مدل‌سازی آزمایشی یک سیستم مفروض با شبکه عصبی، معادله (۹) همان نقش ساختار مدل را ایفا می‌کند و وزن‌ها و بایاس‌ها نیز به‌عنوان پارامترهای مجهول $(\underline{\theta})$ ساختار مدل هستند که بایستی به‌گونه‌ای تعیین شوند که خروجی شبکه (مدل) بسیار نزدیک به خروجی واقعی سیستم باشد (در حالت ایده‌آل برابر باشند).

یکی از متداول‌ترین ساختار مدل‌ها در شناسایی سیستم‌های دینامیکی غیرخطی و فشرده *SISO*، ساختار *NARX* ازیر است:

$$y[k] = f(y[k-1],...,y[k-n],u[k],...,u[k-m]) + e[k] \qquad (۱۰)$$

به‌طوری‌که $u$ ورودی و $y$ خروجی سیستم بوده و $f$ یک تابع غیرخطی و نامعلوم است که رابطه ریاضی بین خروجی فعلی با ورودی‌های گذشته (و فعلی) و همچنین خروجی‌های گذشته را توصیف می‌کند. توجه داریم که اگر $f$ فقط تابعی از $u[k]$ باشد $(y[k] = f(u[k]))$، رابطه (۱۰) توصیف‌کننده یک سیستم استاتیکی خواهد بود که می‌توان آن را برای مدل‌سازی سیگنال‌های یقینی نیز تعمیم داد. در همه روش‌های مدل‌سازی آزمایشی سیستم‌های غیرخطی، هدف اصلی در حقیقت تخمین/تقریب زدن تابع $f$ در ساختار *NARX* ارائه‌شده در رابطه (۱۰) است، به‌گونه‌ای که خطای این تخمین کمینه باشد. جهت تقریب‌زدن تابع $f$ ساختار *NARX* با استفاده از یک شبکه عصبی پرواضح است که ورودی‌های شبکه عصبی بایستی همان ورودی‌های این تابع باشد. بنابراین، تعریف می‌کنیم:

$$\underline{p}[k] = [p_1[k] \cdots p_n[k] \; p_{n+1}[k] \cdots p_{n+m+1}[k]]^T \triangleq [y[k-1] \cdots y[k-n] \; u[k] \cdots u[k-m]]^T$$

توجه داشته باشید که با تعریف بردار ورودی شبکه مطابق فوق، مجموعه داده‌های آموزشی شبکه نیز برابر خواهند بود با:

$$\{(u[k], y[k])\}_{k=1}^{N} \Rightarrow \{(\underline{p}[k], y[k])\}_{k=1}^{N}$$

در این صورت، پارامترهای شبکه عصبی $a[k] = f_{NN}(\underline{p}[k] \mid \underline{\theta})$ بایستی به‌گونه‌ای تعیین شوند که در هر گام زمانی $k = 1,2,...,N$، با اعمال مقادیر ورودی اندازه‌گیری شده از سیستم مفروض به شبکه عصبی، مقادیر خروجی شبکه عصبی به مقادیر خروجی اندازه‌گیری شده از سیستم بسیار نزدیک باشد. به‌عبارت‌دیگر، اگر خطای خروجی شبکه (مدل) در هر گام زمانی را به‌صورت زیر تعریف کنیم:

$$\varepsilon[k] \triangleq y[k] - a[k] = y[k] - f_{NN}(\underline{p}[k] \mid \underline{\theta}) \;, \quad \forall k \in \{1,2,...,N\} \qquad (۱۱)$$

1. Overfit
2. Supervised Learning
3. Generalization
4. Interpolation
5. Training Set
6. Observations

بایستی به طریقی سعی کنیم که به ازای هر $k \in \{1,2,...,N\}$، $\varepsilon[k]$ کمترین مقدار را داشته باشد. تنها پارامترهایی که با تنظیم آن‌ها می‌توان $\varepsilon[k]$را کمینه کرد، وزن‌ها و بایاس‌های شبکه ($\underline{\theta}$) هستند. بنابراین، پارامترهای شبکه عصبی ($\underline{\theta}$) باید به‌گونه‌ای تعیین شوند که به ازای هر $k \in \{1,2,...,N\}$، $\varepsilon[k]$ کمینه شود. بدین منظور، تابع هزینه زیر را که به‌صورت مجموع مربعات خطا تعریف شده است، در نظر می‌گیریم:

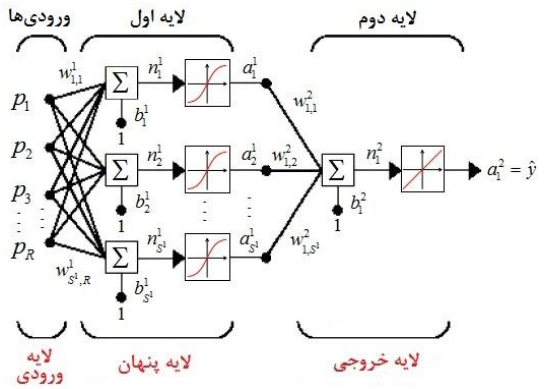
$$J = \sum_{k=1}^{N} \varepsilon^2[k] = \sum_{k=1}^{N} (y[k] - f_{NN}(\underline{p}[k] | \underline{\theta}))^2 \qquad (۱۲)$$

همان‌طور که ملاحظه می‌کنید، در این تابع هزینه به ازای هر $k \in \{1,2,...,N\}$، مقدار خروجی اندازه‌گیری شده $y[k]$ از مقدار خروجی شبکه یعنی $a[k]$ کم شده و سپس به توان ۲ رسانده شده است (یعنی به ازای هر مقدار ورودی شبکه $\underline{p}[k]$ با $k = 1,2,...,N$، یک مقدار خروجی اندازه‌گیری شده $(k = 1,2,...,N \; ; \; y[k])$، یک مقدار خروجی شبکه عصبی $(k = 1,2,...,N \; ; \; a[k])$ و درنتیجه یک مقدار برای خطای مدل $(k = 1,2,...,N \; ; \; \varepsilon[k])$ خواهیم داشت). سپس مقدار خطای مدل در همه نمونه‌ها باهم جمع شده و خطای شبکه محاسبه‌شده است. به‌این‌ترتیب، ما درکی نسبی از عملکرد شبکه خواهیم داشت، به این شکل که هرچقدر خطا کمتر باشد، احتمالاً عملکرد شبکه بهتر است و هرچقدر که خطا بیشتر باشد، شبکه عملکرد خوبی نخواهد داشت. درنتیجه، مسئله آموزش یا یادگیری[1] شبکه عصبی با استفاده از مجموعه داده‌های آموزشی (داده‌های اندازه‌گیری شده از سیستم) را می‌توان معادل حل مسئله بهینه‌سازی زیر در نظر گرفت:

$$\min_{\underline{\theta}} J = \min_{\underline{\theta}} \sum_{k=1}^{N} (y[k] - f_{NN}(\underline{p}[k] | \underline{\theta}))^2 \qquad (۱۳)$$

می‌دانیم بسته به اینکه تابع $f_{NN}$ خطی یا غیرخطی باشد، مسئله بهینه‌سازی فوق به‌صورت تحلیلی[2] یا عددی[3] قابل‌حل است و مقادیر بهینه پارامترهای مجهول شبکه (وزن‌ها و بایاس‌ها) به دست می‌آیند. قضیه زیر بیانگر این حقیقت است که برای مدل‌سازی آزمایشی سیستم‌های غیرخطی با شبکه‌های عصبی، $f_{NN}$ یک تابع غیرخطی خواهد بود. بنابراین، برای حل مسئله بهینه‌سازی (۱۳)، بایستی از روش‌های بهینه‌سازی عددی سود جست.

**قضیه ۱)** شبکه دولایه پیشرو شکل۷ را در نظر بگیرید که لایه اول آن از نرون‌های حلقوی تانژانت هیپربولیک (یا حلقوی لگاریتمی) و لایه دوم آن از یک نرون خطی تشکیل‌شده باشند. با انتخاب مناسب تعداد نرون‌های لایه پنهان این شبکه و تعیین مقادیر بهینه وزن‌ها و بایاس‌ها، این شبکه عصبی می‌تواند هر تابع پیوسته دلخواه $y = f(\underline{p})$ با $f : \mathbb{R}^R \to \mathbb{R}$ را تقریب بزند [۵، ۷۸].




شکل۷. یک شبکه عصبی پیشرو با دولایه: لایه اول و دوم به ترتیب متشکل از نرون‌های حلقوی تانژانت هیپربولیکی و خطی

## ۴-۱ تعیین تعداد نرون‌های لایه پنهان یک شبکه عصبی دو لایه

شبکه عصبی پیشرو دولایه با تعداد $R$ ورودی و $M$ خروجی را در نظر بگیرید که شامل $S^1$ نرون در لایه اول (پنهان) و $S^2$ نرون در لایه دوم (خروجی) است. پرواضح است که باید $S^2 = M$ باشد. اما چگونه می‌توان تعداد نرون‌های لایه پنهان، مقدار $S^1$، را تعیین نمود؟

روش‌های متعددی برای تعیین مناسب (نه بهینه) تعداد نرون‌های لایه پنهان یک شبکه عصبی دولایه پیشنهادشده است. مثلاً در [۵، ۸۱] ذکرشده است که تعداد نرون‌های لایه پنهان را می‌توان تحت یکی از روش‌های زیر تعیین نمود:

۱. تعداد نرون‌های لایه پنهان $(S^1)$ بین تعداد ورودی $(R)$ و تعداد خروجی $(M)$ باشد.

۲. $S^1 = \frac{2}{3}(R+M)$

۳. $S^1 < 2R$

۴. $S^1 = \frac{N}{\lambda(R+M)}$، که $N$ تعداد نمونه‌های مجموعه داده آموزشی بوده و $\lambda$ ضریب ثابتی در بازه $\lambda \in [2,10]$ است.

۵. $S^1 = \frac{1}{2}(R+M)$

توجه داشته باشید که اگر برای لایه پنهان، تعداد نرون کمتری انتخاب شود، منجر به کم‌برازش[4] می‌شود. درحالی‌که اگر تعداد زیادی نرون را انتخاب کنیم، ممکن است منجر به بیش‌برازش[5]، واریانس بالا و افزایش زمان لازم برای آموزش شبکه شود.

## ۴-۲ بهینه‌سازی عددی و کاربرد آن در آموزش شبکه‌های عصبی

منظور از بهینه‌سازی عددی، تکرار یک الگوریتم ساده برای رسیدن به نقطه بهینه در تابع هزینه است. برخلاف الگوریتم‌های تحلیلی که به‌صورت مستقیم و کلی از روی ریشه‌های مشتق یا گرادیان، نقطه بهینه را

1. Learning
2. Analytical
3. Numerical
4. Underfitting
5. Overfitting

تعیین می‌کنند، در الگوریتم‌های بهینه‌سازی عددی، نقطه بهینه به‌صورت تکراری[1] و در طی چند مرحله محاسبه می‌شود. الگوریتم‌های بهینه‌سازی عددی، امروزه اهمیت زیادی در علوم مختلف پیدا کرده‌اند و در بهینه‌سازی سیستم‌ها به‌وفور استفاده می‌شوند. یک مثال رایج از این کاربردها، شبکه‌های عصبی است. در بخش ۳ دیدیم که در شبکه‌های عصبی، تعداد زیادی پارامتر وجود دارند که باید به صورتی انتخاب شوند که شبکه عصبی همانند سیستم دلخواهی که از قبل مشخص‌شده است، رفتار کند. به همین دلیل، تابع هزینه به صورت مجموع مربعات خطا (بین خروجی سیستم واقعی و شبکه عصبی) تعریف می‌شود و از روی کمینه‌سازی/حداقل‌سازی این تابع هزینه، بهترین پارامترهای ممکن برای شبکه عصبی انتخاب می‌شوند. حل تحلیلی این مسئله بهینه‌سازی با استفاده از گرادیان، به دلیل پیچیدگی بسیار زیاد سیستم، معمولاً زمان‌بر بوده و گاهی اصلاً امکان‌پذیر نیست. درنتیجه روش جایگزین، استفاده از الگوریتم‌های بهینه‌سازی عددی است.

امروزه در همه شبکه‌های عصبی از الگوریتم‌های بهینه‌سازی عددی برای بهینه‌سازی پارامترهای آن‌ها (که در اصطلاح به آن آموزش شبکه عصبی نیز گفته می‌شود) استفاده می‌کنند.

الگوریتم‌های گسترده‌ای برای بهینه‌سازی عددی وجود دارند که از مهم‌ترین آن‌ها می‌توان به الگوریتم سریع‌ترین نزول[2]، الگوریتم نیوتن[3]، الگوریتم مومنتوم[4]، الگوریتم ژنتیک[5] و ... اشاره کرد [۸۳-۸۲]. هرکدام از این الگوریتم‌ها برای دسته خاصی از مسائل بهینه‌سازی بهترین عملکرد را دارند و ما باید با توجه به نوع مسئله بهینه‌سازی که می‌خواهیم آن را حل کنیم، بهترین الگوریتم بهینه‌ساز را انتخاب کنیم. در ادامه به تشریح برخی از این روش‌های بهینه‌سازی خواهیم پرداخت.

الگوریتم‌ها یا روش‌های گرادیانی یکی از روش‌های کلاسیکی هستند که در مسائل بهینه‌سازی محلی[6] مورداستفاده قرار می‌گیرند. اصول کار این روش‌ها بیشتر بر پایه مفاهیمی چون گرادیان[7] و هسین[8] بناشده است. الگوریتم با دریافت یک حدس اولیه، سعی می‌کند به مقادیر بهینه محلی همگرا شود.

تابع چندمتغیره $J:\mathbb{R}^n \to \mathbb{R}$ زیر را در نظر بگیرید:

$$J(\underline{z}) = J(z_1, z_2, ..., z_n) \qquad (۱۴)$$

گرادیان این تابع نسبت به بردار $\underline{z} \in \mathbb{R}^n$ برابر است با:

$$\nabla J(\underline{z}) = \begin{bmatrix} \dfrac{\partial J(\underline{z})}{\partial z_1} \\ \vdots \\ \dfrac{\partial J(\underline{z})}{\partial z_n} \end{bmatrix} \qquad (۱۵)$$

چنانچه گرادیان تابع $J$ در یک نقطه برابر با صفر شود $(\nabla J(\underline{z}^*) = \underline{0})$، $\underline{z}^*$ را نقطه ایستا[9] یا نقطه بحرانی (کمینه‌کننده یا بیشینه‌کننده یا نقطه زینی[10]) می‌گویند که شرط لازم برای بهینه‌سازی تابع $J$ است. به‌عبارت‌دیگر، اگر $\underline{z}^*$ یک کمینه‌کننده محلی برای تابع $J$ باشد، آنگاه $\nabla J(\underline{z}^*) = \underline{0}$ است.

از طرفی، ماتریس هسین تابع $J$ نیز به صورت زیر تعریف می‌گردد:

$$H \triangleq \nabla^2 J(\underline{z}) = \begin{bmatrix} \dfrac{\partial^2 J}{\partial z_1^2} & \dfrac{\partial^2 J}{\partial z_1 \partial z_2} & \cdots & \dfrac{\partial^2 J}{\partial z_1 \partial z_n} \\ \dfrac{\partial^2 J}{\partial z_2 \partial z_1} & \dfrac{\partial^2 J}{\partial z_2^2} & \cdots & \dfrac{\partial^2 J}{\partial z_2 \partial z_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 J}{\partial z_n \partial z_1} & \dfrac{\partial^2 J}{\partial z_n \partial z_2} & \cdots & \dfrac{\partial^2 J}{\partial z_n^2} \end{bmatrix} \qquad (۱۶)$$

شرط کافی به‌منظور بهینه‌سازی تابع $J$ (تعیین نوع نقطه بحرانی: کمینه‌کننده، بیشینه‌کننده یا نقطه زینی)، با استفاده از ماتریس هسین آن مشخص می‌گردد. اگر ماتریس هسین تابع $J$ در نقطه $\underline{z}^*$ یک ماتریس مثبت معین باشد، آنگاه تابع $J$ دارای یک نقطه کمینه محلی در $\underline{z}^*$ است. همچنین، با منفی معین بودن ماتریس هسین در نقطه $\underline{z}^*$، تابع $J$ دارای یک نقطه بیشینه محلی در $\underline{z}^*$ خواهد بود. اگر ماتریس هسین، هم دارای مقادیر ویژه با علامت‌های مثبت و هم منفی در نقطه $\underline{z}^*$ باشد، آنگاه این نقطه یک نقطه زینی برای تابع $J$ در نظر گرفته می‌شود. در غیر این صورت، ماتریس هسین کارایی نخواهد داشت. به همین منظور، به قضیه زیر توجه کنید که اثبات آن را می‌توانید در مرجع [۸۲] دنبال کنید.

**قضیه۲)** فرض کنید که تابع $J:\mathbb{R}^n \to \mathbb{R}$ دو بار مشتق‌پذیر باشد. اگر $\underline{z}^*$ کمینه‌کننده محلی باشد و در رابطه $\nabla J(\underline{z}^*) = \underline{0}$ صدق کند، آنگاه ماتریس هسین تابع $J$ در نقطه $\underline{z}^*$ یک ماتریس نیمه مثبت معین است $(\nabla^2 J(\underline{z}^*) \geq 0)$.

با استفاده از قضیه۲، در ادامه دو الگوریتم عددی سریع‌ترین نزول و نیوتن جهت بهینه‌سازی محلی یک تابع تشریح می‌شود.

### ۱-۲-۴ الگوریتم سریع‌ترین نزول

سریع‌ترین نزول یک الگوریتم بهینه‌سازی تکرارپذیر است که به کمک آن می‌توان مقدار کمینه یک تابع موردنظر را محاسبه کرد. در اینجا تابع موردنظر، همان تابع هزینه است.

الگوریتم سریع‌ترین نزول را با یک مثال ساده تشریح می‌کنیم. تصور کنید که در کوهستان مشغول پیاده‌روی هستید و می‌خواهید از ستیغ کوه به دره برسید، ولی به علت مه‌آلود بودن هوا متوجه نمی‌شوید که چه موقعی

1. Iterative
2. Steepest Descent
3. Newton
4. Momentum
5. Genetic
6. Local Optimization
7. gradient
8. Hessian
9. Stationary Point
10. Saddle Point

به دره رسیده‌اید. مطابق شکل۸، از شیب زمین و سرازیری کمک می‌گیرد. از سرازیری به پایین می‌آیید. در ابتدای راه گام‌های شما بلند است، زیرا می‌دانید که هدف (دره) دور از شما قرار دارد و احتیاج به‌دقت در تشخیص موقعیت خود ندارید. ولی زمانی که شیب ملایم می‌شود، می‌فهمید که به دره نزدیک شده‌اید و باید گام‌هایتان را کوتاه بردارید تا دره را پیدا کنید. به‌این‌ترتیب در هر گام، موقعیت بعدی‌تان را بر اساس موقعیت فعلی تنظیم می‌کنید تا از مسیر خارج نشوید. به این شیوه مشخص است که گام بعدی شما به گام قبلی بستگی دارد. در این حالت، کاهش طول گام‌ها را با کاهش شیب هماهنگ می‌کنید تا زمانی که دیگر شیبی وجود ندارد. در این هنگام، به دره که هدفتان بود، رسیده‌اید. این عملکرد همان چیزی است که در الگوریتم‌های بهینه‌سازی عددی مبتنی بر گرادیان از قبیل سریع‌ترین نزول و نیوتن رخ می‌دهد [۷۷].

در الگوریتم سریع‌ترین نزول، نقطه کمینه کننده تابع $J(\underline{z})$ با حل معادله بازگشتی زیر به دست می‌آید:

$$\underline{z}_{k+1} = \underline{z}_k - \alpha_k \underline{g}_k \qquad (۱۷)$$


شکل۸ تشریح روش سریع‌ترین نزول با یک مثال ساده [۷۷]

که $\underline{g}_k$ برابر با مقدار گرادیان تابع $J(\underline{z})$ در نقطه $\underline{z} = \underline{z}_k$ است (که همان نقش مقدار شیب سرازیری در مثال پیاده‌روی در کوهستان را دارد). به‌عبارت‌دیگر:

$$\underline{g}_k = \nabla J(\underline{z})|_{\underline{z}=\underline{z}_k} \qquad (۱۸)$$

در رابطه (۱۷)، $\alpha_k$ به نرخ یادگیری[1] معروف است و معمولاً بر اساس تجربه تعیین می‌گردد.

### ۴-۲-۲ الگوریتم نیوتن

در الگوریتم بهینه‌سازی عددی نیوتن، نقطه کمینه‌کننده تابع $J(\underline{z})$ با حل معادله بازگشتی زیر به دست می‌آید:

$$\underline{z}_{k+1} = \underline{z}_k - (H_k)^{-1}\underline{g}_k \quad ; \quad H_k = \nabla^2 J(\underline{z})|_{\underline{z}=\underline{z}_k} \qquad (۱۹)$$

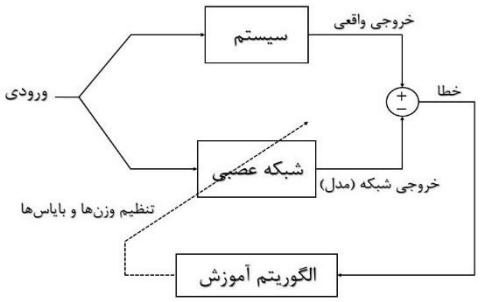
که مشابه الگوریتم سریع‌ترین صعود $\underline{g}_k = \nabla J(\underline{z})|_{\underline{z}=\underline{z}_k}$ است. توجه داریم که در روش نیوتن، به‌جای نرخ یادگیری $(\alpha_k)$ در الگوریتم سریع‌ترین نزول، از معکوس ماتریس هسین تابع $J(\underline{z})$ استفاده می‌شود.

برای مطالعه بیشتر و آشنایی با سایر الگوریتم‌های بهینه‌سازی عددی از قبیل الگوریتم مومنتوم، ناحیه اطمینان و ...، مرجع [۷۷] را مطالعه کنید.

## ۴-۳ آموزش شبکه‌های عصبی با روش‌های بهینه‌سازی عددی کلاسیک

همان‌گونه که در شروع بخش ۴ گفته شد، ما در آموزش شبکه عصبی به الگوریتمی نیاز داریم که وزن‌ها و بایاس‌ها را آن‌قدر تغییر بدهد (اصلاح کند) تا تابع هزینه تعریف‌شده در (۱۳) کمینه گردد. توجه داشته باشید که لزومی ندارد با یک بار تغییر وزن‌ها و بایاس‌ها به جواب برسیم. در مسائل پیچیده ممکن است فرایند بهینه‌سازی هزاران بار تکرار شود و درنهایت تازه به یک خطای قابل‌قبول برسد نه خطای صفر!.

به بیان ساده، الگوریتم‌های آموزش/یادگیری شبکه عصبی روش‌های بهینه‌سازی عددی هستند که جهت محاسبه وزن‌ها و بایاس‌های شبکه عصبی استفاده می‌شوند. مطابق شکل۹، در یادگیری‌های با نظارت، مجموعه داده‌های آموزشی (داده‌های ورودی و خروجی اندازه‌گیری شده از سیستم) دریافت می‌گردد. سپس خروجی شبکه به‌ازای داده‌های ورودی آموزشی محاسبه می‌شود و با تفاضل آن از داده‌های خروجی آموزشی، خطای شبکه محاسبه می‌گردد. بر اساس این خطا، الگوریتم آموزش شبکه شروع به سعی و خطا در جهت کاستن این خطا (به بیان دقیق‌تر کاستن مقدار تابع هزینه تعریف‌شده بر اساس خطا) می‌نماید تا به مقادیر وزن‌ها و بایاس‌هایی که به جواب مسئله نزدیک است، همگرا شود.




شکل۹. روش تنظیم وزن‌ها و بایاس‌های شبکه عصبی توسط یک الگوریتم آموزش

کاربرد اصلی و اولیه هر شبکه عصبی، آموزش یا فراگیری تعدادی نقطه از پیش موجود (مجموعه داده‌های آموزشی) با تنظیم وزن‌ها و بایاس‌ها است. پس از آموزش، شبکه عصبی می‌تواند برای محاسبه خروجی هر ورودی دلخواه استفاده شود (بحث تعمیم[2] در شبکه‌های عصبی). یعنی اینکه شبکه عصبی آموزش داده‌شده می‌تواند مانند یک تابع عمل کند و به ازای هر ورودی دلخواه، خروجی سیستم را پیش‌بینی کند.

1. Learning Rate
2. Generalization

در ادامه این بخش، با کمک روش بهینه‌سازی سریع‌ترین نزول، چگونگی محاسبه وزن‌ها و بایاس‌های شبکه عصبی (آموزش شبکه عصبی) تشریح می‌شود.

توجه داشته باشید که با دیگر روش‌های بهینه‌سازی کلاسیک مطرح شده در بخش قبلی نیز می‌توانید فرمول‌های مشابهی را برای محاسبه وزن‌ها و بایاس‌ها استخراج کنید.

### ۱-۳-۴ الگوریتم آموزشی گرادیان نزولی

الگوریتم آموزشی گرادیان نزولی[1] اساسی‌ترین و پرکاربردترین الگوریتم بهینه‌سازی به‌حساب می‌آید که به‌طور گسترده در آموزش شبکه‌های عصبی استفاده می‌شود. گرادیان نزولی به الگوریتم بهینه‌سازی مرتبه اول گفته می‌شود که به مشتق مرتبه اول تابع هزینه بستگی دارد.

فرض کنید که مجموعه داده‌هایی که برای آموزش شبکه عصبی موردنیاز است، به‌صورت زیر داده شده باشند:

$$\{(\underline{p}[k],y[k])\}_{k=1}^{N}=\{(\underline{p}[1],y[1]),...,(\underline{p}[N],y[N])\}$$

در مرحله آموزش یک شبکه عصبی، وزن‌ها و بایاس‌ها (پارامترهای شبکه عصبی) باید به‌گونه‌ای محاسبه شوند که برای هر $k\in\{1,2,...,N\}$، تابع هزینه تعریف شده در (۱۳) کمینه گردد؛ یعنی

$$\min_{\underline{\theta}} J=\min_{\underline{\theta}}\sum_{k=1}^{N}(y[k]-f_{NN}(\underline{p}[k]|\underline{\theta}))^2$$

که بردار $\underline{\theta}$ شامل ضرایب وزن و بایاس شبکه عصبی است. درصورتی‌که تعریف کنیم:

$$\underline{y}=[y[1]\ y[2]\ \cdots\ y[N]]^T$$
$$\underline{a}=[a[1]\ a[2]\ \cdots\ a[N]]^T$$
$$\underline{\varepsilon}=\underline{y}[k]-\underline{a}[k]=[\varepsilon[1]\ \varepsilon[2]\ \cdots\ \varepsilon[N]]^T$$

آنگاه تابع هزینه $J$ در رابطه (۱۳) را می‌توان به فرم زیر بازنویسی نمود:

$$J=\sum_{k=1}^{N}\varepsilon^2[k]=\underline{\varepsilon}^T\underline{\varepsilon} \qquad (۲۰)$$

بر طبق روش بهینه‌سازی سریع‌ترین نزول، چنانچه در هر تکرار (گام)، وزن شبکه عصبی بر طبق رابطه بازگشتی زیر محاسبه شود، تابع هزینه (۲۰) کمینه می‌گردد:

$$W_{k+1}=W_k-\alpha\underline{g}_k \qquad (۲۱)$$

بخش پیچیده این رابطه، $\underline{g}_k$ است. $\underline{g}_k$ در واقع همان گرادیان تابع $J$ نسبت به متغیر وزن است. بنابراین می‌توان نوشت:

$$\underline{g}_k=\frac{\partial J}{\partial W_k} \qquad (۲۲)$$

بر طبق قانون زنجیره‌ای، رابطه (۲۲) را می‌توان بسط داد (بدون از دست رفتن کلیت مسئله، جهت سهولت، اندیس $k$ را در روابط بعدی نمی‌نویسیم):

$$\underline{g}=\frac{\partial J}{\partial W}=\frac{\partial J}{\partial \underline{n}}\times\frac{\partial \underline{n}}{\partial W} \qquad (۲۳)$$

اگر برای هدف مدل‌سازی آزمایشی سیستم، شبکه عصبی را شامل دولایه فرض کنیم (با استناد به قضیه ۱ که هر تابع پیوسته‌ای را می‌توان با یک شبکه عصبی دولایه با توابع فعال‌ساز حلقوی لگاریتمی یا حلقوی تانژانت هیپربولیک برای لایه اول و خطی برای لایه دوم تقریب زد) و ورودی لایه اول $\underline{p}=\underline{a}^0$ و ورودی لایه دوم همان خروجی لایه اول، یعنی $\underline{a}^1$، باشد، آنگاه خواهیم داشت:

$$\underline{n}^1=W^1\underline{a}^0+\underline{b}^1\quad,\quad \underline{n}^2=W^2\underline{a}^1+\underline{b}^2$$

اگر $m$ بیانگر شماره لایه باشد $(m=1,2)$، در این صورت می‌توان نوشت:

$$\begin{cases}\dfrac{\partial \underline{n}^1}{\partial W^1}=\underline{a}^0\\ \dfrac{\partial \underline{n}^2}{\partial W^2}=\underline{a}^1\end{cases}\Rightarrow\quad \frac{\partial \underline{n}^m}{\partial W^m}=\underline{a}^{m-1} \qquad (۲۴)$$

حال اگر برای هر لایه تعریف کنیم:

$$\underline{s}^m\triangleq\frac{\partial J}{\partial \underline{n}^m} \qquad (۲۵)$$

آنگاه رابطه بازگشتی جهت محاسبه وزن‌ها برابر خواهد شد با:

$$W_{k+1}^m=W_k^m-\alpha\underline{s}^m(\underline{a}^{m-1})^T \qquad (۲۶)$$

اما مقدار $\underline{s}^m$ چگونه محاسبه می‌شود؟ $\underline{s}^m$ نمادی از حساسیت لایه $m$ام شبکه است و با استفاده از مفهوم پس‌انتشار[2] به دست می‌آید. چراکه رابطه $\underline{s}^m$ یک رابطه بازگشتی است. به‌عبارت‌دیگر، $\underline{s}^2$ متناظر با لایه دوم به $\underline{s}^1$ لایه اول وابسته بوده و $\underline{s}^1$ نیز به لایه اول یا همان $\underline{s}^0$ وابسته است. جهت واضح شدن این مفهوم، دوباره از قانون زنجیره‌ای برای محاسبه $\underline{s}^m$ استفاده می‌کنیم:

$$\underline{s}^m=\frac{\partial J}{\partial \underline{n}^m}=\underbrace{\left(\frac{\partial \underline{n}^{m+1}}{\partial \underline{n}^m}\right)^T}_{(I)}\times\underbrace{\frac{\partial J}{\partial \underline{n}^{m+1}}}_{(II)} \qquad (۲۷)$$

حتماً توجه دارید که تابع هزینه خطای تعریف‌شده در (۲۰)، برابر تفاضل مقدار واقعی و خروجی لایه آخر است. ازاین‌رو، مشتق تابع هزینه نسبت به لایه آخر در رابطه (۲۷) ظاهرشده است.

برای ترم اول $(I)$ در رابطه (۲۷) می‌توان نوشت:

$$\underline{n}^{m+1}=W^{m+1}\times F(\underline{n}^m)+\underline{b}^{m+1}$$
$$\Rightarrow\left(\frac{\partial \underline{n}^{m+1}}{\partial \underline{n}^m}\right)^T=\dot{F}^m(\underline{n}^m)\times(W^{m+1})^T \qquad (۲۸)$$

که در این رابطه، $\dot{F}^m(\underline{n}^m)$ مشتق تابع فعال‌ساز لایه $m$ است. از طرفی، برای ترم دوم $(II)$ در رابطه (۲۷) نیز طبق تعریف داریم:

1. Gradient Descent
2. Backpropogation

$$\frac{\partial J}{\partial \underline{n}^{m+1}} = \underline{s}^{m+1} \qquad (۲۹)$$

بنابراین، با جایگذاری روابط (۲۸) و (۲۹) در رابطه (۲۷) خواهیم داشت:

$$\underline{s}^{m} = \dot{F}^{m}(\underline{n}^{m}) \times (W^{m+1})^{T} \times \underline{s}^{m+1} \qquad (۳۰)$$

با توجه به رابطه فوق، پرواضح است که محاسبات لایه $m$ به محاسبات لایه $m+1$ وابسته است. طبق تعریف ارائه شده در رابطه (۲۵)، برای $\underline{S}^{m+1}$ می‌توان نوشت:

$$\begin{aligned} \underline{s}^{m+1} &= \frac{\partial J}{\partial \underline{n}^{m+1}} = \frac{\partial (\underline{\varepsilon}^{T}\underline{\varepsilon})}{\partial \underline{n}^{m+1}} = -2\frac{\partial \underline{a}}{\partial \underline{n}^{m+1}} \times (\underline{y} - \underline{a}) \\ &= -2\dot{F}^{m+1}(\underline{n}^{m+1}) \times (\underline{y} - \underline{a}) \end{aligned} \qquad (۳۱)$$

که در این رابطه، $\dot{F}^{m+1}(\underline{n}^{m+1})$ مشتق تابع فعال‌ساز لایه $m+1$ است. این روند محاسبات را می‌توان برای تعیین رابطه بازگشتی بایاس شبکه عصبی نیز انجام داد.

به طور خلاصه و به‌عنوان جمع‌بندی، برای آموزش یک شبکه عصبی با $M$ لایه به روش گرادیان نزولی، از روابط بازگشتی زیر جهت محاسبه وزن‌ها و بایاس‌های هر لایه استفاده می‌شود:

$$\begin{cases} W_{k+1}^{m} = W_{k}^{m} - \alpha \underline{s}^{m} (\underline{a}^{m-1})^{T} \\ \underline{b}_{k+1}^{m} = \underline{b}_{k}^{m} - \alpha \underline{s}^{m} \end{cases} \qquad (۳۲)$$
$$m = M, M-1, \ldots, 2, 1$$

در این روابط $\underline{a}^{m-1}$ خروجی لایه $m$-1 شبکه و $\alpha$ نرخ یادگیری است. $\underline{s}$ نمادی از حساسیت شبکه است که برای لایه آخر $M$ و سایر لایه‌ها به‌صورت زیر تعریف می‌شود:

$$\begin{cases} \underline{s}^{M} = -2\dot{F}^{M}(\underline{n}^{M}) \times (\underline{y} - \underline{a}) \\ \underline{s}^{m} = \dot{F}^{m}(\underline{n}^{m}) \times (W^{m+1})^{T} \times \underline{s}^{m+1} \end{cases} \qquad (۳۳)$$
$$m = M-1, M-2, \ldots, 2, 1$$

که $\underline{a}$ خروجی شبکه و $\underline{y}$ خروجی واقعی است. همچنین، $\dot{F}$ نمادی از مشتق تابع فعال‌ساز هر لایه است که به‌صورت یک ماتریس قطری به فرم زیر تعریف می‌شود:

$$\dot{F}^{m}(\underline{n}^{m}) = \begin{bmatrix} \dot{f}^{m}(\underline{n}_{1}^{m}) & 0 & \cdots & 0 \\ 0 & \dot{f}^{m}(\underline{n}_{2}^{m}) & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \dot{f}^{m}(\underline{n}_{S^{m}}^{m}) \end{bmatrix} \qquad (۳۴)$$

توجه داشته باشید که روابط (۳۲) تا (۳۴) را می‌توان برای آموزش شبکه‌های عصبی با هر تعداد لایه استفاده نمود. توجه داشته باشید که حتی بدون استخراج روابط (۳۲) تا (۳۴) و به طور مستقیم با بهره‌گیری از روش‌های بهینه‌سازی که در بخش۴-۲ معرفی شد نیز می‌توان به آموزش شبکه‌های عصبی پرداخت.

### ۴-۳-۲ الگوریتم آموزشی گرادیان نزولی با مومنتوم[1]

همانطور که در بخش قبل گفته شد، الگوریتم آموزشی گرادیان نزولی اصلی‌ترین روش برای آموزش شبکه‌های عصبی است، اما برای کاربردهای عملی کُند است. لذا می‌توان از الگوریتم گرادیان نزولی با مومنتوم استفاده نمود.

در الگوریتم آموزشی گرادیان نزولی با مومنتوم، وزن‌ها و بایاس‌های شبکه عصبی طبق روابط زیر محاسبه می‌شوند:

$$\begin{aligned} &W_{k+1}^{m} = \gamma W_{k}^{m} - (1-\gamma)\alpha \underline{s}^{m} (\underline{a}^{m-1})^{T} \\ &m = M, M-1, \ldots, 2, 1 \end{aligned} \qquad (۳۵)$$

$$\begin{aligned} &\underline{b}_{k+1}^{m} = \gamma \underline{b}_{k}^{m} - (1-\gamma)\alpha \underline{s}^{m} \quad ; \\ &m = M, M-1, \ldots, 2, 1 \end{aligned} \qquad (۳۶)$$

پارامتر $\gamma$ مومنتوم نام دارد و مقدار آن بین صفر تا یک است. این پارامتر نمادی از حافظه و جهت تعیین سهم محاسبات قدیم وزن و بایاس در به‌روزرسانی وزن و بایاس جدید است.

### ۴-۳-۳ الگوریتم آموزشی نیوتن

تاکنون در همه الگوریتم‌های آموزشی ارائه‌شده قبلی فرض شده بود که باید تابع خطای زیر که اختلاف بین خروجی مطلوب و خروجی شبکه عصبی است، در هر گذر حداقل شود:

$$L = (\underline{\varepsilon}[k])^{T}\,\underline{\varepsilon}[k] = (\underline{y}[k] - \underline{a}[k])^{T}(\underline{y}[k] - \underline{a}[k]) \qquad (۳۷)$$

همان‌گونه که در بخش بهینه‌سازی عددی دیدید (بخش۴-۲)، مقدار کمینه این تابع از الگوریتم بازگشتی سریع‌ترین نزول قابل‌محاسبه بود. بر طبق این روش، باید وزن‌ها را به‌صورت زیر به‌روزرسانی نمود تا به حداقل تابع خطا رسید:

$$W_{k+1} = W_{k} - \alpha \underline{g}_{k} \qquad (۳۸)$$

نتیجه این الگوریتم بازگشتی پس از محاسبه گرادیان تابع خطا $(\underline{g}_{k})$، الگوریتم آموزشی گرادیان نزولی خواهد بود که در بخش گذشته معرفی شد. اما متأسفانه این الگوریتم علی‌رغم همگرایی مناسب، کُند است. بنابراین، می‌توان از روش بهینه‌سازی عددی نیوتن استفاده نمود و وزن‌ها را به‌صورت زیر به دست آورد:

$$W_{k+1} = W_{k} - (H_{k})^{-1}\underline{g}_{k} \qquad (۳۹)$$

نتیجه این الگوریتم بازگشتی پس از محاسبه گرادیان تابع خطا $(\underline{g}_{k})$ و معکوس ماتریس هسین $(H_{k})^{-1}$، الگوریتم آموزشی نیوتن است. رابطه بازگشتی برای تعیین بایاس‌ها نیز مشابه رابطه (۳۹) است.

### ۴-۳-۴ الگوریتم آموزشی لونبرگ- مارکوارت[2]

متأسفانه الگوریتم آموزشی نیوتن نیاز به معکوس ماتریس هسین $(H_{k})^{-1}$ دارد که محاسبه آن معمولاً پیچیده و دشوار است. لذا روش

1. Momentum
2. Levenberg–Marquardt

آموزش لونبرگ-مارکوارت به کار گفته می‌شود [۵]. اگر بدون اثبات قبول کنیم که

$$\underline{g}_k = 2\left(\frac{\partial \underline{\varepsilon}[k]}{\partial W_k}\right)\times \underline{\varepsilon}[k] \qquad (۴۰)$$

$$H_k \approx 2J_k^T \times J_k = 2A \qquad (۴۱)$$

که در این روابط، $J_k = (\partial \underline{\varepsilon}[k] / \partial W_k)$ ماتریس ژاکوبین بوده و طبق تعریف، برابر با مشتق اول خطای شبکه نسبت به وزن‌ها و بایاس‌ها است. در این صورت، الگوریتم آموزشی لونبرگ-مارکوارت به‌صورت زیر تعریف می‌شود:

$$W_{k+1} = W_k - (J_k^T \times J_k + \mu_k I)^{-1} \times J_k^T \times \underline{\varepsilon}[k] \qquad (۴۲)$$

که در این رابطه، $\mu_k$ یک عدد ثابت است. همان‌طور که ملاحظه می‌کنید، این الگوریتم آموزشی مزیت هر دو روش نیوتن و گرادیان نزولی را دارد؛ چراکه اگر $\mu_k = 0$ باشد، الگوریتم نیوتن حاصل می‌شود و اگر $\mu_k$ بزرگ باشد، الگوریتم گرادیان نزولی با نرخ یادگیری پایین ایجاد می‌شود.

در عمل، الگوریتم لونبرگ – مارکوارت با مقدار کوچکی از $\mu_k$ مانند ۰.۰۱ شروع می‌شود. اگر این انتخاب منجر به کاهش تابع هزینه نشود، گام فعلی دوباره با افزایش $\mu_k$ (مثلاً ده برابر قبلی) برای بهره‌گیری از خواص الگوریتم گرادیان نزولی تکرار می‌شود. درنهایت، وقتی تابع هزینه کاهش یافت، برای مرحله بعد با کاهش $\mu_k$ (مثلاً یک‌دهم قبل) برای بهره‌گیری از خواص الگوریتم نیوتن، محاسبه انجام می‌شود.

در انتهای این بخش بایستی اشاره نمود که به غیر از چهار الگوریتم آموزشی مطرح شده، گرادیان نزولی - گرادیان نزولی با مومنتوم – نیوتن و لونبرگ – مارکوارت، روش‌های دیگری نیز وجود دارند که در جدول۳ تعدادی از این الگوریتم‌ها با معایب و مزایا آن‌ها آورده شده است [۵، ۷۸].

**مثال ۱)** فرض کنید که داده‌های اندازه‌گیری شده از یک سیستم استاتیکی *SISO* مطابق زیر باشد:

$$\{(u[k], y[k]\}_{k=1}^{7} = \{(0,-1),(1,1.3187),(2,3.7073),$$
$$(4,4.0394),(5,5.5627),(6,7.5866)\}$$

هدف این است که با استفاده از الگوریتم آموزشی لونبرگ – مارکوارت، یک شبکه عصبی مناسب برای مدل‌سازی رفتار این سیستم طراحی نماییم. طبق قضیه۱، جهت مدل‌سازی این سیستم می‌توان از یک شبکه عصبی دولایه استفاده نمود که در لایه اول آن از توابع فعال‌ساز حلقوی لگاریتمی و در لایه دوم آن از توابع فعال‌ساز خطی استفاده‌شده باشد. از طرفی، چون سیستم مفروض دارای یک خروجی است، بنابراین باید در لایه آخر شبکه، تنها یک نرون وجود داشته باشد. همچنین، از آنجایی که این سیستم دارای یک ورودی است، لذا شبکه در نظر گرفته‌شده باید تنها یک ورودی داشته و تنها یک ورودی وارد نرون‌های لایه اول گردد. برای تعیین تعداد مناسب نرون‌های لایه اول نیز می‌توان نوشت:

$$S^1 = \frac{N}{\lambda(R+M)} = \frac{7}{2(1+1)} = 1.75 \quad \Rightarrow \quad S^1 = 2$$

به‌صورت تصادفی، مقادیر اولیه زیر را برای ماتریس وزن‌ها و بایاس‌های هر لایه انتخاب می‌کنیم:

$$W_0^1 = \begin{bmatrix}1\\1\end{bmatrix} \quad , \quad \underline{b}_0^1 = \begin{bmatrix}0.5\\-0.5\end{bmatrix}$$
$$W_0^2 = [0.5 \;\; 0.5] \quad , \quad b_0^2 = 0.1$$

در صورتی که از الگوریتم آموزشی لونبرگ – مارکوارت برای یادگیری این شبکه استفاده شود، بعد از ۲۵ تکرار مقدار تابع هزینه تعریف شده در رابطه (۲۰) به مقداری کمتر از ۰.۰۰۱ رسیده و الگوریتم یادگیری متوقف می‌شود و مقادیر زیر برای وزن‌ها و بایاس‌ها بدست می‌آیند:

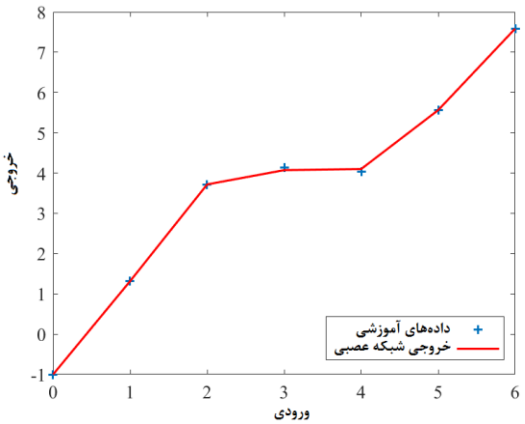
$$W_{25}^1 = \begin{bmatrix}2.6227\\19.2436\end{bmatrix} \quad , \quad \underline{b}_{25}^1 = \begin{bmatrix}-2.664\\-96.5441\end{bmatrix}$$
$$W_{25}^2 = [5.4572 \;\; 3.4844] \quad , \quad b_{25}^2 = -1.3550$$

در شکل۱۰ خروجی شبکه عصبی به همراه داده‌های آموزشی نشان داده‌شده است.

جدول۳. الگوریتم‌های آموزشی متداول برای آموزش شبکه‌های عصبی

| روش آموزشی | نام روش | مزایا و معایب |
|---|---|---|
| آموزش افزایشی[1] | گرادیان نزولی | |
| | گرادیان نزولی با مومنتوم | |
| آموزش دسته‌ای[2] | گرادیان نزولی | کُند |
| | گرادیان نزولی با مومنتوم با نرخ ثابت | کُند اما بهتر از روش قبل |
| | گرادیان نزولی با مومنتوم با نرخ متغیر | کُند اما بهتر از دو روش قبل |
| | فلچر –ریوز[3] (گرادیان مزدوج) | بهتر از سه روش قبل |
| | پلاک-ریبایر[4] | |
| | پاول- بیل[5] | |
| | گرادیان مزدوج مقیاس‌بندی شده[6] | روش پیشنهادی اول برای شبکه‌های بزرگ |
| | روش‌های گوسی - نیوتن | روش پیشنهادی دوم برای شبکه‌های کوچک |
| | لونبرگ – مارکوارت | روش پیشنهادی اول برای شبکه‌های کوچک |

1. Incremental Training
2. Batch Training
3. Fletcher-Reeves
4. Polak-Ribiere
5. Powell–Beale
6. Scaled Conjugate Gradient




شکل ۱۰. خروجی شبکه عصبی به همراه داده‌های آموزشی (مثال ۱)

## ۴-۴ مدل‌سازی آزمایشی با شبکه‌های عصبی مبتنی بر توابع پایه شعاعی (RBF)[1]

همان‌گونه که در بخش‌های قبلی بیان شد، عنصر سازنده یک شبکه عصبی، نرون‌ها هستند. هر نرون به دو شکل نرون جمعی (مدل مک‌کلاچ - پیتز ارائه‌شده در شکل۴) یا نرون شعاعی[2] قابل مدل‌سازی است. تاکنون در بحث مدل‌سازی آزمایشی (شناسایی) سیستم‌ها با شبکه‌های عصبی، نرون‌های جمعی و شبکه‌های عصبی مرتبط با آن‌ها معرفی و بحث شد. در این بخش، نرون‌های شعاعی و شبکه‌های مرتبط با آن تشریح می‌شود.

### ۴-۴-۱ مدل نرون شعاعی

مدل یک نرون شعاعی که به‌صورت گرافیکی در شکل۱۱ نشان داده‌شده است، در حالت کلی به‌صورت رابطه زیر بیان می‌شود [۷۸]:

$$\underline{a} = f\,(\underline{n}) = f\,(\|\underline{w} - \underline{p}\|\underline{b}) \qquad (۴۳)$$

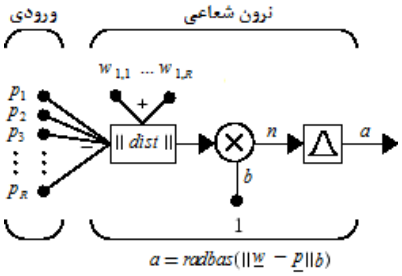
نمادهای به‌کاررفته در این نوع نرون‌ها مشابه نرون‌های جمعی است، اما معمولاً تابع فعال‌ساز $f$ از رابطه توابع پایه شعاعی پیروی می‌کند:

$$f\,(n) = e^{-n^2} \qquad (۴۴)$$

و منظور از $\|\underline{w} - \underline{p}\|$، فاصله بین وزن‌ها و ورودی‌ها است که به‌صورت زیر محاسبه می‌شود:

$$\|\underline{w} - \underline{p}\| = \sqrt{(w_{1,1} - p_1)^2 + \ldots + (w_{1,R} - p_R)^2} \qquad (۴۵)$$

از آنجائی که در توابع پایه شعاعی، اگر مقدار $n$ صفر شود، خروجی تابع ماکزیمم خواهد شد، لذا در یک نرون شعاعی، هرچقدر فاصله ورودی $\underline{p}$ از $\underline{w}$ کم باشد، خروجی نرون بزرگ‌تر خواهد شد. همچنین، هرچقدر بایاس کوچک‌تر باشد، حساسیت نرون افزایش‌یافته و خروجی نرون شعاعی بزرگ‌تر خواهد بود.

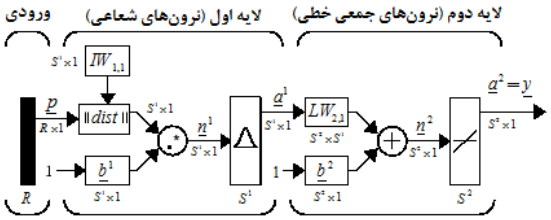



$$a = radbas(\|\underline{w} - \underline{p}\|b)$$

شکل ۱۱. مدل یک نرون شعاعی

### ۴-۴-۲ شبکه‌های عصبی شعاعی (RBF)

چنانچه در یک لایه از شبکه‌های عصبی از نرون‌های شعاعی استفاده شود، یک شبکه عصبی مبتنی بر توابع پایه شعاعی (*RBF*) یا به‌اختصار شبکه عصبی شعاعی ساخته می‌شود. کاربرد اصلی شبکه‌های عصبی شعاعی در مبحث رگرسیون (برازش)، تخمین توابع و مدل‌سازی آزمایشی است. یکی از پرکاربردترین شبکه‌های عصبی شعاعی در شکل۱۲ قابل‌مشاهده است.




شکل۱۲. شبکه عصبی دولایه با لایه اول متشکل از نرون‌های شعاعی و لایه دوم متشکل از نرون‌های جمعی خطی (با نماد فشرده وزن‌ها و بایاس‌ها)

مطابق شکل ۱۲، در لایه اول این شبکه عصبی از نرون‌های شعاعی استفاده‌شده است. اما لایه دوم آن متشکل از نرون‌های جمعی خطی است. معادلات حاکم بر شبکه عصبی شکل۱۲ در لایه اول، با $S^1$ نرون و $R$ ورودی، به‌صورت برداری عبارت است از:

$$\begin{cases} a_1^1 = f^1(\sqrt{(w_{1,1}^1 - p_1)^2 + \ldots + (w_{1,R}^1 - p_R)^2}\; b_1^1) \\ a_2^1 = f^1(\sqrt{(w_{2,1}^1 - p_1)^2 + \ldots + (w_{2,R}^1 - p_R)^2}\; b_2^1) \\ \quad\vdots \\ a_{S^1}^1 = f^1(\sqrt{(w_{S^1,1}^1 - p_1)^2 + \ldots + (w_{S^1,R}^1 - p_R)^2}\; b_{S^1}^1) \end{cases}$$

در نتیجه، برای خروجی لایه اول این شبکه داریم:

$$\underline{a}^1 = f^1(\|W^1 - \underline{p}\|\underline{b}^1) \qquad (۴۶)$$

برای لایه دوم این شبکه عصبی، با $S^2$ نرون و $S^1$ ورودی، نیز می‌توان نوشت:

$$\begin{cases} a_1^2 = f^2(w_{1,1}^2 a_1^1 + \ldots + w_{1,S^1}^2 a_{S^1}^1 + b_1^2) \\ a_2^2 = f^2(w_{2,1}^2 a_1^1 + \ldots + w_{2,S^1}^2 a_{S^1}^1 + b_2^2) \\ \quad\vdots \\ a_{S^2}^2 = f^2(w_{S^2,1}^2 a_1^1 + \ldots + w_{S^2,S^1}^2 a_{S^1}^1 + b_{S^2}^2) \end{cases}$$

1. Radial Basis Function (*RBF*)
2. Radial Neuron

بنابراین

$$\underline{a}^2 = f^2(W^2\underline{a}^1 + \underline{b}^2) \qquad (۴۷)$$

### ۳-۴-۴ الگوریتم آموزش شبکه‌های عصبی شعاعی

برای آموزش شبکه‌های عصبی شعاعی، مراحل زیر طی می‌شود:

۱. ابتدا تعداد نرون‌های لایه اول صفر فرض می‌شود.
۲. وزن‌ها و بایاس‌های لایه دوم صفر می‌گردد (پرواضح است که خروجی شبکه عصبی نیز صفر خواهد شد).
۳. از مقایسه مقدار صفر لایه دوم با خروجی‌های مطلوب (اندازه‌گیری شده)، بیش‌ترین خطا محاسبه می‌شود.
۴. یک نرون برای لایه اول ایجاد می‌شود و وزن این نرون برابر با ورودی دارای بیش‌ترین خطا انتخاب می‌شود.
۵. بایاس لایه اول نیز با توجه به پارامتر گسترش[1] تنظیم می‌شود.
۶. وزن و بایاس لایه دوم از روش حداقل مربعات[2] (*LS*) محاسبه می‌گردد.
۷. مراحل بالا تکرار می‌شود.

**مثال ۲)** داده‌های ورودی و خروجی زیر را در نظر بگیرید:

$$y = g(u)$$
$$\{(u[k], y[k])\}_{k=1}^{3} = \{(4,16),(2,4),(3,9)\}$$

هدف این است که با استفاده از شبکه‌های عصبی شعاعی، وزن‌ها و بایاس‌ها محاسبه شوند. ابتدا مقدار خروجی شبکه عصبی را محاسبه می‌کنیم:

$$\begin{cases} a^2[1] = 0 \\ y[1] = 16 \end{cases} \Rightarrow \varepsilon[1] = 16 - 0 = 16$$

$$\begin{cases} a^2[2] = 0 \\ y[2] = 4 \end{cases} \Rightarrow \varepsilon[2] = 4 - 0 = 4$$

$$\begin{cases} a^2[3] = 0 \\ y[3] = 9 \end{cases} \Rightarrow \varepsilon[3] = 9 - 0 = 9$$

همان‌طور که ملاحظه می‌شود، خطای ورودی اول حداکثر است؛ بنابراین یک نرون به لایه اول اضافه کرده و مقدار وزن آن را برابر با مقدار ورودی یا همان مقدار ۴ انتخاب می‌کنیم. مقدار بایاس نیز همواره برابر با ۰.۸۳۲۶ انتخاب می‌شود (البته می‌توان مقدار پارامتر گسترش را تغییر داد؛ مقدار بایاس از حاصل تقسیم ۰.۸۳۲۶ بر مقدار پارامتر گسترش محاسبه می‌شود). بنابراین داریم:

$$IW_{1,1} = 4 \;,\; b_1^1 = \frac{0.8326}{1} = 0.8326$$

در نتیجه، خروجی لایه اول به‌ازای این دو مقدار برابر خواهد بود با:

$$\begin{aligned} n &= \| w_{1,1}^1 - p \| \times b_1^1 \\ &= [\sqrt{(4-4)^2} \;\; \sqrt{(4-2)^2} \;\; \sqrt{(4-3)^2}\,] \times 0.8326 \\ &= [0 \;\; 1.6652 \;\; 0.8326] \end{aligned}$$

$$\begin{aligned} a_1^1 = f(n) &= [e^{-(0)^2} \;\; e^{-(1.6652)^2} \;\; e^{-(0.8326)^2}\,] \\ &= [1 \;\; 0.0625 \;\; 0.5] \end{aligned}$$

حال برای محاسبه وزن و بایاس لایه دوم متشکل از نرون جمعی با تابع فعال‌ساز خطی، از روش حداقل مربعات استفاده می‌شود. برای خروجی لایه دوم (خروجی شبکه) می‌توان نوشت:

$$y = w_1^2 a_1^1 + b^2 = [a_1^1 \;\; 1]\begin{bmatrix} w_1^2 \\ b^2 \end{bmatrix}$$

معادله فوق را می‌توان به‌صورت رگرسیون خطی $y = \underline{\varphi}^T \underline{\theta}$ نوشت؛ به‌طوری‌که

$$\underline{\varphi} = \begin{bmatrix} a_1^1 \\ 1 \end{bmatrix} \;,\; \underline{\theta} = \begin{bmatrix} w_1^2 \\ b^2 \end{bmatrix}$$

در نتیجه خواهیم داشت:

$$\begin{cases} y[1] = [a_1^1[1] \;\; 1]\begin{bmatrix} w_1^2 \\ b^2 \end{bmatrix} \\ y[2] = [a_1^1[2] \;\; 1]\begin{bmatrix} w_1^2 \\ b^2 \end{bmatrix} \\ y[3] = [a_1^1[3] \;\; 1]\begin{bmatrix} w_1^2 \\ b^2 \end{bmatrix} \end{cases} \Rightarrow \begin{bmatrix} 16 \\ 4 \\ 9 \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 0.0625 & 1 \\ 0.5 & 1 \end{bmatrix}\begin{bmatrix} w_1^2 \\ b^2 \end{bmatrix}$$

حال به روش حداقل مربعات، می‌توان مقادیر وزن و بایاس لایه دوم را تعیین نمود:

$$\begin{aligned} \begin{bmatrix} w_1^2 \\ b^2 \end{bmatrix} &= \left( \begin{bmatrix} 1 & 1 \\ 0.0625 & 1 \\ 0.5 & 1 \end{bmatrix}^T \begin{bmatrix} 1 & 1 \\ 0.0625 & 1 \\ 0.5 & 1 \end{bmatrix} \right)^{-1} \begin{bmatrix} 1 & 1 \\ 0.0625 & 1 \\ 0.5 & 1 \end{bmatrix}^T \begin{bmatrix} 16 \\ 4 \\ 9 \end{bmatrix} \\ &= \begin{bmatrix} 12.8284 \\ 2.9852 \end{bmatrix} \end{aligned}$$

با تعیین شدن مقادیر وزن و بایاس لایه دوم، می‌توان خروجی شبکه عصبی را محاسبه نمود:

$$a^2[1] = 12.8284 \times 1 + 2.9852 = 15.8136$$
$$a^2[2] = 12.8284 \times 0.0625 + 2.9852 = 3.787$$
$$a^2[3] = 12.8284 \times 0.5 + 2.9852 = 9.3994$$

چنانچه مقدار MSE، بر مبانی خروجی‌های واقعی و خروجی‌های محاسبه‌شده توسط شبکه قابل‌قبول باشد، فرایند آموزش متوقف می‌شود. در غیر این صورت، فرایند بالا تکرار می‌شود. بنابراین، مجدداً ابتدا مقدار خروجی شبکه محاسبه می‌شود:

$$\begin{cases} a^2[1] = 15.8136 \\ y[1] = 16 \end{cases} \Rightarrow \varepsilon[1] = 16 - 15.8136 = 0.1864$$

$$\begin{cases} a^2[2] = 3.787 \\ y[2] = 4 \end{cases} \Rightarrow \varepsilon[2] = 4 - 3.787 = 0.213$$

$$\begin{cases} a^2[3] = 9.3994 \\ y[3] = 9 \end{cases} \Rightarrow \varepsilon[3] = 9 - 9.3994 = -0.3994$$

1. Spread
2 Least squares

ملاحظه می‌شود که خطای ورودی دوم حداکثر است. بنابراین، یک نرون دیگر به لایه اول اضافه کرده و مقدار وزن آن را برابر با مقدار ورودی یا همان مقدار ۲ انتخاب می‌کنیم. مقدار بایاس را نیز برابر با مقدار بایاس محاسبه‌شده در مرحله قبل یا همان ۰.۸۳۲۶ انتخاب می‌کنیم. درنتیجه خواهیم داشت:

$$a_2^1 = f\ (n)$$
$$n = \| w_{2,1}^1 - p \| \times b_2^1$$
$$= [\sqrt{(2-4)^2}\ \ \sqrt{(2-2)^2}\ \ \sqrt{(2-3)^2}\ ] \times 0.8326$$
$$= [1.6652\ \ \ 0\ \ 0.8326]$$
$$a_2^1 = f\ (n) = [e^{-(1.6652)^2}\ \ e^{-(0)^2}\ \ e^{-(0.8326)^2}]$$
$$= [0.0625\ \ 1\ \ 0.5]$$

در ادامه، برای محاسبه وزن و بایاس لایه دوم، از روش حداقل مربعات استفاده می‌کنیم. برای خروجی لایه دوم می‌توان نوشت:

$$y = w_1^2 a_1^1 + w_2^2 a_2^1 + b^2$$
$$= [a_1^1\ \ a_2^1\ \ 1][w_1^2\ \ w_2^2\ \ b^2]^T = \underline{\varphi}^T \underline{\theta}$$

که به‌صورت یک رگرسیون خطی قابل‌بیان است. بنابراین از روش حداقل مربعات می‌توان برای تخمین وزن‌ها و بایاس لایه دوم استفاده نمود:

$$\begin{cases} y[1] = [a_1^1[1]\ \ a_2^1[1]\ \ 1] \begin{bmatrix} w_1^2 \\ w_2^2 \\ b^2 \end{bmatrix} \\ y[2] = [a_1^1[2]\ \ a_2^1[2]\ \ 1] \begin{bmatrix} w_1^2 \\ w_2^2 \\ b^2 \end{bmatrix} \\ y[3] = [a_1^1[3]\ \ a_2^1[3]\ \ 1] \begin{bmatrix} w_1^2 \\ w_2^2 \\ b^2 \end{bmatrix} \end{cases}$$

$$\Rightarrow \begin{bmatrix} 16 \\ 4 \\ 9 \end{bmatrix} = \begin{bmatrix} 1 & 0.0625 & 1 \\ 0.0625 & 1 & 1 \\ 0.5 & 0.5 & 1 \end{bmatrix} \begin{bmatrix} w_1^2 \\ w_2^2 \\ b^2 \end{bmatrix}$$

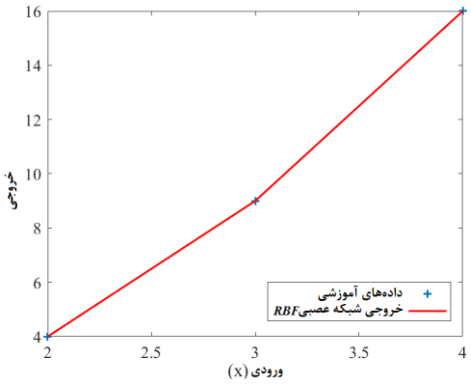
درنتیجه خواهیم داشت:

$$[w_1^2\ \ w_2^2\ \ b^2] = [22.4\ \ 9.6\ \ -7]^T$$

با ادامه این روند تا تکراری که مقدار تابع هزینه (MSE) کمتر یا مساوی با مقدار ۰.۰۰۱ باشد، نتایج زیر به دست می‌آیند:

$$\underline{b}^1 = \begin{bmatrix} 0.8326 \\ 0.8326 \end{bmatrix},\ b^2 = -7$$
$$\underline{W}^1 = [4\ \ 2]^T,\ \underline{W}^2 = 22.4\ \ \ 9.6$$

در شکل۱۱، خروجی واقعی و خروجی شبکه عصبی RBF طراحی شده، با یکدیگر مقایسه شده‌اند.



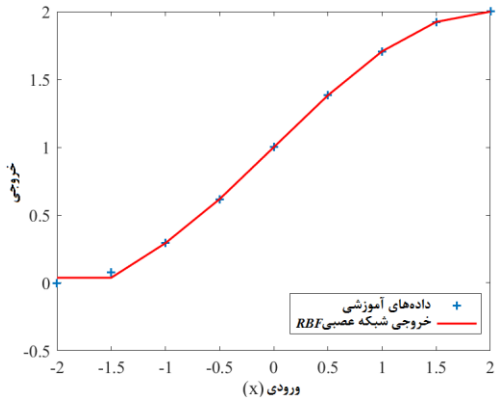

شکل۱۱. مقایسه خروجی واقعی و خروجی شبکه عصبی شعاعی طراحی‌شده برای مثال۲

**مثال۳)** داده‌های ورودی – خروجی ارائه شده در جدول۴ را در نظر بگیرید که از یک سیستم استاتیکی SISO اندازه‌گیری شده‌اند.

جدول۴. داده‌های ورودی و خروجی برای مثال۳

| | | | | | |
|---|---|---|---|---|---|
| ***x*** | -2 | -1.5 | -1 | -0.5 | 0 |
| ***y*** | 0 | 0.0761 | 0.2929 | 0.6173 | 1 |
| ***x*** | 0.5 | 1 | 1.5 | 2 | |
| ***y*** | 1.3827 | 1.7071 | 1.9239 | 2 | |

هدف این است که یک شبکه عصبی شعاعی یا RBF برای مدل‌سازی رفتار این سیستم طراحی گردد. با استفاده از داده‌های آموزشی داده شده (جدول۴)، رفتار این سیستم را می‌توان با ساخت یک شبکه عصبی شعاعی دو لایه مدل نمود. این شبکه عصبی شعاعی در لایه پنهان دارای نرون‌های شعاعی با تابع تبدیل radbas و در لایه خروجی دارای نرون‌های جمعی با تابع تبدیل خطی است. پس از یادگیری این شبکه عصبی شعاعی با الگوریتم تشریح شده، منحنی خروجی تخمین زده توسط شبکه عصبی شعاعی طراحی شده مطابق شکل۱۲ به دست می‌آید.

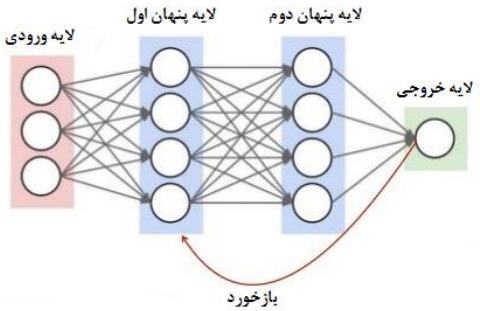



شکل۱۲. خروجی واقعی و خروجی شبکه عصبی شعاعی طراحی‌شده برای مثال۳

## ۴-۵ مدل‌سازی آزمایشی سیستم‌های دینامیکی با شبکه عصبی بازگشتی

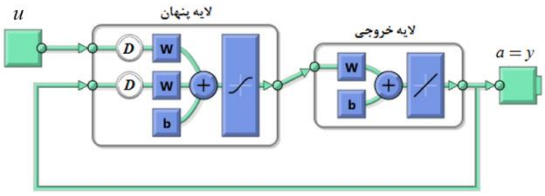
شبکه عصبی بازگشتی[1]، شبکه‌ای است که دارای بازخورد/فیدبک[2] می‌باشد. به‌عبارت‌دیگر، در این نوع از شبکه‌های عصبی، تعدادی از خروجی‌ها به‌عنوان ورودی در گام‌های دیگر مورداستفاده قرار می‌گیرند. در شکل۱۳، نمای کلی از یک شبکه عصبی بازگشتی (*RNN*) دو لایه نشان داده‌شده است [۵، ۷۹-۷۸، ۸۱].

شبکه‌های عصبی بازگشتی نیز مشابه شبکه‌های عصبی پیشرو می‌توانند دارای چندین لایه باشند و از نمادهای وزن و بایاس ارائه شده برای شبکه‌های عصبی پیشرو، در این نوع از شبکه‌های عصبی نیز استفاده می‌شود. بنابراین، چنانچه در یک شبکه پیشرو بازخورد ایجاد شود، در واقع شبکه عصبی پیشرو به شبکه عصبی بازگشتی تبدیل می‌شود.



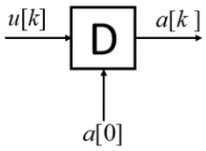

شکل۱۳. نمای کلی از یک شبکه عصبی بازگشتی دولایه

مفهوم جدید شبکه‌های عصبی بازگشتی (*RNN*)، المان تأخیر[3] است. مطابق شکل۱۳، المان تأخیر که با نماد $D$ قبل از نماد وزن در مدل نرون‌ها ظاهر می‌شود، جهت نمایش ارتباط بین خروجی گام فعلی با مقدار ورودی گام قبلی به کار می‌رود. مطابق شکل۱۳، ورودیِ حقیقی مقدارِ $u[k]$ با تأخیر $D$ وارد شبکه عصبی شده است. چنانچه این تأخیر برابر با یک گام زمانی باشد، آنگاه $u[k-1]$ به‌عنوان ورودی به نرون وارد می‌شود. حال چنانچه این تأخیر برابر با دو گام زمانی باشد، آنگاه $u[k-2]$ به‌عنوان ورودی به نرون وارد می‌شود. بنابراین در حالت کلی، اگر تأخیر برابر با $d$ گام زمانی باشد، آنگاه نرون ورودی $u[k-d]$ را در محاسبات استفاده خواهد کرد. همین توضیحات در رابطه با ورودی بازخورد شده $y[k]$ نیز صادق است.




شکل۱۳. شبکه عصبی بازگشتی با دو لایه

به‌عنوان جمع‌بندیِ مفاهیمِ المان تأخیر در شبکه‌های عصبی بازگشتی، می‌توان شکل۱۴ را همواره در نظر گرفت. در این شکل، المان تأخیر یک واحدی $D$ باعث می‌شود که سیگنال $u[k]$ پس از ورود به بخش تأخیر، با یک تأخیر خارج شود ($a[k]=u[k-1]$)، درحالی‌که مقدار اولیه سیگنال خروجی المان تأخیر برابر با $a[0]$ است. باید توجه داشت که خروجی در گام اول ($k=1$)، با استفاده از ورودی قابل‌محاسبه نیست و لذا مقدار خروجی در گام اولیه باید قبلاً تعیین‌شده باشد.




شکل۱۴. نمایی از المان تأخیر یک واحدی (D=1)

### ۱-۵-۴ معادله ریاضی حاکم بر شبکه‌های عصبی بازگشتی

شبکه عصبی بازگشتی نمایش‌داده شده در شکل۱۳ را دوباره در نظر بگیرید. اگر بخواهیم معادله ریاضی بین ورودی و خروجی این شبکه عصبی را بنویسیم، رابطه کلی زیر حاکم است:

$$y[k]=f\left(u[k-1],...,u[k-d],y[k-1],...\right.$$
$$\left.\ldots,y[k-d]\right) \qquad (۴۸)$$

پرواضح است که معادله فوق یک ساختار *NARX* است و بیانگر یک سری زمانی بوده که خروجی $y[k]$ در هر لحظه تابعی از مقادیر گذشته خودش ($y[k-1],...,y[k-d]$) و مقادیر گذشته ورودی ($u[k-1],...,u[k-d]$) است. به‌عبارت‌دیگر، یک معادله تفاضلی قابل‌بیان با رابطه فوق است و لذا می‌توان با داشتن ورودی‌ها و خروجی‌های یک آزمایش، یک شبکه عصبی بازگشتی برای معادله تفاضلی ایجاد نمود و سپس با الگوریتم‌های آموزشی مقادیر وزن، بایاس و میزان تأخیر آن را محاسبه نمود.

همانند شبکه‌های عصبی پیشرو، می‌توان قضیه زیر را برای تعیین ساختار شبکه عصبی بازگشتی استفاده نمود [۵، ۷۸].

**قضیه۳)** هر معادله تفاضلی (در حالت کلی با ساختار NARX) را می‌توان با یک شبکه عصبی بازگشتی مدل‌سازی نمود، به‌شرط آنکه تعداد ورودی‌های شبکه عصبی برابر با ورودی‌های معادله تفاضلی بوده و همچنین تعداد نرون‌های لایه آخر شبکه برابر با تعداد خروجی‌های معادله تفاضلی باشد. همچنین، تعداد لایه‌های پنهان نیز متناسب با پیچیدگی مسئله تعیین گردد. چنانچه معادله تفاضلی خطی باشد، نیاز به لایه پنهان نیست. در غیر این صورت، یک لایه پنهان (یا بیشتر) قابل‌استفاده است.

**مثال ٤)** بردارهای ورودی و خروجی زیر از یک سیستم دینامیکی با زمان نمونه‌برداری یک ثانیه جمع‌آوری‌شده است:

$$X=[1,1,1,-1,1,-1,-1,1]$$
$$Y=[-1.68,-1.11,1.51,4.63,5.54,5.05,3.91,-0.01]$$

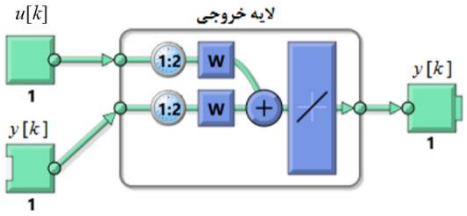
ساختار مدل را به‌صورت معادله تفاضلی زیر در نظر بگیرید و هدف تخمین پارامترهای مجهول این ساختار است:

$$y[k]=ax[k-1]+bx[k-2]$$
$$+cy[k-1]+dy[k-2]$$

1. Recurrent Neural Network (RNN)
2. Feedback
3. Delay

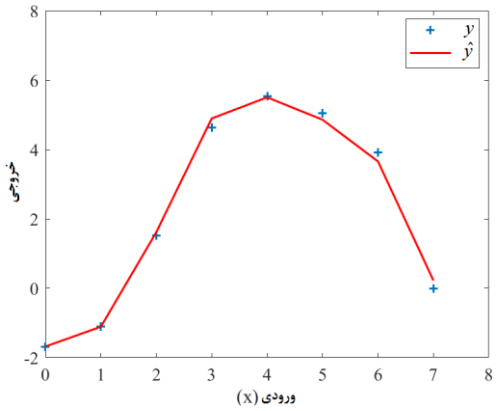
با توجه به اینکه ساختار داده‌شده یک ساختار خطی است، می‌توان نشان داد که معادله رگرسیون آن از نوع خطی بوده و درنتیجه، به روش حداقل مربعات خطی می‌توان پارامترهای مجهول ساختار را تخمین زد و در این مقاله، به این روش پرداخته نمی‌شود. در اینجا جهت حل این مسئله، از شبکه‌های عصبی بازگشتی استفاده می‌کنیم، به‌طوری‌که با طراحی یک شبکه عصبی مناسب، بر اساس داده‌های داده‌شده، به آموزش شبکه عصبی (بازگشتی) پرداخته و مقادیر $a$، $b$، $c$ و $d$ را محاسبه می‌کنیم.

با توجه به اینکه معادله تفاضلی داده‌شده به‌صورت خطی است، با استفاده از قضیه۳، شبکه عصبی یک لایه شکل۱۵ را که معادل با معادله تفاضلی داده‌شده است، در نظر می‌گیریم.




شکل۱۵. شبکه عصبی بازگشتی در نظر گرفته‌شده برای مثال۴

این شبکه عصبی مشابه شبکه‌های عصبی پیشرو بوده و دارای یک نرون با چهار ورودی و یک خروجی است. در این راستا، برای این شبکه عصبی، چهار ورودی فرض شده و مقادیر وزن‌ها در شروع کار (مقادیر اولیه) صفر قرار داده می‌شوند. توجه داشته باشید که به دلیل ساختار خطی معادله تفاضلی داده‌شده، برای شبکه عصبی مفروض بایاس در نظر گرفته نشده است. این وزن‌ها، پس از کاهش خطا به مقدار تعیین‌شده $(5\times10^{-2})$ که در گذر ۱۳۰ام اتفاق افتاده است، حاصل می‌شود. محاسبه و ترسیم خروجی محاسبه شده با شبکه عصبی بازگشتی $(a \equiv \hat{y})$ و خروجی واقعی اندازه‌گیری شده از سیستم $(y)$ است که در شکل۱۶ نمایش داده‌شده است. حتماً توجه دارید که در محاسبات $\hat{y}$، تنها به دو مقدار اولیه $y$ نیاز است و بقیه مقادیر $\hat{y}$ توسط معادله تفاضلی به‌دست‌آمده محاسبه می‌شود (طبق صورت‌مسئله، فرض شده است که هر داده به فاصله زمانی یک ثانیه جمع‌آوری‌شده است).




شکل۱۶. منحنی خروجی شبکه عصبی (بازگشتی) طراحی‌شده برای مثال۴

## ۵- کاربرد شبکه‌های عصبی در کنترل سیستم‌های پیچیده

شبکه‌های عصبی علاوه بر مدل‌سازی، در کنترل سیستم‌های دینامیکی پیچیده نیز عملکرد موفقی دارند. توانایی تخمین و جامع شبکه‌های عصبی، آن‌ها را انتخاب مناسبی برای کنترل سیستم‌های غیرخطی قرار داده است. در این بخش از مقاله، مقدمه‌ای از سه معماری پرطرفدار شبکه‌های عصبی برای پیش‌بینی و کنترل را مطرح می‌کنیم که عبارت‌اند از [۵، ۷۹، ۸۱]:

- کنترل پیش‌بین مدل[1]
- کنترل NARMA-L2 [2]
- کنترل مدل مرجع[3]

عموماً دو گام برای استفاده از شبکه‌های عصبی در سیستم‌های کنترل وجود دارد:

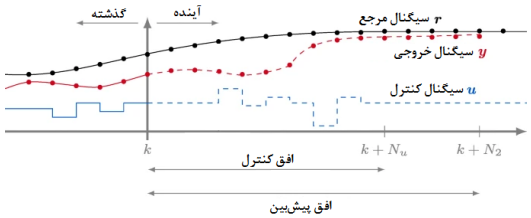
۱. تعریف سیستم
۲. طراحی کنترل‌کننده

در مرحله تعریف سیستم، یک شبکه عصبی برای کنترل یک سیستم (پلنت، ماشین یا دستگاه و ...) تعریف شده و برای کنترل سیستم موردنظر آموزش داده می‌شود. در هر یک از سه معماری کنترلی مذکور در بالا، مرحله تعریف سیستم دارای روال ثابتی است. اما مرحله طراحی کنترل‌کننده برای هر یک از آن‌ها متفاوت است. به‌طورخلاصه، در کنترل پیش‌بین مدل، از مدل سیستم/پلنت برای پیش‌بینی رفتار آینده سیستم استفاده می‌شود و سپس از یک الگوریتم بهینه‌سازی برای تعیین سیگنال‌های کنترلی که تابع هزینه (کارایی) را در آینده بهینه می‌کنند، استفاده می‌گردد. در کنترل *NARMA-L2*، کنترل‌کننده مدل ساده‌ای از سیستم مورد مدل‌سازی است. در کنترل مدل مرجع، کنترل‌کننده یک شبکه عصبی است که برای کنترل سیستم آموزش داده می‌شود و از یک مدل مرجع پیروی می‌کند. از مدل شبکه عصبی برای همکاری در روند آموزش کنترل‌کننده استفاده می‌شود. در ادامه این بخش، این سه معماری کنترلی بیشتر مورد بحث و بررسی قرار می‌گیرد. توجه داشته باشید که هر یک از این معماری‌های کنترلی یا به زبان ساده‌تر کنترل‌کننده‌ها، دارای نقاط قوت و ضعف می‌باشند و نمی‌توان یک معماری کنترلی را برای همه کاربردها مناسب دانست.

### ۵-۱ استفاده از شبکه‌های عصبی به عنوان مدل پیش‌بین در کنترل‌کننده‌های پیش‌بین مدل

کنترل‌کننده‌های پیش‌بین مدل بر اساس بهینه‌سازی تکراری (برخط) یک تابع هزینه با افق محدود از عملکرد سیستم کار می‌کند. شکل ۱۷ اصول کار این کنترل‌کننده پیشرفته را نشان می‌دهد [۷۷].

1. Model Predictive Control (MPC)
2NonlinearAutoRgressive-Moving Average-L2 (NARMA-L2)
3. Model Reference Control



شکل۱۷. اصول کلی کنترل پیش‌بین مدل

مطابق شکل فوق، فرض کنید که وضعیت تمام سیگنال‌های موجود در سیستم تا نمونه $k$ام در دسترس باشد، یعنی حالت‌ها و خروجی‌های سیستم از نمونه صفر تا نمونه $k$ام و همچنین سیگنال کنترلی نیز از نمونه صفر تا $k$-1ام در دسترس باشند. توجه داریم که اگر زمان نمونه‌برداری مشخص باشد، شماره نمونه به‌راحتی تبدیل به زمان واقعی خواهد شد ($t = k\,T_s$، که $T_s$ زمان نمونه‌برداری است). هدف این است که سیگنال‌های کنترلی از نمونه $k$ تا نمونه $k+N_u$ را طوری تعیین کنیم که خروجی سیستم از نمونه $k$ تا $k+N_2$ تا حد ممکن به سیگنال مرجع (خروجی مطلوب) نزدیک باشد. برای این منظور، باید خروجی سیستم از زمان $k$ تا زمان $k+N_2$ پیش‌بینی شود که در صورت معلوم بودن مدل دینامیکی سیستم، به‌راحتی می‌توان پیش‌بینی را انجام داد. به پارامتر $N_2$ افق پیش‌بینی[1] گفته می‌شود و در واقع طول بازه پیش‌بینی را نشان می‌دهد. به پارامتر $N_u$ نیز افق کنترل[2] می‌گویند. به‌عنوان‌مثال، اگر فرض کنیم که سیستم دینامیکی گسسته سیستم غیرخطی موردبررسی به صورت ساختار *NARX* توصیف شده و خروجی واقعی آن به صورت زیر باشد:

$$y[k] = f\,(y[k-1],...,y[k-n],u[k],\\ ...,u[k-m]) + e[k] \qquad (۴۹)$$

آنگاه، با توجه به اینکه اطلاعات خروجی سیستم تا نمونه $k$ و اطلاعات سیگنال کنترلی تا نمونه $k$-1 مشخص است، لذا خروجی یک گام به جلو به صورت زیر قابل پیش‌بینی خواهد بود:

$$\hat{y}[k+1|k] = f\,(y[k],...,y[k+1-n],\\ u[k+1],...,u[k+1-m]) \qquad (۵۰)$$

از علامت $\hat{y}$ برای نشان دادن مقادیر پیش‌بینی استفاده می‌شود و $\hat{y}(k+1|k)$ به معنی پیش‌بینی یک گام به جلو با داشتن اطلاعات تا زمان $k$ است. حال با جایگذاری $k$+1 به جای $k$ در رابطه بالا، پیش‌بینی دو گام به جلو به دست می‌آید:

$$\hat{y}[k+2|k] = f\,(y[k+1],...\\ ...,u[k+2],...,u[k+2-m]) \qquad (۵۱)$$

به همین ترتیب، با جایگذاری‌های متوالی می‌توان پیش‌بینی تا $N_2$ گام به جلو را محاسبه نمود:

$$\hat{y}[k+N_2|k] = f\,(y[k+N_2-1],...\\ ...,u[k+N_2],...,u[k+N_2-m]) \qquad (۵۲)$$

پرواضح است که تمام پیش‌بینی‌ها، وابسته به سیگنال کنترلی در طول افق پیش‌بین (یعنی $u[k]$ تا $u[k+N_2-1]$) است که در صورتی که $N_u < N_2$ باشد، فقط سیگنال‌های کنترلی $u[k]$ تا $u[k+N_u]$ را در پیش‌بینی‌ها در نظر گرفته و بقیه سیگنال‌های کنترلی را صفر در نظر می‌گیریم. حال برای اینکه پیش‌بینی‌ها تا حد امکان به خروجی‌های مرجع نزدیک باشند، از تابع هزینه مجموع مربعات خطای ردیابی استفاده می‌شود. خطای ردیابی $n$ گام به جلو به صورت زیر تعریف می‌شود:

$$e[k+n] = y_{ref}[k+n] - \hat{y}[k+n|k] \qquad (۵۳)$$

که در این رابطه، $n$ می‌تواند از ۱ تا $N_2$ تغییر کند و $y_{ref}[k+n]$ مقدار سیگنال مرجع است. حال تابع هزینه را به صورت مجموع مربعات خطا تعریف می‌کنیم:

$$J = \sum_{i=1}^{N_2} \left( y_{ref}[k+n] - \hat{y}[k+n|k] \right)^2 \qquad (۵۴)$$

با بهینه‌سازی تابع هزینه فوق، سیگنال کنترلی در طول افق کنترل به دست می‌آید. اما همان‌طور که قبلاً اشاره شد، علاوه بر حداقل‌سازی تابع هزینه فوق، می‌توان در کنترل پیش‌بین مدل قیدهای فیزیکی نیز در نظر گرفت که از مزیت‌های این نوع کنترل‌کننده به شمار می‌رود. از طرفی، بهینه‌سازی تابع هزینه (۵۴) ممکن است منجر به سیگنال کنترلی با دامنه بالا شود. برای رفع این مشکل عموماً در کنترل پیش‌بین مدل، انرژی سیگنال کنترلی در طول افق کنترل نیز به تابع هزینه اضافه می‌شود:

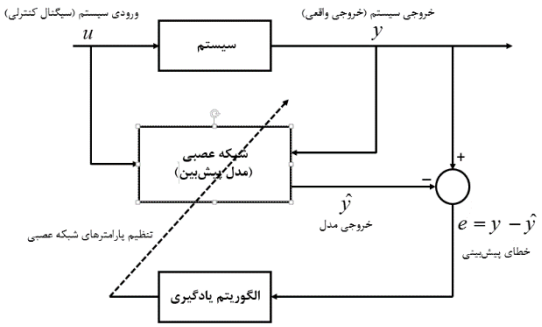
$$J = \sum_{i=1}^{N_2} \left( y_{ref}[k+n] - \hat{y}[k+n] \right)^2 \\ + \lambda \sum_{i=1}^{N_u} u^2[k+n] \qquad (۵۵)$$

پارامتر $\lambda$ در این تابع هزینه به‌منظور تنظیم اهمیت نسبی خطای ردیابی نسبت به دامنه سیگنال کنترلی استفاده می‌شود. هر چه خطای ردیابی نسبت به دامنه سیگنال کنترلی اهمیت بیشتری داشته باشد، اندازه $\lambda$ کوچک‌تر انتخاب می‌شود. توجه داریم که تابع هزینه تعریف شده در (۵۵) دارای متغیرهای $u[k]$ تا $u[k+N_u]$ است و با حل مسئله بهینه‌سازی، مقادیر بهینه آن‌ها محاسبه می‌شوند. اما طبق اصل افق کاهنده، فقط اولین مقدار محاسبه شده مربوط به سیگنال کنترلی بهینه به سیستم اعمال می‌شود و در زمان بعدی، یعنی $k$+1، تمام محاسبات قبل دوباره تکرار می‌شود. بنابراین، فقط $u[k]$ به سیستم اعمال می‌شود و اثر آن یعنی $y[k+1]$ اندازه‌گیری می‌شود. حال ما اطلاعات خروجی سیستم تا زمان $k$+1 را داریم و دوباره با طی کردن مسیر قبل، سیگنال کنترلی بهینه جدید، یعنی $u[k+1]$، را به دست می‌آوریم و آن را به سیستم اعمال می‌کنیم تا

1 Prediction horizon
2 Control horizon

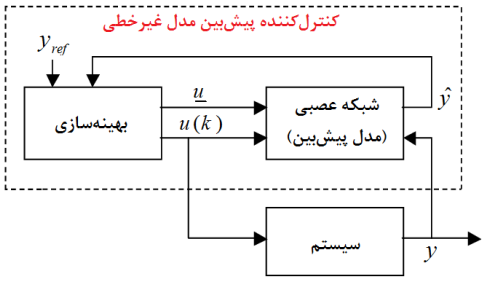
$y[k+2]$ به دست آید و این فرآیند همواره در هر گام زمانی تکرار می‌شود. دلیل استفاده از اصل افق کاهنده، کاهش اثر عدم قطعیت‌ها، اغتشاشات و نویزهای موجود در سیستم است؛ چراکه این سیگنال‌ها اغلب ماهیت تصادفی دارند و تا زمانی که رخ ندهند، نمی‌توانیم آن‌ها را اندازه‌گیری کنیم. حال اگر سیگنال‌های کنترلی $u[k]$ تا $u[k+N_u]$ به سیستم اعمال شود، بدین معنی است که اثرات اغتشاشات در طول افق پیش‌بین لحاظ نخواهد شد.

در طراحی کنترل‌کننده‌های پیش‌بین مدل برای سیستم‌های غیرخطی، می‌توان از شبکه‌های عصبی به عنوان مدل پیش‌بین برای پیش‌بینی رفتار آینده سیستم غیرخطی استفاده نمود. بدین منظور، از یک شبکه عصبی برای ارائه یک مدل دینامیکی مستقیم از سیستم استفاده می‌کنیم. برای آموزش شبکه عصبی، خطای پیش‌بینی بین خروجی سیستم و خروجی شبکه عصبی مورداستفاده قرار می‌گیرد. این روند در شکل۱۸ نمایش داده شده است. شبکه عصبی از ورودی‌ها و خروجی‌های لحظات قبل سیستم برای پیش‌بینی مقادیر آینده خروجی سیستم استفاده می‌کند. ساختار شبکه عصبی مورداستفاده در این ساختار می‌تواند یک شبکه عصبی پیشرو با دولایه باشد که ورودی‌های آن به ترتیب ورودی‌ها و خروجی‌های گذشته سیستم باشد. این شبکه عصبی می‌تواند به صورت برون‌خط و دسته‌ای آموزش داده شود. داده‌های آموزشی با توجه به کارکرد سیستم جمع‌آوری می‌شود. هر یک از الگوریتم‌های یادگیری پس‌انتشار (مطرح شده در بخش قبلی) در فرآیند یادگیری قابل‌استفاده می‌باشند. روش کنترل پیش‌بین مدل مبتنی بر اصل افق کاهنده است. شبکه عصبی رفتار سیستم را در طول افق پیش‌بین، پیش‌بینی می‌کند. این پیش‌بینی‌ها در تابع هزینه ارائه شده در رابطه (۵۵) جایگذاری شده و با حل یک مسئله بهینه‌سازی عددی در هر گام زمانی، سیگنال‌های کنترلی $u[k]$ تا $u[k+N_u]$ تعیین می‌گردند و نهایتاً طبق اصل افق کاهنده، فقط سیگنال کنترلی $u[k]$ به سیستم اعمال شده و در گام زمانی بعدی، مجدداً این مسئله بهینه‌سازی برای تعیین سیگنال کنترلی حل می‌گردد.




شکل۱۸. استفاده از شبکه عصبی به عنوان مدل پیش‌بین

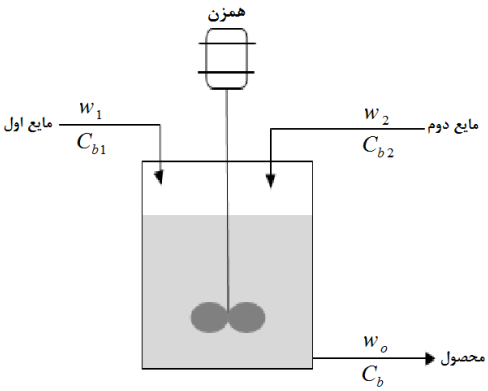
بلوک دیاگرام شکل۱۹، فرآیند کنترل پیش‌بین مدل با شبکه عصبی (به عنوان پیش‌بینی‌کننده رفتار آینده سیستم) را نشان می‌دهد.




شکل۱۹. فرآیند کنترل‌پیش‌بین مدل غیرخطی با یک شبکه عصبی

این کنترل‌کننده شامل دو بخش است: یک بخش برای پیش‌بینی خروجی‌های آینده است که توسط یک شبکه عصبی انجام می‌شود. بخش دوم این کنترل‌کننده، بخش بهینه‌سازی است که در هر گام زمانی مسئله بهینه‌سازی با تابع هزینه (۵۵) را برای تعیین سیگنال‌های کنترلی $\underline{u}=[u[k]\ u[k+1]\ \dots\ u[k+N_u]]^T$ حل می‌کند و نهایتاً طبق اصل افق کاهنده، فقط سیگنال کنترلی $u[k]$ به سیستم اعمال می‌شود.

**مثال۵)** سیستم *CSTR* نشان داده شده در شکل ۲۰ را در نظر بگیرید. مطابق شکل در این سیستم، دو مایع با غلظت و دبی حجمی متفاوت وارد مخزن شده و پس از انجام فرآیندهای شیمیایی بر روی این دو مایع در داخل مخزن، یک مایع به عنوان محصول از این سیستم خارج می‌شود. دبی و غلظت مایع اول به ترتیب با $C_{b1}$ و $w_1(t)$، دبی و غلظت مایع دوم به ترتیب با $C_{b2}$ و $w_2(t)$ و دبی و غلظت مایع خروجی (محصول) به ترتیب با $C_b(t)$ و $w_o(t)$ نمایش داده شده است. سطح مایع داخل تانک نیز با متغیر $h$ نمایش داده شده است.

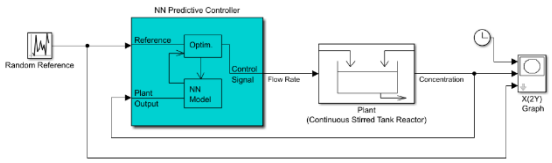



شکل ۲۰. شماتیکی از یک سیستم CSTR

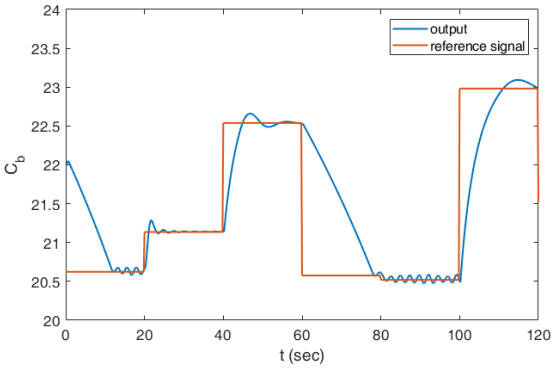
مدل دینامیکی این سیستم به صورت زیر است [۸۴]:

$$\begin{cases} \dfrac{dh(t)}{dt} = w_1(t)+w_2(t)-0.2\sqrt{h(t)} \\ \dfrac{dC_b(t)}{dt} = (C_{b1}-C_b(t))\dfrac{w_1(t)}{h(t)} \\ \qquad\qquad +(C_{b2}-C_b(t))\dfrac{w_2(t)}{h(t)} \\ \qquad\qquad -\dfrac{k_1 C_b(t)}{(1+k_2 C_b(t))^2} \end{cases} \qquad (۵۶)$$

هدف، کنترل غلظت مایع خروجی با استفاده از دبی مایع اول است (تنظیم ورودی $w_1(t)$ به‌گونه‌ای که $C_b$ برابر سیگنال مطلوب باشد). جهت سادگی، فرض می‌شود که $C_{b1}=24.9$ و $C_{b2}=0.1$ بوده و ثابت‌های $k_1$ و $k_2$ هر دو برابر یک هستند. همچنین، فرض می‌شود که دبی مایع دوم ثابت و برابر $w_1=0.1$ است. برای تحقق هدف کنترلی مذکور، از یک کنترل‌کننده پیش‌بین مدل غیرخطی استفاده شده است. شکل ۲۱ نمایی از بلوک دیاگرام شبیه‌سازی این سیستم در نرم‌افزار MATLAB را نشان می‌دهد. برای بخش مدل پیش‌بین این کنترل‌کننده، از یک شبکه عصبی برای پیش‌بینی خروجی‌های آینده (مدل پیش‌بین) استفاده شده است. این شبکه عصبی متشکل از دو لایه بوده که لایه پنهان آن دارای ۷ نورون است. جهت جمع‌آوری داده‌های آموزش و نهایتاً آموزش این شبکه در حالت حلقه باز، یک سیگنال PRBS به ورودی $w_1$ اعمال کرده و سیگنال خروجی متناظر ثبت شده است. سپس با استفاده از داده‌های آموزش جمع‌آوری شده و روش یادگیری لونبرگ – مارکوارت، شبکه عصبی طراحی شده آموزش داده شده است. برای بخش بهینه‌سازی کنترل‌کننده، از تابع هزینه تعریف شده در (۵۵) استفاده شده است. افق پیش‌بینی برابر $N_2=7$، افق کنترلی برابر $N_u=2$ و $\lambda=0.05$ انتخاب شده است. در شکل ۲۲، خروجی سیستم حلقه بسته تحت کنترل‌کننده پیش‌بین غیرخطی طراحی شده به همراه سیگنال مرجع نشان داده شده است. با توجه به شکل، مشاهده می‌شود که کنترل‌کننده به خوبی توانسته است سیگنال مرجع را دنبال نماید.




شکل ۲۱. نمایی از بلوک دیاگرام شبیه‌سازی سیستم CSTR با کنترل‌پیش‌بین مدل غیرخطی در نرم‌افزار MATLAB




شکل ۲۲. خروجی سیستم حلقه بسته به همراه سیگنال مرجع (مثال ۵)

## ۵-۲ کنترل‌کننده NARMA-L2

در مراجع مختلف، شبکه عصبی شرح داده شده در این بخش، با دو نام مختلف کنترل‌کننده NARMA-L2 و خطی‌سازی با فیدبک[1] شناخته می‌شود. همانگونه در ادامه با جزئیات بیشتر شرح داده خواهد شد، زمانی که مدل سیستم دارای فرم خاص NARMA باشد، از این ساختار تحت عنوان خطی‌سازی با فیدبک و زمانی که مدل سیستم قابل تخمین به فرم NARMA باشد، از آن تحت عنوان NARMA-L2 یاد می‌شود. ایده اصلی در این نوع کنترل‌کننده، تبدیل غیرخطی از دینامیک سیستم به یک دینامیک خطی با حذف ترم‌های غیرخطی است.

### ۵-۲-۱ تعریف مدل NARMA-L2

یک ساختار مدل استاندارد مورد استفاده در سیستم‌های غیرخطی زمان گسسته، مدل NARMA است که مطابق رابطه زیر تعریف می‌شود:

$$y[k+p]=f\,(y[k],y[k-1],\ldots,\ldots,y[k-n+1],u[k],\ldots,\ldots,u[k-n+1]) \quad (۵۷)$$

که $u[k]$ ورودی سیستم و $y[k]$ خروجی سیستم است. برای تعیین مدل سیستم (مدل‌سازی آزمایشی)، می‌توان از یک شبکه عصبی برای تخمین تابع غیرخطی $f$ در رابطه (۵۷) استفاده نمود که مشابه تعیین مدل پیش‌بین در معماری کنترلی تشریح شده قبلی (کنترل پیش‌بین مدل غیرخطی) است. اگر هدف این باشد که خروجی سیستم یک سیگنال مرجع را دنبال نماید، یعنی $y[k+p]=y_{ref}[k+p]$، به یک کنترل‌کننده غیرخطی با ساختار کلی زیر نیاز داریم:

$$u[k]=g\,(y[k],\ldots,y[k-n+1],y_{ref}[k+p],u[k],\ldots,\ldots,u[k-m+1]) \quad (۵۸)$$

مشکل استفاده از این کنترل‌کننده غیرخطی این است که اگر بخواهیم یک شبکه عصبی را برای ایجاد تابع $g$ به‌گونه‌ای که خطای مرجع میانگین (MSE) کمینه گردد، آموزش دهیم، باید از پس‌انتشار دینامیکی استفاده کنیم [۸۶-۸۵] که این کار بسیار کُند قابل انجام است. یک راه حل برای این مشکل، تخمین مدل‌ها برای نمایش سیستم است [۸۷]. کنترل‌کننده‌ای که در این بخش مورد استفاده قرار می‌گیرد، بر مبنای مدل تخمینی NARMA-L2 عمل می‌کند. این مدل به صورت زیر است:

$$y[k+d]=f\,(y[k],y[k-1],\ldots,\ldots,y[k-n+1],u[k-1],\ldots,\ldots,u[k-m+1])+g\,(y[k],\ldots,y[k-n+1],u[k-1],\ldots,u[k-m+1])u[k] \quad (۵۹)$$

این مدل در فرم هم‌نشین[2] بوده و سیگنال کنترلی $u[k]$ فاقد ترم‌های غیرخطی است. مزیت این فرم در این است که این فرم برای ورودی‌هایی که خروجی آن‌ها دارای سیگنال مرجع است، کارایی خوبی دارد. کنترل‌کننده حاصل دارای ساختار زیر خواهد بود:

[1] Feedback linearization

[2] Compation

$$z[k]=\frac{y_{ref}[k+p]-f(y[k],...,u[k-n+1])}{g(y[k],...,u[k-n+1])}$$

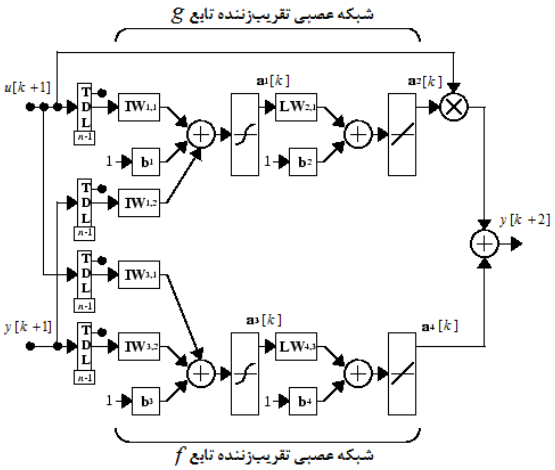
(۶۰)

استفاده از این معادله به صورت مستقیم دارای مشکلاتی است. چراکه شما باید ورودی‌های کنترلی $u[k]$ را برمبنای خروجی در همان زمان تعیین نمایید. برای رفع این مشکل، می‌توان از ساختار زیر استفاده نمود:

$$\begin{aligned}y(k+p)=f(&y[k],y[k-1],...,y[k-n+1],\\&u(k),u[k-1],...,u[k-n+1])\\+g(&y[k],...,y[k-n+1],\\&u[k],u[k-1],...,u[k-n+1])u[k+1]\end{aligned}$$

(۶۱)

در این ساختار، $p\geq 2$ در نظر گرفته می‌شود. شکل۲۳ ساختار شبکه عصبی مورداستفاده در این نوع کنترل‌کننده را نمایش می‌دهد.




شکل۲۳. استفاده از شبکه عصبی برای تقریب توابع f و g در ساختار NARMA-L2

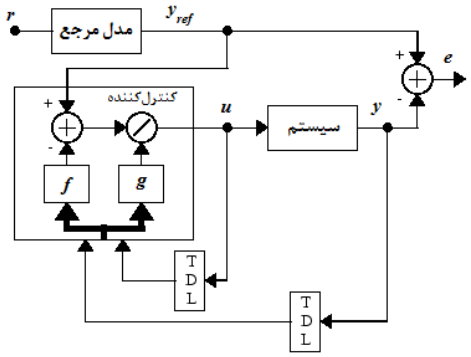
## ۲-۲-۵ کنترل‌کننده NARMA-L2

با استفاده از NARMA-L2 می‌توان کنترل‌کننده زیر را طراحی نمود:

$$u[k+1]=\frac{y_{ref}[k+p]-f(y[k],...,u[k-n+1])}{g(y[k],...,u[k-n+1])}$$

(۶۲)

که $p\geq 2$ تعریف شده است. در شکل۲۴ بلوک دیاگرام مربوط به کنترل‌کننده NARMA-L2 نشان داده شده است. همچنین در شکل۲۵، ساختار شبکه عصبی مورداستفاده در این نوع کنترل‌کننده نمایش داده شده است.



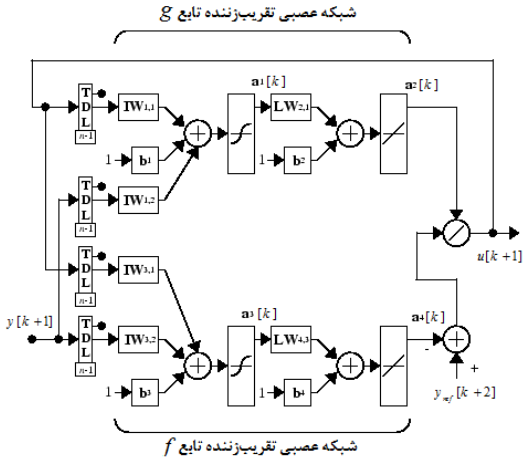

شکل۲۴. بلوک دیاگرام کنترل‌کننده NARMA-L2

**مثال۶)** سیستم شکل۲۶ را در نظر بگیرید. هدف، کنترل موقعیت آهنربای معلق در بالای آهنربای الکتریکی است. آهنربا فقط می‌تواند به صورت عمودی حرکت کند.

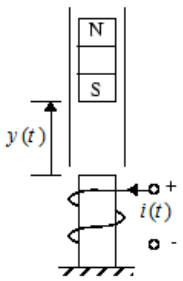
معادله دینامیکی حاکم بر این سیستم به صورت زیر است [۷۹]:

$$\frac{d^2y(t)}{dt^2}=-g+\frac{\alpha i^2(t)}{My(t)}-\frac{\beta}{M}\frac{dy(t)}{dt}$$

(۶۳)




شکل۲۵. ساختار شبکه عصبی مورداستفاده در کنترل‌کننده NARMA-L2

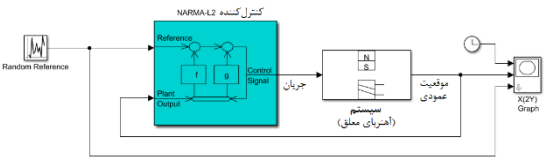



شکل۲۶. نمایی از سیستم آهنربای معلق

که $y(t)$ فاصله آهنربا از آهنربای الکتریکی، $i(t)$ شدت جریان آهنربای الکتریکی، $M$ جرم آهنربا و $g$ ثابت گرانش است. پارامتر $\beta$ ثابت اصطکاک چسبندگی است که با توجه به ماده‌ای که آهنربا در آن جابجا می‌شود، تعیین می‌گردد. ثابت $\alpha$ نیز با توجه به تعداد دورهای سیم حول آهنربای الکتریکی و قدرت آهنربا تعریف می‌شود. در این مثال، مقادیر زیر برای پارامترهای مدل و شرایط اولیه فرض شده است:

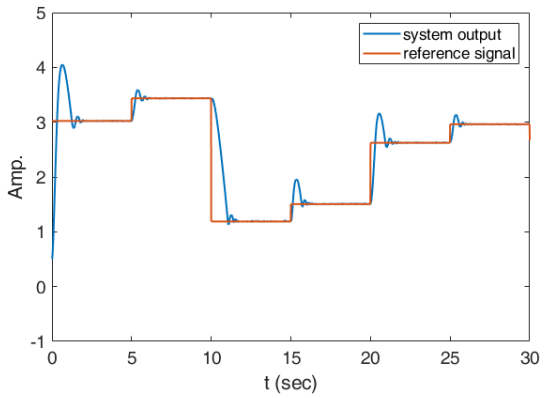
$$\beta = 12 \ , \ M = 3 \ , \ g = 9.8 \ , \ \alpha = 15$$
$$y(0) = 0.5 \ , \ \dot{y}(0) = 0$$

هدف، کنترل موقعیت آهنربای معلق در بالای آهنربای الکتریکی است. برای این منظور، از یک کنترل‌کننده NARMA-L2 استفاده شده است. شبکه‌های عصبی استفاده شده برای تقریب توابع $f$ و $g$ در ساختار NARMA-L2 از نوع MLP دو لایه بوده که لایه پنهان آن‌ها از ۹ نرون تشکیل شده‌اند. بلوک دیاگرام شبیه‌سازی این سیستم در نرم‌افزار MATLAB به همراه کنترل‌کننده طراحی شده در شکل ۲۷ آورده شده است.




شکل ۲۷. بلوک دیاگرام سیستم حلقه بسته آهنربای معلق به همراه کنترل‌کننده NARMA-L2

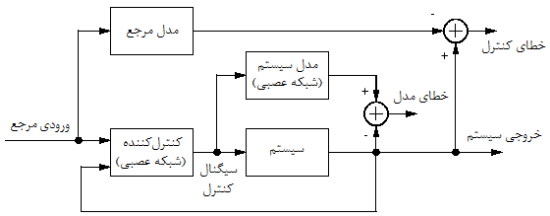
در شکل ۲۸، منحنی خروجی سیستم حلقه بسته و سیگنال مرجع تحت شبیه‌سازی به مدت ۳۰ ثانیه آورده شده است. همان‌گونه در شکل نیز مشاهده می‌شود، هدف کنترلی به خوبی محقق شده است. البته با تنظیم پارامترهای کنترل‌کننده می‌توان میزان فراجهش را نیز کم نمود.




شکل ۲۸. منحنی‌های خروجی سیستم حلقه بسته و سیگنال مرجع برای سیستم آهنربای معلق (مثال ۶)

## ۳-۵ کنترل‌کننده مدل مرجع

مطابق شکل ۲۹، معماری شبکه عصبی کنترل‌کننده مدل مرجع از دو شبکه عصبی تشکیل شده است: یک شبکه عصبی به عنوان کنترل‌کننده بوده و یک شبکه عصبی نیز برای توصیف مدل سیستم.




شکل ۲۹. معماری کنترل‌کننده مدل مرجع با استفاده از شبکه‌های عصبی

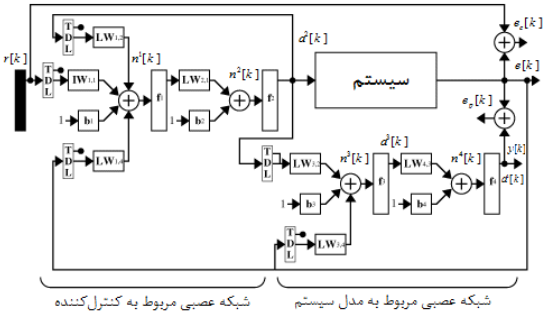
در این معماری کنترلی، هر شبکه عصبی دارای دو لایه است. تعداد نرون‌های موجود در لایه پنهان قابل تعیین توسط کاربر است که می‌توان از روش‌های مطرح شده در بخش ۴ مقاله (بخش مدل‌سازی) برای تعیین تعداد مناسب نرون‌های لایه مخفی استفاده نمود. ورودی‌های کنترل‌کننده به سه دسته تقسیم می‌شوند:

- ورودی‌های تأخیری (لحظات قبل) سیگنال مرجع
- خروجی‌های تأخیری (لحظات قبل) کنترل‌کننده
- خروجی‌های تأخیری (لحظات قبل) سیستم

برای هر یک از این ورودی‌ها، تعداد تأخیرها قابل‌تعیین است. عموماً تعداد تأخیرها با مرتبه سیستم افزایش می‌یابد. ورودی‌های شبکه عصبی مربوط به مدل سیستم به دو دسته تقسیم می‌شود:

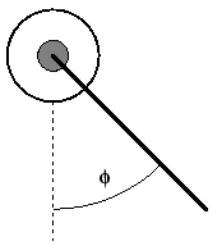
- خروجی‌های تأخیری کنترل‌کننده
- خروجی‌های تأخیری سیستم

شکل ۳۰ معماری شبکه عصبی مورداستفاده در کنترل‌کننده مدل مرجع را نمایش می‌دهد.




شکل ۳۰. معماری شبکه عصبی مورداستفاده در کنترل‌کننده مدل مرجع

**مثال ۷)** ربات تک مفصله نشان داده شده در شکل ۳۱ را در نظر بگیرید. مطابق شکل، این ربات فقط دارای یک بازو و یک مفصل (نقطه اتصال) است.




شکل ۳۱. سیستم ربات ساده

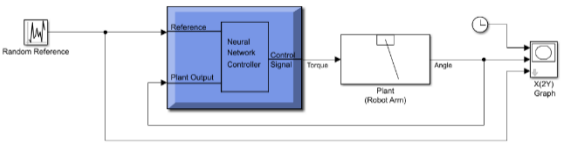
معادله دینامیکی حرکت این بازو به صورت رابطه ریاضی زیر قابل بیان است [۷۹]:

$$\frac{d^2\varphi(t)}{dt^2} = -10\sin\varphi(t) - \frac{2d\,\varphi(t)}{dt} + u(t) \qquad (۶۴)$$

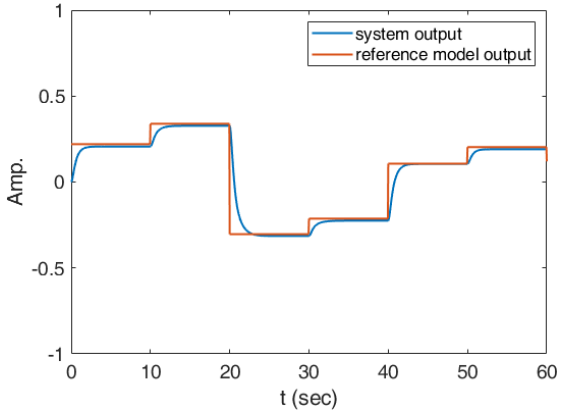
که $\varphi$ زاویه بازو بوده و $u$ نیز گشتاور بازو است که توسط موتور $DC$ (به عنوان عملگر) تأمین می‌شود. هدف، طراحی یک کنترل‌کننده به‌گونه‌ای است که زاویه بازو در هر لحظه خروجی مدل مرجع زیر را دنبال نماید:

$$\frac{d^2 y_{ref}(t)}{dt^2} = -9y_{ref}(t) - 6\frac{dy_{ref}(t)}{dt} + 9r(t) \quad (۶۵)$$

به‌طوری‌که $y_{ref}$ خروجی مدل مرجع و $r$ سیگنال مرجع است. برای این منظور، از معماری کنترل مدل مرجع با شبکه‌های عصبی شکل ۲۹ استفاده می‌کنیم. ورودی‌های کنترل‌کننده شامل دو ورودی تأخیری مرجع، دو خروجی تأخیری سیستم و یک خروجی تأخیری کنترل‌کننده است. همچنین، زمان نمونه‌برداری برابر ۰.۰۵ ثانیه در نظر گرفته شده است. در شکل ۳۲، بلوک دیاگرام شبیه‌سازی این سیستم در نرم‌افزار MATLAB به همراه کنترل‌کننده مدل مرجع طراحی شده برای آن نمایش داده شده است. در شکل ۳۳، منحنی خروجی سیستم حلقه بسته و سیگنال مرجع تحت شبیه‌سازی به مدت ۶۰ ثانیه آورده شده است. همان‌گونه در شکل نیز مشاهده می‌شود، هدف کنترلی به خوبی محقق شده است.




شکل ۳۲. بلوک دیاگرام سیستم حلقه بسته ربات تک مفصله به همراه کنترل‌کننده مدل مرجع با شبکه‌های عصبی




شکل ۳۳. منحنی‌های خروجی سیستم حلقه بسته و سیگنال مرجع برای ربات تک مفصله (مثال ۷)

## ۶- جمع‌بندی و پیشنهاد برای کارهای آتی

در این مقاله، به تحلیل سیستم‌های هوشمند و به طور خاص شبکه‌های عصبی از دیدگاه تئوری کلاسیک پرداخته و کاربرد شبکه‌های عصبی در مدل‌سازی و کنترل سیستم‌های پیچیده پرداخته شد. در این راستا، ابتدا نرون به عنوان عنصر اصلی شبکه‌های عصبی معرفی و مدل آن ارائه گردید. سپس انواع شبکه‌های عصبی از قبیل شبکه‌های عصبی پیشرو (MLP)، شبکه‌های عصبی شعاعی (RBF) و شبکه‌های عصبی بازگشتی (RNN) تشریح گردید. در ادامه، در مورد تعداد لایه‌ها و تعداد نرون‌های مناسب شبکه‌های عصبی برای کاربردهای مختلف و به‌خصوص جهت تقریب توابع غیرخطی و مدل‌سازی آزمایشی سیستم‌های پیچیده بحث شد. سپس الگوریتم‌های یادگیری شبکه‌های عصبی از دیدگاه تئوری کلاسیک و بحث بهینه‌سازی عددی موردمطالعه و بررسی قرار گرفت و چندین مثال از مدل‌سازی سیستم‌های غیرخطی (استاتیکی و دینامیکی) نیز ارائه گردید. پس از بحث مدل‌سازی، به بحث کاربرد شبکه‌های عصبی در کنترل سیستم‌های پیچیده پرداخته شد و سه معماری کنترلی مهم کنترل پیش‌بین مدل با مدل پیش‌بین از نوع شبکه‌های عصبی، کنترل NARMA-L2 و کنترل مدل مرجع ارائه گردید و نهایتاً با ارائه شبیه‌سازی‌هایی، عملکرد و کارایی این معماری‌های کنترلی بررسی شد. به‌عنوان ادامه این کار، در مقاله آتی، به مروری بر انواع تکنیک‌های مبتنی بر مدل و تکنیک‌های مبتنی بر داده جهت مدل‌سازی و کنترل سیستم‌ها پرداخته شده و مورد مقایسه قرار می‌گیرند و جایگاه یادگیری در برابر بهینه‌سازی مقایسه شده و به انواع تکنیک‌های یادگیری، یادگیری باناظر، بدون ناظر، یادگیری تقویتی، یادگیری عمیق از منظرگاه تئوری سیستم‌ها پرداخته خواهد شد. همچنین، پیرامون سرعت ارائه داده‌ها، به شکل دنباله داده‌ها یا جویبار داده در هر دو بُعد زمانی و خاص مطالعه خواهد شد. در برخی کاربردها، توزیع احتمال نمونه‌های جویبار داده تغییر می‌کند (رانش یا گذار مفهوم). همچنین، به تأثیرات سرعت این تغییرات در جویبار داده‌ها، تنوع داده‌ها و میزان بزرگی داده‌ها در سیستم یادگیری موردمطالعه قرار می‌گیرد. سیستم یادگیری که به‌عنوان مدل موردمطالعه قرار می‌گیرد، دارای رابطه جایگزینی دقت در برابر حجم حافظه می‌باشد. ویژگی‌های اصلی این سیستم یادگیری را نقد خواهیم نمود و به جایگاه یادگیری ماشین در مقابل یادگیری آماری و علم داده و ارتباط با هوش مصنوعی و علوم سایبرنتیک و نهایتاً محاسبات شناختی و کنترل شناختی خواهیم پرداخت. کنترل شناختی یا عملکردهای اجرایی فرآیندهای شناختی، به بررسی فرایندهای شناختی که در یک محیط دائماً در حال تغییر هستند می‌پردازد. این مطالعه، با پوشش دادن تحقیقات، مدل‌ها و نظریه‌های کلاسیک شناختی و همچنین عصب‌شناسی، نگاهی عمیق به این فرآیندها از منظرگاه تئوری سیستم‌ها ارائه می‌نماید. شناسایی سیستم‌ها، رباتیک شناختی، طراحی و ساخت واسط‌های ماشین و مغز، توسعه سیستم‌های شناختی مصنوعی یادگیر، روش‌های علوم اعصاب محاسباتی، تجزیه‌وتحلیل ارتباط ویژگی‌های ساختاری و عملکردی مغز و تغییرات دینامیکی آن‌ها در مقیاس‌های زمانی مختلف، هسته اصلی ملاحظات مطالعه آتی خواهند بود.

## مراجع

[۱]. محمدباقر منهاج، هدی نخبه الفقهایی، "تحلیل و کنترل سیستم های دینامیکی و چند متغیره خطی"، جلد اول، انتشارات دانشگاه صنعتی امیرکبیر، ۱۴۰۱.

[۲]. محمدباقر منهاج، هدی نخبه الفقهایی، "تحلیل و کنترل سیستم های دینامیکی و چند متغیره خطی"، جلد دوم، پیش‌انتشار.

[۳]. محمدباقرمنهاج، محمد لطفی، "سیستم‌های دینامیکی هایبرید، جلد اول، پیش‌انتشار.

[۴]. محمدباقر منهاج، "سیستم‌های کنترل تطبیقی"، انتشارات نهر دانش، ۱۳۹۵

[۵]. محمدباقرمنهاج، "مبانی شبکه‌های عصبی"، انتشارات دانشگاه صنعتی امیرکبیر، ۱۳۹۷.

[۶]. محمدباقر منهاج، "محاسبات فازی"، انتشارات دانش نگار، ۱۳۹۴.

[۷].محمدباقر منهاج، یاسر شکری کلاندرق، "کنترل فازی"، انتشارات نهر دانش، ۱۳۹۴.

[8]. F. P. Brooks, "What's real about virtual reality?," in IEEE Computer Graphics and Applications, vol. 19, no. 6, pp. 16-27, Nov.-Dec. 1999, doi: 10.1109/38.799723.

[9]. Xue, Dingyü, YangQuan Chen, and Derek P. Atherton. Linear feedback control: analysis and design with MATLAB. Society for Industrial and Applied Mathematics, 2007.

[10]. Mayr, Otto. "The origins of feedback control." Scientific American 223.4, 110-119, 1970.

[11]. Minorsky, Nicolas. "Directional stability of automatically steered bodies." Journal of the American Society for Naval Engineers 34.2, 280-309, 1922.

[12]. Ziegler, John G., and Nathaniel B. Nichols. "Optimum settings for automatic controllers." Transactions of the American society of mechanical engineers 64.8, 759-765, 1942.

[13]. Nyquist, Harry. "Regeneration theory." Bell system technical journal 11.1, 126-147, 1932.

[14]. Bode, Hendrik W. "Network analysis and feedback amplifier design.", Princeton, NJ:Van Nostrand, 1945.

[15]. James, Hubert Maxwell, et al., eds. Theory of servomechanisms. Vol. 25. New York: McGraw-Hill, 1947.

[16]. Evans, Walter R. "Graphical analysis of control systems." Transactions of the American Institute of Electrical Engineers 67.1, 547-551, 1948.

[17]. Pontryagin, L. S., Boltyanskii, V. G., Gamkrelidze, R. V., & Mishchenko, E. F., "The mathematical theory of optimal processes", Wiley. New York, 1962.

[18]. Bellman, Richard. "Dynamic programming" Science 153.3731, 34-37, 1966.

[19]. Kalman, Rudolf E. "On the general theory of control systems." Proceedings First International Conference on Automatic Control, Moscow, USSR, 1960.

[20]. Kalman, Rudolf Emil. "Mathematical description of linear dynamical systems." Journal of the Society for Industrial and Applied Mathematics, Series A: Control 1.2, 152-192, 1963.

[21]. Kalman, Rudolf Emil. "When is a linear control system optimal?.", Transactions of ASME Journal of Basic Engineering Series D, 51-60, 1964.

[22]. Doyle, John, and Gunter Stein. "Robustness with observers." IEEE transactions on automatic control 24.4, 607-611, 1979.

[23]. Bryson, Arthur Earl. Applied optimal control: optimization, estimation and control. Routledge, 2018.

[24]. Berkovitz, Leonard David. Optimal control theory. Vol. 12. Springer Science & Business Media, 2013.

[25]. Hocking, Leslie M. Optimal control: an introduction to the theory with applications. Oxford University Press, 1991.

[26]. Krstic, Miroslav, and Hua Deng. Stabilization of nonlinear uncertain systems. London: Springer, 1998.

[27]. Gajic, Zoran. Optimal control of singularly perturbed linear systems and applications. CRC Press, 2001.

[28]. Trentelman, Harry L., Anton A. Stoorvogel, and Malo Hautus. Control theory for linear systems. Springer Science & Business Media, 2001.

[29]. Naidu, Desineni Subbaram. "optimal control systems.", CRC press, 2002.

[30]. Chen, Wen-Hua, Donald J. Ballance, and Peter J. Gawthrop. "Optimal control of nonlinear systems: a predictive control approach." Automatica 39.4 (2003): 633-641.

[31]. Kirk, Donald E. Optimal control theory: an introduction. Courier Corporation, 2004.

[32]. Athans, Michael, and Peter L. Falb. Optimal control: an introduction to the theory and its applications. Dover publication, 2006.

[33]. Lewis, Frank L., Lihua Xie, and Dan Popa. Optimal and robust estimation: with an introduction to stochastic control theory. CRC press, 2007.

[34]. Sinha, Alok. Linear systems: optimal and robust control. CRC press, 2007.

[35]. Bhattacharyya, Shankar P., Aniruddha Datta, and Lee H. Keel. Linear control theory: structure, robustness, and optimization. CRC press, 2009.

[36]. Speyer, Jason L., and David H. Jacobson. Primer on optimal control theory. Society for Industrial and Applied Mathematics, 2010.

[37]. Berkovitz, Leonard David, and Negash G. Medhin. Nonlinear optimal control theory. CRC press, 2012.

[38]. Lewis, Frank L., Draguna Vrabie, and Vassilis L. Syrmos. Optimal control. John Wiley & Sons, 2012.

[39]. Sussmann, Hector J. Nonlinear controllability and optimal control. Routledge, 2017.

[40]. Kolosov, Gennadii E. Optimal Design of Control Systems: Stochastic and Deterministic Problems (Pure and Applied Mathematics: A Series of Monographs and Textbooks/221), 3rd Edition, CRC Press, 1999.

[41]. Fortuna, Luigi, Mattia Frasca, and Arturo Buscarino. Optimal and Robust Control: Advanced Topics with MATLAB®. CRC press, 2021.

[42]. Jadamba, Baasansuren, et al., eds. Deterministic and Stochastic Optimal Control and Inverse Problems. CRC Press, 2021.

[43]. Thyagarajan, T., and D. Kalpana. Linear and Non-Linear System Theory. CRC Press, 2020.

[44]. Hernández-Lerma, Onésimo, et al. An introduction to optimal control theory: The dynamic programming approach. Springer, 2023.
[45]. G. Johnson, "A deterministic theory of estimation and control," in IEEE Transactions on Automatic Control, vol. 14, no. 4, pp. 380-384, 1969, doi: 10.1109/TAC.1969.1099191.
[46]. Isermann, Rolf, and Rolf Isermann. "Stochastic Control Systems (Introduction)." Digital Control Systems: Volume 2: Stochastic Control, Multivariable Control, Adaptive Control, Applications (1991): 3-9.
[47]. Chen, Goong, Guanrong Chen, and Shih-Hsun Hsu. Linear stochastic control systems. Vol. 3. CRC press, 1995.
[48]. Krstic, Miroslav, and Hua Deng. Stabilization of nonlinear uncertain systems. London: Springer, 1998.
[49]. Deng, Hua, Miroslav Krstic, and Ruth J. Williams. "Stabilization of stochastic nonlinear systems driven by noise of unknown covariance." IEEE Transactions on automatic control 46.8 (2001): 1237-1253.
[50]. Åström, Karl J. Introduction to stochastic control theory. Dover Publications, 2006.
[51]. Kouvaritakis, Basil, and Mark Cannon. "Model predictive control." Switzerland: Springer International Publishing 38 (2016).
[52]. Zhang, Weihai, Lihua Xie, and Bor-Sen Chen. Stochastic H2/H∞ control: A Nash game approach. CRC Press, 2017.
[53]. Pinsky, Mark A. Stochastic analysis and applications. CRC Press, 2019.
[54]. Buckdahn, Rainer, Hans J. Engelbert, and Marc Yor, eds. Stochastic Processes and Related Topics: Proceedings of the 12th Winter School, Siegmundsburg (Germany), February 27-March 4, 2000. Vol. 12. CRC Press, 2002.
[55]. Wei, Guoliang, Zidong Wang, and Wei Qian. Nonlinear Stochastic Control and Filtering with Engineering-oriented Complexities. CRC Press, 2016.
[56]. Kulkarni, Vidyadhar G. Modeling and analysis of stochastic systems. Crc Press, 2016.
[57]. Jadamba, Baasansuren, et al., eds. Deterministic and Stochastic Optimal Control and Inverse Problems. CRC Press, 2021.
[58]. Zames, George. "Feedback and optimal sensitivity: Model reference transformations, multiplicative seminorms, and approximate inverses." IEEE Transactions on automatic control 26.2 (1981): 301-320.
[59]. Bhattacharyya, Shankar P., and Lee H. Keel, eds. Control of uncertain dynamic systems. Vol. 230. Boca Raton, FL: CRC Press, 1991.
[60]. Wang, Youyi, Lihua Xie, and Carlos E. De Souza. "Robust control of a class of uncertain nonlinear systems." Systems & control letters 19.2 (1992): 139-149.
[61]. C Edwards and S Spurgeon, " Sliding Mode Control Theory And Applications", CRC Press, 1998.
[62]. Bechlioulis, Charalampos P., and George A. Rovithakis. "Robust adaptive control of feedback linearizable MIMO nonlinear systems with prescribed performance." IEEE Transactions on Automatic Control 53.9 (2008): 2090-2099.
[63]. Safonov, Michael George. "Robust control, stability margin." Encyclopedia of Optimization 5 (1999): 44-49.
[64]. Wen, Changyun, et al. "Robust adaptive control of uncertain nonlinear systems in the presence of input saturation and external disturbance." IEEE Transactions on Automatic Control 56.7 (2011): 1672-1678.
[65]. Duan, Guang-Ren, and Hai-Hua Yu. LMIs in control systems: analysis, design and applications. CRC press, 2013.
[66]. Liu, Tengfei, Zhong-Ping Jiang, and David J. Hill. Nonlinear control of dynamic networks. CRC Press, 2018.
[67]. Raol, Jitendra R., and Ramakalyan Ayyagari. "Control systems: classical, modern, and AI-based approaches". CRC Press, 2019.
[68]. Tsui, Chia-Chi. Robust control system design: advanced state space techniques. CRC Press, 2022.
[69]. Marino, R. I. C. C. A. R. D. O., and P. A. T. R. I. Z. I. O. Tomei. "Global adaptive output-feedback control of nonlinear systems." Proceedings of the 30th IEEE Conference on Decision and Control. IEEE, 1991.
[70]. Khalil, Hassan K. "Adaptive output feedback control of nonlinear systems represented by input-output models." IEEE transactions on Automatic Control 41.2 (1996): 177-188.
[71]. Astolfi, Alessandro, Dimitrios Karagiannis, and Romeo Ortega. Nonlinear and adaptive control with applications. Vol. 187. London: Springer, 2008.
[72]. Yang, Guang-Hong, and Dan Ye. Reliable control and filtering of linear systems with adaptive mechanisms. CRC Press, 2010.
[73]. Chen, Mou, Shuzhi Sam Ge, and Beibei Ren. "Adaptive tracking control of uncertain MIMO nonlinear systems with input constraints." Automatica 47.3 (2011): 452-465.
[74]. Bellman, Richard, and Robert Kalaba, eds. Classic Papers in Control Theory. Courier Dover Publications, 2017.
[75]. Chalam, V. V. Adaptive control systems: Techniques and applications. Marcel Dekker, Inc., 2017.
[76]. Song, Yong-Duan. Control of nonlinear systems via PI, PD and PID: Stability and performance. CRC Press, 2018.
[77]. https://faradars.org/
[۷۸]. محمد فتحی، فرینا زمانی اسکویی، تئوری و کاربرد شبکه‌های عصبی در MATLAB، انتشارات کانون نشر علوم، چاپ اول، ۱۳۹۷.
[۷۹]. سید مصطفی کیا، شبکه‌های عصبی در MATLAB، انتشارات نشر دانشگاهی کیا، چاپ پنجم، ۱۳۹۵.

[80].Feldman, Jerome A., and Dana H. Ballard. "Connectionist models and their properties." Cognitive science 6.3 (1982): 205-254.

[81]. J. Heaton, "Introduction to Neural Networks for Java", 2nd Edition, Heaton Research, Inc., 2008.

[82]. J. Nocedal and S. J. Wright, "Numerical Optimization", 2Nd Edition, Springer-Verlag New York, 2006.

[83]. A. Papoulis, "Probability & Statistics", Prentice Hall, 1990.

[84]. Sistu, Phani B., and B. Wayne Bequette. "Nonlinear predictive control of uncertain processes: Application to a CSTR." AIChE Journal 37.11 (1991): 1711-1723.

[85]. Narendra, Kumpati S. and Kannan Parthasarathy, “Learning Automata Approach to Hierarchical Multiobjective Analysis,” IEEE Transactions on Systems, Man and Cybernetics, Vol. 20, No. 1, January/February 1991, pp. 263–272.

[86]. Hagan, M.T., O. De Jesus, and R. Schultz, “Training Recurrent Networks for Filtering and Control,” Chapter 12 in Recurrent Neural Networks: Design and Applications, L. Medsker and L.C. Jain, Eds., CRC Press, pp. 311–340.

[87]. Narendra, K.S., and S. Mukhopadhyay, “Adaptive Control Using Neural Networks and Approximate Models,” IEEE Transactions on Neural Networks, Vol. 8, 1997, pp. 475–485.