الگوریتمی براساس اتوماتان‌های یادگیر برای تعیین حداقل تعداد واحدهای مخفی در یک شبکه عصبی سه لایه

حمید بیگی
دانشجوی دکترا

محمّدرضا میبدی
دانشیار

دانشکده مهندسی کامپیوتر ، دانشگاه صنعتی امیرکبیر

چکیده

یکی از مسائل حل نشده در شبکه های عصبی سه لایه انتخاب ساختار بهینه (ساختار با حداقل تعداد واحدهای لایه میانی) برای شبکه می باشد. در شبکه های عصبی سه لایه، در ابتدای آموزش، یك ساختار مشخص برای شبکه انتخاب و سپس شبکه آموزش داده می شود. کارایی الگوریتم آموزش برای این شبکه ها، به میزان زیادی به ساختار انتخاب شده دارد. برای مشخص نمودن ساختار بهینه باید یك جستجوی کامل در فضای ساختارها انجام گیرد که این خود دارای هزینه بالایی است. مسئله پیدا کردن ساختار بهینه برای شبکه های عصبی، در گروه مسائل NP-Hard قرار دارد و به همین جهت برای کاهش زمان تعیین ساختار مناسب برای شبکه ، روش های تقریبی از جمله الگوریتم های سازنده، الگوریتم های هرس، الگوریتم های ترکیبی و الگوریتم های تکاملی پیشنهاد شده اند که ساختار نزدیك به بهینه را تولید می کنند. در اغلب این روش ها، از الگوریتم های کوهنوردی به عنوان ابزار جستجو استفاده می شود که مشکل گرفتاری در حداقل های محلی را دارند. در این مقاله الگوریتی به نام الگوریتم بقا، برای تعیین ساختار شبکه عصبی سه لایه معرفی شده است که از یك اتوماتان یادگیر (به عنوان یك ابزار جستجوی عمومی) و الگوریتم انتشار خطا به عقب استفاده می کند و در ضمن آموزش، ساختار نزدیك به بهینه را برای شبکه تعیین می نماید. استفاده از اتوماتان های یادگیر به عنوان یك ابزار جستجوی عمومی مشکل گرفتاری در حداقل های محلی را تا حدودی حل می نماید و احتمال رسیدن به بهترین ساختار را افزایش می دهد. در الگوریتم بقا، آموزش از یك شبکه بزرگ شروع شده و اتومان یادگیر با افزودن و کاستن واحدهای مخفی، حداقل تعداد واحدهای مورد نیاز لایه مخفی را تعیین می کند.

A Learning Automata Based Algorithm for Determination of Minimum Number of Hidden Units for Three Layers Neural Networks

H. Beigy
Ph.D. Student

M. R. Meybodi
Associate Professor

Computer Engineering Department,
Amirkabir University of Technology

Abstract

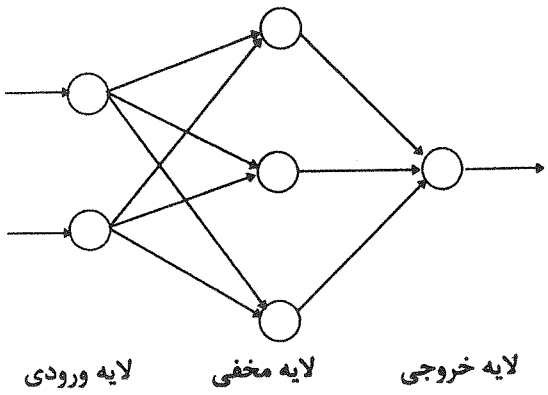
One of the unsolved problems in multi-layer neural networks is the problem of determination of optimal topology: a topology with minimum number of hidden units. There is no method to determine the optimal topology for multi-layer neural networks for a given problem. Usually the designer selects a topology for the network and then trains it. The selected topology remains fixed during the training period. Since the problem of determination of optimal topology for neural networks belongs to class of NP-Hard problems, most of the existing algorithms for determination of the topology are approximate algorithms. These algorithms could be classified into four main groups: pruning algorithms, constructive algorithms, hybrid algorithms, and evolutionary algorithms. These algorithms can produce near optimal solutions. Most of these algorithms use hill-climbing method and may stuck at local minima. In this paper a learing automata based algorithm called survival algorithm, for determination of the number of hidden units of three layers neural networks is proposed. The proposed algorithm uses learning automaton as a global search method in order to increase the probability of escaping from local minima and hence to increase the probability of obtaining the optimal topology. In survival algorithm, the training begins with a large network, and then by addıng and deleting hidden units the optimal topology is obtained.

کلمات کلیدی

ساختار شبکه عصبی، آموزش شبکه عصبی، اتوماتان یادگیر، الگوریتم انتشار خطا به عقب، شبکه های سه لایه.

مقدمه

شبکه های عصبی دارای کاربردهای زیادی می باشند [۲۹] [۴۱]. بیشتر این کاربردها از شبکه های عصبی چند لایه پرسپترونی[1] استفاده می کنند. شبکه های عصبی چند لایه پرسپترونی (که به اختصار شبکه های عصبی چند لایه نامیده می شوند)، شبکه های پیش خوری[2] هستند که در آنها یک یا بیشتر لایه مخفی بین لایه های ورودی و خروجی وجود دارد. در این شبکه ها معمولاً اتصال های بین لایه ها کامل در نظر گرفته می شود و ورودی های واحدهای هر لایه فقط به خروجی لایه قبل از خود متصل هستند. شکل ۱ یک شبکه عصبی ۳ ـ لایه پرسپترونی را که در این مقاله مدنظر می باشد، نشان می دهد.



شکل (۱) شبکه عصبی ۳ ـ لایه پرسپترونی.

شبکه عصبی چند لایه پرسپترونی توسط رزنبلت[3] پیشنهاد شد [۴۱] که در آن از تابع پله[4] برای تابع فعالیت[5] واحدها استفاده گردیده است. فقدان یک الگوریتم یادگیری مناسب یکی از محدودیت هایی است که شبکه های چند لایه پرسپترونی دارا هستند. یک روش معمول برای آموزش چنین شبکه هایی، استفاده از روش تکراری نزول در امتداد گرادیان می باشد که به دلیل عدم مشتق پذیری تابع پله این روش قابل استفاده نیست. استفاده از یک تابع غیرخطی مشتق پذیر برای تابع

فعالیت این مشکل را حل می کند. الگوریتم آموزش انتشار خطا[۶] به عقب که در شکل (۲) نشان داده شده است یک روش برای آموزش شبکه های عصبی چند لایه پرسپترونی می باشد. در این الگوریتم، یک الگو به شبکه داده می شود. مقایسه پاسخ شبکه با پاسخ مورد نظر برای این الگو، این قابلیت را می دهد که وزن ها به گونه ای تنظیم شوند تا شبکه در زمان های بعد، پاسخ دقیق تری را تولید نماید [۲۹].

```
procedure StandardBackPropagationAlgorithm
 input:
  Training Set (X, T) , where X is inputs & T is desired outputs
 output:
  Network Weight Vector: W

  Initialize the W to small random values
  repeat
    for all training pair (X, T) in training set do
      call FeedForward
      call ComputeGradiant
    end for
    call UpdateWeights
  until termination condition Satisfied
  return W
end procedure
```

شکل (۲) الگوریتم آموزش انتشار خطا به عقب.

الگوریتم آموزش انتشار خطا به عقب از روش جستجو در امتداد گرادیان برای حداقل نمودن تابع هزینه زیر (مجموع مربعات خطا) استفاده می کند.

$$E(n) = \frac{1}{2} \sum_{p=1}^{\#patterns} \sum_{j=1}^{\#outputs} [T_{p,j} - O_{p,j}]^2$$

که #patterns و #outputs به ترتیب تعداد الگوهای آموزشی و تعداد واحدهای لایه خروجی شبکه می باشند و $T_{p,j}$ و O_{pj} به ترتیب پاسخ مورد نظر و پاسخ حقیقی شبکه برای الگوی آموزشی p در واحد خروجی j را نشان می دهند. برای حداقل نمودن تابع هزینه، در قدم n ام وزن های شبکه به صورت زیر بهنگام می گردند.

$$W(n+1) = W(n) - \eta \nabla E(n) + \alpha [W(n) - W(n-1)]$$

که W بردار وزن، η نرخ یادگیری ، α ضریب گشتاور و $\nabla E(n)$ گرادیان خطا نسبت به وزن ها در قدم n ام می باشد.

کارایی الگوریتم آموزش انتشار خطا به عقب برای یک کاربرد، به میزان زیادی به ساختار شبکه عصبی (تعداد لایه های مخفی، تعداد واحدهای مخفی در هر لایه و اتصالات بین آنها) وابسته است. برای ارزیابی کارایی شبکه های عصبی معیارهای متعددی پیشنهاد شده است. پیچیدگی آموزش، قدرت حافظه و قدرت تعمیم شبکه از آن جمله اند. ابتدا معیارهای فوق به طور خلاصه تعریف و سپس رابطه ساختار شبکه با این معیارها بحث خواهد شد.

پیچیدگی آموزش[۸]

پیچیدگی آموزش شبکه در واقع همان پیچیدگی زمانی الگوریتم آموزش می باشد که بستگی به کاربرد مورد نظر، ساختار شبکه و الگوریتم آموزش دارد.

قدرت حافظه[۹]

توان شبکه در دادن پاسخ صحیح به نمونه های آموزشی را قدرت حافظه شبکه می نامند. هر چه این قابلیت بیشتر باشد، شبکه به تعداد بیشتری از نمونه های آموزشی پاسخ صحیح می دهد.

قدرت تعمیم[۱۰]

قدرت تعمیم شبکه درصد پاسخ های صحیح یا نزدیک به صحیح به ورودی هایی است که شبکه در حین آموزش ندیده است (الگوهای آزمایشی). هر چه تعداد پاسخ های مناسب شبکه به اینگونه ورودی ها بیشتر باشد، قدرت تعمیم شبکه بیشتر خواهد بود.

حال به بررسی رابطه معیارهای فوق و ساختار شبکه می پردازیم.

رابطه ساختار با پیچیدگی آموزش

الگوریتم های یادگیری شبکه های چند لایه عموماً از روش نزول در امتداد گرادیان استفاده می کنند که همگرایی آنها به شکل سطح خطا که خود بستگی به تعداد واحدهای مخفی دارد، وابسته است. تنها نتیجه ای که در ارتباط با رابطه شکل سطح خطا و ساختار شبکه عصبی گزارش شده است، مربوط به شبکه های عصبی سه لایه می باشد. طبق [۴۰] اگر تعداد واحدهای مخفی در یک شبکه عصبی سه لایه به اندازه یک واحد کمتر از داده های آموزشی انتخاب شود، سطح خطای این شبکه بدون پستی و بلندی (حداقل محلی) خواهد بود. در صورتی که

سطح خطا دارای پستی و بلندی باشد، ممکن است الگوریتم یادگیری در حداقل های محلی گرفتار شود و قادر به آموزش شبکه نباشد.

رابطه ساختار وقدرت تعمیم

فرایند یادگیری در شبکه های عصبی را می توان به عنوان یک عمل تقریب تابع در نظر گرفت که در آن شبکه یک نگاشت غیرخطی بین ورودی و خروجی ایجاد می نماید. با این تعبیر، قدرت تعمیم شبکه همان قابلیت درونیابی[11] روی الگوهای آموزشی است [۴۱]. خطای درونیابی برای الگوهای آموزشی یا الگوهای آزمایشی به تعداد واحدهای مخفی شبکه بستگی دارد. اگر تعداد واحدهای مخفی کم باشد، تابعی که توسط شبکه تقریب زده می شود هموار است و در نتیجه شبکه ویژگی های داده های آموزشی را بهتر یاد می گیرد و به همین دلیل قدرت تعمیم شبکه بالا خواهد بود. در صورتی که تعداد واحدهای مخفی زیاد باشد، تابع تقریب زده شده از تمام یا بیشتر نقاط الگوهای آموزشی عبور می کند و در نتیجه شبکه به جای یاد گرفتن ویژگی های داده های آموزشی، الگوهای آموزشی را به خاطر می سپارد و به همین دلیل خطای آن در مواجه با داده هایی که ندیده است زیاد می گردد و در نتیجه قدرت تعمیم شبکه کم می شود.

رابطه ساختار با قدرت حافظه

قدرت حافظه شبکه نشان دهنده خطای آموزش شبکه است. در صورتی که مقدار خطای آموزش شبکه کم باشد، قدرت حافظه بالا و در صورت زیاد بودن خطای آموزش شبکه قدرت حافظه پایین می باشد. از طرفی خطای آموزش شبکه با تعداد واحدهای مخفی رابطه معکوس دارد، زیرا در صورتی که تعداد واحدهای شبکه کم باشد امکان عبور از همه نقاط مشخص شده توسط الگوهای آموزشی کم می شود و در نتیجه شبکه ویژگی های الگوهای آموزشی را به خاطر می سپارد. در صورتی که تعداد واحدهای شبکه بالا باشد، امکان عبور از این نقاط زیاد است. چنین شبکه ای مشابه یک جدول عمل می کند و فقط الگوهای آموزشی را به خاطر می سپارد و به همین دلیل دارای قدرت حافظه بالا خواهد بود. بنابر این قدرت حافظه یک شبکه با تعداد واحدهای مخفی آن رابطه مستقیم دارد.

باتوجه به مباحث انجام گرفته در چند پاراگراف قبل واضح است که معیارهای فوق الذکر به همدیگر وابسته هستند و افزایش یکی از آنها ممکن است منجر به کاهش دیگری شود مثلاً افزایش قدرت تعمیم شبکه سبب کاهش قدرت حافظه آن می شود. توازن بین این معیارها از وظایف الگوریتم آموزش شبکه است. توازن بین این معیارها در صورتی برقرار می شود که ساختار مناسب برای شبکه انتخاب شده باشد. در حال حاضر هیچ روش تحلیلی برای تعیین ساختار مناسب شبکه برای یک مسئله وجود ندارد و ساختار مناسب را تنها برای مسائل و شبکه های بخصوصی و برای افزایش معیار مشخصی می توان تعیین نمود. مثلاً در یک شبکه سه لایه، برای تولید یک نگاشت صحیح N الگوی ورودی، تعداد واحد - های مخفی لازم برابر با N-1 می باشد [۲] [۱۰][۳۵] و الزاماً این ساختار مناسب ترین ساختار نیست و توازن بین همه معیارها را برقرار نمی کند، زیرا در این شرایط تنها قدرت حافظه حداکثر می شود.

یک شبکه عصبی را می توان توسط یک گراف $N = (V, E)$ نمایش داد که V مجموعه واحدها و E مجموعه وزن های شبکه می باشد. روش های بهینه سازی ساختار شبکه از وضعیت[12] موجود شبکه $N_1 = (V_1, E_1)$ به دنبال یافتن وضعیت مناسب تر شبکه $N_2 = (V_2, E_2)$ هستند. الگوریتم بهینه سازی ساختار شبکه را می توان به عنوان یک نگاشت به صورت $\Delta: N \rightarrow N$ تعریف نمود که از روی ساختار N_1، ساختار N_2 را پیدا می کند. فضای جستجو را نیز می توان به عنوان یک گراف $A = (N, \delta)$ در نظر گرفت که N مجموعه گره ها و δ مجموعه کمان های این گراف است. هر گره از این گراف یک ساختار از شبکه را نشان می دهد و کمان های این گراف نمایانگر نگاشت در الگوریتم تعیین ساختار شبکه می باشد. هدف الگوریتم های تعیین ساختار شبکه، پیمایش این گراف به منظور پیدا کردن مناسبترین گره (شبکه) با حداقل هزینه می باشد.

طراحی یک شبکه با ساختار بهینه یک مسئله NP-Hard است [۱۴]. به همین جهت بیشتر الگوریتم های ارائه شده برای تعیین ساختار شبکه های عصبی، الگوریتم های تقریبی هستند. این الگوریتم ها قبل، در حین یا بعد از یادگیری ساختار مناسب برای شبکه را تعیین می نمایند. بعضی از این الگوریتم ها از اطلاعات محلی و بعضی دیگر از اطلاعات عمومی برای یافتن ساختار مناسب استفاده می کنند. این الگوریتم ها را می توان به چهار گروه زیر تقسیم کرد:

الف ـ الگوریتم‌های هرس[13]

این الگوریتم‌ها از یک شبکه بزرگ شروع نموده و به تدریج در حین آموزش یا بعد از آن واحد ها یا وزن‌های اضافی را از شبکه هرس می‌کنند. در این الگوریتم‌ها نیاز است که تعداد واحدهای مخفی را در ابتدای آموزش مشخص نماییم. البته این مشکل حادی را ایجاد نمی‌کند زیرا کران بالای تعداد واحدهای مخفی برای یک مسئله معین و برای بعضی از انواع شبکه‌ها مشخص است [۲] [۱۰] [۳۵]. این الگوریتم‌ها هم از مزایای شبکه‌های بزرگ (کم بودن پیچیدگی آموزش و دوری از حداقل‌های محلی) و هم از مزایای شبکه‌های کوچک (قدرت تعمیم بالا) بهره می‌برند. در این الگوریتم‌ها برای دو وضعیت متوالی $N_1 = (V_1, E_1)$ و $N_2 = (V_2, E_2)$ دو شرط $V_1 \subseteq V_2$ و $E_2 \subseteq E_1$ برقرار می‌باشد [۵] [۱۱] [۸۲] [۳۰] [۳۱].

ب ـ الگوریتم‌های سازنده[14]

این الگوریتم‌ها با یک شبکه کوچک شروع می‌کند و به تدریج در حین آموزش شبکه، واحد یا لایه مخفی به شبکه می‌افزایند. این الگوریتم‌ها معمولاً شبکه‌های کوچکی را تولید می‌کنند که دارای پیچیدگی آموزش بالایی هستند [۱۳]. در این الگوریتم‌ها برای دو ساختار متوالی $N_1 = (V_1, E_1)$ و $N_2 = (V_2, E_2)$ دو شرط $V_1 \subseteq V_2$ و $E_1 \subseteq E_2$ برقرار می‌باشد [۳] [۶] [۷] [۱۳] [۱۶] [۱۹] [۲۲] [۳۲] [۳۹].

ج ـ الگوریتم‌های ترکیبی

این گروه از الگوریتم‌ها، از ترکیبی از الگوریتم‌های سازنده و الگوریتم‌های هرس برای تعیین ساختار شبکه استفاده می‌کنند. در این الگوریتم‌ها برای رسیدن به شبکه مطلوب می‌توان وزن، واحد یا لایه را کم یا زیاد نمود [۹] [۲۴].

د ـ الگوریتم‌های تکاملی[15]

در این الگوریتم‌ها تعیین ساختار بهینه برای شبکه از طریق جستجو در فضای ساختارها انجام می‌گیرد. هر نقطه از این فضا نماینده یک ساختار شبکه است. الگوریتم جستجو با استفاده از یک معیار کارایی مانند حداقل خطا و یا پیچیدگی آموزش به دنبال مناسب‌ترین ساختار می‌باشد [۱] [۱۵] [۳۳] [۳۷] [۳۸].

برای اطلاعات بیشتر در مورد الگوریتم‌های فوق می‌توانید به مرجع [۴] مراجعه نمایید.

در این مقاله روشی براساس اتوماتان‌های یادگیر[16] برای تعیین ساختار نزدیک به بهینه شبکه‌های عصبی سه لایه (ساختار با حداقل تعداد واحدهای مخفی) پیشنهاد شده است. این روش، آموزش را با یک شبکه بزرگ شروع می‌کند و اتومان یادگیر با افزودن و کاستن واحدها سعی در پیدا کردن ساختار مناسب برای شبکه را دارد. هر چند ممکن است این الگوریتم را در گروه الگوریتم‌های ترکیبی قرار داد، زیرا واحدهای مخفی هم هرس و هم اضافه می‌شوند، اما هدف استفاده از الگوریتم‌های سازنده یا الگوریتم‌های هرس یا ترکیبی از آنها نیست بلکه تعیین ساختار شبکه به عنوان مسئله افراز[17] مجموعه تعریف شده است.

بخش‌های بعدی مقاله به صورت زیر سازماندهی شده است. مقدمه‌ای بر اتوماتان‌های یادگیر در بخش ۲ آمده است. اتوماتان پیشنهادی و الگوریتم بقا برای تعیین ساختار شبکه‌های عصبی سه لایه به ترتیب در بخش‌های ۳ و ۴ بیان شده‌اند. نتایج آزمایش‌ها برای مسائل مختلف در بخش ۵ و بررسی رفتار اتوماتان پیشنهاد شده در بخش ۶ آمده است. آخرین بخش این مقاله نتیجه‌گیری می‌باشد.

۲ ـ اتوماتان‌های یادگیر

یادگیری در اتوماتان‌های یادگیر، انتخاب یک اقدام[18] بهینه از میان یک مجموعه از اقدام‌های مجاز می‌باشد. این اقدام روی یک محیط تصادفی اعمال می‌شود و محیط به این اقدام اتوماتان به وسیله یک پاسخ از مجموعه پاسخ‌های مجاز جواب می‌دهد. پاسخ محیط به صورت آماری به اقدام اتوماتان وابسته است. اصطلاح محیط شامل اجتماع تمام شرایط خارجی و تأثیرات آنها روی عملکرد اتوماتان می‌باشد.

محیط به صورت سه تایی $(\underline{\alpha}, \underline{C}, \underline{\beta})$ نشان داده می‌شود. مجموعه $\underline{\alpha} = \{\alpha_1, \ldots, \alpha_r\}$ مجموعه ورودی‌ها، مجموعه $\underline{C} = \{c_1, \ldots, c_r\}$ مجموعه احتمالات (c_i احتمال شکست اقدام α_i می‌باشد) و مجموعه $\underline{\beta} = \{\beta_1, \beta_2\}$ خروجی دودویی محیط می‌باشد [۲۵]. اتصال یک اتوماتان با محیط در شکل (۳) نشان داده شده است.

شکل (۳) اتصال اتوماتان یادگیری با محیط.

اتوماتان های یادگیر به دو خانواده اتوماتان یادگیر با ساختار ثابت[۱۹] و اتوماتان یادگیر با ساختار متغیر[۲۰] دسته بندی می شوند. اتوماتان های Krinsky, Tsetline و Krylov مثال هایی از خانواده اتوماتان های با ساختار ثابت هستند. یک اتوماتان یادگیر با ساختار ثابت را می توان با یک پنج تایی $\{\underline{\alpha}, \underline{\Phi}, \underline{\beta}, F, G\}$ نشان داد. که $\underline{\alpha} = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه اقدام های مجاز برای اتوماتان یادگیر، $\underline{\Phi} = \{\Phi_1, \Phi_2, ..., \Phi_S\}$ مجموعه وضعیت های اتوماتان، $\underline{\beta} = \{0, 1\}$ مجموعه ورودی ها (در این مجموعه یک نمایانگر شکست[۲۱] و صفر نمایانگر موفقیت[۲۲] می باشد)، $F: \underline{\Phi} \times \underline{\beta} \rightarrow \underline{\Phi}$ تابع نگاشت وضعیت ها و $G: \underline{\Phi} \rightarrow \underline{\alpha}$ تابع نگاشت خروجی می باشد.

اقدام اتوماتان به عنوان ورودی به محیط داده می شود و محیط بعد از اعمال اقدام داده شده توسط اتوماتان یک پاسخ تصادفی که می تواند موفق یا ناموفق باشد را تولید می کند که به عنوان ورودی به اتوماتان داده می شود. اتوماتان باتوجه به پاسخ محیط اقدام مربوطه را جریمه می کند و یا به آن پاداش می دهد. اگر احتمال تغییر وضعیت ها در اتوماتان ثابت باشد، آن را اتوماتان یادگیر با ساختار ثابت و در غیر این صورت آن را اتوماتان یادگیر با ساختار متغیر می نامند. در ادامه به طور مختصر چند اتوماتان با ساختار ثابت شرح داده می شود. برای اطلاعات بیشتر در مورد اتوماتان های با ساختار متغیر به مراجع [۱۷] [۲۵] [۴۴] [۴۵] [۴۶] مراجعه نمایید.

الف ـ اتوماتان Tsetline ($L_{2N,2}$)

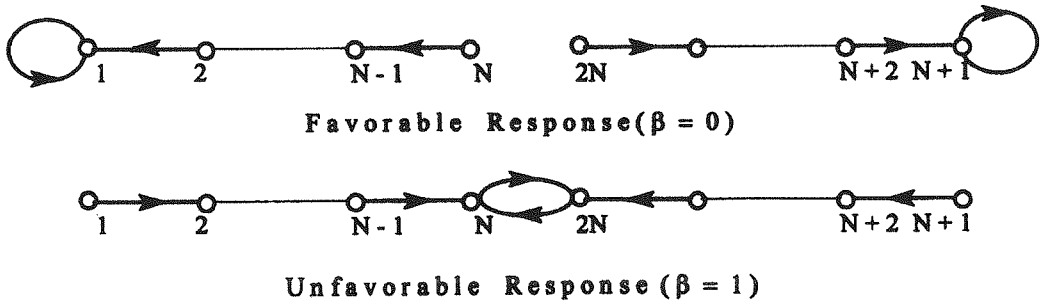
این اتوماتان 2N وضعیت و ۲ اقدام دارد وسعی دارد از رفتار گذشته محیط برای تصمیم گیری های آینده خود استفاده نماید. این اتوماتان تعداد پاسخ های دریافت شده برای هر اقدام را می شمارد و در صورتی اقدام خود را تغییر می دهد که تعداد پاسخ های ناموفق بیشتر از تعداد پاسخ های موفق و یا عدد ثابت N (عمق حافظه) باشد. برای هر پاسخ دریافت شده موفق (ناموفق)، اتوماتان به سمت وضعیت های داخلی تر (وضعیت های خارجی تر) اقدام انتخاب شده حرکت می کند. گراف تغییر حالت اتوماتان در شکل (۴) نشان داده شده است.

ب ـ اتوماتان Krinsky

عملکرد این اتوماتان برای پاسخ ناموفق محیط مانند عملکرد اتوماتان Tsetline است. اما برای پاسخ های موفق محیط، این اتوماتان به داخلی ترین وضعیت اقدام انتخاب شده حرکت می کند. یعنی برای هر پاسخ موفق، N پاسخ متوالی ناموفق لازم است تا اقدام اتوماتان تغییر نماید. گراف تغییر حالت اتوماتان در شکل (۵) نشان داده شده است.

ج ـ اتوماتان Krylov

عملکرد این اتوماتان برای پاسخ موفق محیط مانند عملکرد اتوماتان Tsetline است. اما برای پاسخ های ناموفق محیط، این اتوماتان با احتمال مساوی به وضعیت بعد یا قبل از خود حرکت می کند. گراف تغییر



شکل (۴) اتوماتان Tsetline.

حالت اتوماتان در شکل (۶) نشان داده شده است.

ملاک های رفتاری اتوماتان[۲۲]

برای بررسی یادگیری در اتوماتان،وضعیت آن را با گذشت زمان بررسی می کنند. یک معیار برای بررسی رفتار اتوماتان، متوسط جریمه دریافتی از محیط است که در زمان n برابر است با $M(n) = \sum_{k=1}^{r} c_k p_k(n)$. که c_k احتمال دریافت پاسخ ناموفق از محیط است در صورتی که اقدام k ام به محیط اعمال شود و $p_k(n)$ احتمال انتخاب اقدام k ام در زمان n می باشد. بدون هیچگونه اطلاعات قبلی، اقدام های اتوماتان با احتمال مساوی انتخاب می شوند. به اتوماتانی که اقدام های آن با احتمال مساوی انتخاب می شوند، اتوماتان شانس[۲۳] می گویند. در اتوماتان شانس متوسط جریمه دریافتی (M_0) برابر است با $M_0 = \frac{\sum_{k=1}^{r} c_k}{r}$.

یک اتوماتان را مصلحت گرا[۲۴] می گویند اگر در زمان n (وقتیکه n به سمت بی نهایت میل نماید) متوسط جریمه دریافت شده کمتر از M_0 باشد یا به بیان دیگر بهتر از اتوماتان شانس عمل نماید.

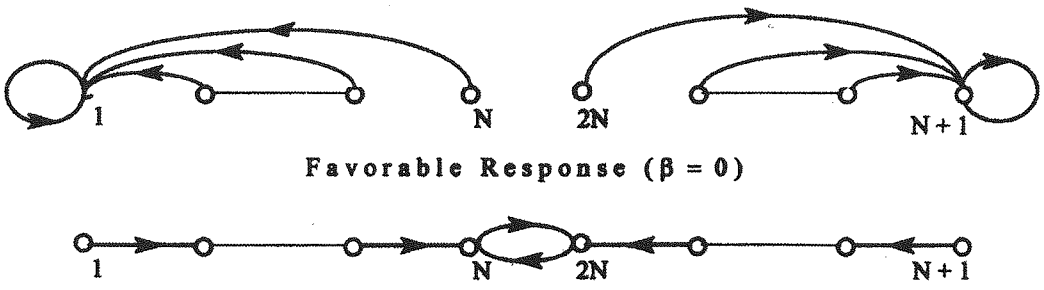
یک اتوماتان را بهینه[۲۵] می گویند اگر در زمان n (وقتیکه n به سمت بی نهایت میل نماید) متوسط جریمه دریافت شده برابر $\min\{c_1, ..., c_2\}$ باشد.

یک اتوماتان را ε-Optimal می گویند اگر برای هر $\varepsilon > 0$ در زمان n (وقتیکه n به سمت بی نهایت میل نماید) متوسط جریمه دریافت شده کوچکتر از $\min\{c_1, c_2, ..., c_r\} + \varepsilon$ باشد.

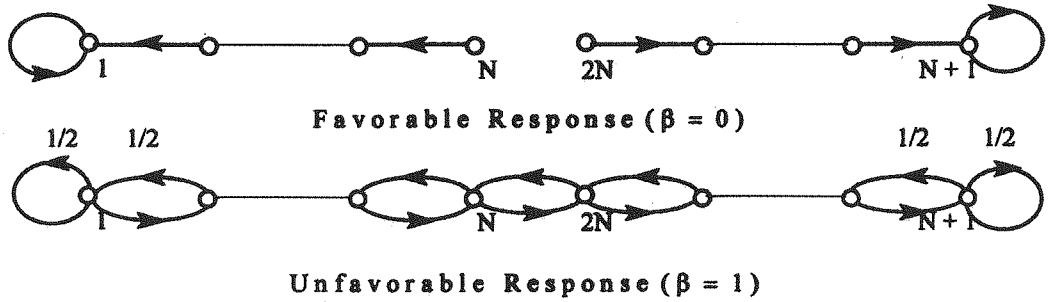
با انتخاب مقادیر مناسب پارامترهای اتوماتان (مثلاً تعداد وضعیت های اتوماتان با ساختار ثابت) می توان به اتوماتان با رفتار مورد نظر دسترسی پیدا کرد.

اتوماتان های یادگیر دارای کاربردهای فراوانی می باشد. بعضی از این کاربردها عبارتند از: مسیریابی در شبکه های ارتباطی [۱۸]، فشرده سازی تصاویر [۸]، شناسایی الگو [۳۶]، برنامه ریزی فرایندها[۲۶] در یک شبکه کامپیوتری [۲۵]، تئوری صف [۲۱]، کنترل دسترسی در شبکه های انتقال ناهمزمان [۱۷]، کمک به آموزش شبکه های عصبی [۲۰] و دسته بندی و افراز اشیاء [۲۶].

در صورتی که اتوماتان های یادگیر برای دسته بندی یا افراز اشیاء استفاده شوند، علیرغم سرعت همگرایی خوب دارای تعداد اقدام های بسیار بالا خواهند بود [۲۶]. برای پایین آوردن تعداد خروجی ها ، اتوماتان مهاجرت اشیاء[۲۴] توسط اومن[۲۵] و ما[۲۶] پیشنهاد شده



شکل (۵) اتوماتان Krinsky.



شکل (۶) اتوماتان Krylov.

است [۲۶]. تعداد اقدام های این اتوماتان به مراتب کمتر از اتومـاتان های قـبلی است ولی در عـوض سـرعت همگرایی آن پایین تر می باشد.

اتوماتان مهاجرت اشیاء

اتومـاتان مـهاجرت اشـیاء توسط پنج تایـــــــی $(\underline{\alpha}, \underline{\Phi}, \underline{\beta}, \underline{F}, \underline{G})$ نشـان داده می شـــــــود [۲۶]. که $\underline{\alpha} = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعه اقدام های مجـــاز، $\underline{\Phi} = \{\Phi_1, \Phi_2, ..., \Phi_S\}$ مجموعه وضعیت ها، $\underline{\beta} = \{0, 1\}$ مـجـمـوعـه ورودی ها، $F : \underline{\Phi} \times \underline{\beta} \rightarrow \underline{\Phi}$ تابع نگاشت وضعیت ها و $G : \underline{\Phi} \rightarrow \underline{\alpha}$ تابع نگاشت خروجی اتوماتان می باشند. این نوع اتومـاتان برای دسـتـه بندی اشـیـاء [۲۶]، انتساب حروف به کلیدها [۲۷]، افراز گراف[۲۷] [۲۸] و تناظر گراف [۲۷] [۴۲] مورد استفاده قرار گرفته است.

در این اتوماتان هر اقدام یک دسته را نشان می دهد. در اتوماتانهای با ساختار ثابت، پاسخ محیط به اتوماتان سبب می شود که اتومـاتان از یک وضـعیت به وضـعیت دیگر منتقل شود، در صورتی که در اتومـاتان مـهاجرت اشیاء ، اشیاء به وضـعیت هـا انتسـاب داده می شوند و پاسخ مـحیط به اتومـاتان سـبب گردش اشـیـاء در بین وضـعیت های اتومـاتان می گردد. از طریق این گـردش دسته بندی اشیاء انجام می گیرد.

اگر شی W_i در اقدامj ام اتوماتان مهاجرت اشیاء قرار داشـته باشـد، این شیء متـعلق به دسـتـــــــه شـماره j است. برای اقدام α_k مجموعه وضـعیـــــــــــت $\{\Phi_{(k-1)N+1}, ..., \Phi_{kN}\}$ در نظر گرفتـه می شود که N عمق حـافظه را نشـان می دهد. بـدون از دست دادن عـمـومـیت بحث می تـــوان $\Phi_{(k-1)N+1}$ را داخلی ترین وضـعیت و Φ_{kN} را خـارجی ترین وضـعیت این اقدام در نظر گرفت. اگر دو شی w_i و w_j به ترتیب در وضـعیت هـــای $\Phi_{(k-1)N+1}$ و $\Phi_{(k-1)N+m}$ (برای $m > 1$) قـرار داشـتـــه باشند، در این صورت احتمال تعلق شیء w_i به این دسته از احتمال تعلق شیء w_j بیشـتر است. بنابراین برای اقدام α_k، وضعیت $\Phi_{(k-1)N+1}$، وضـعیت با بیشترین احـتمال و وضـعیت Φ_{kN} وضعیت با کمترین احتمال نامیده می شود.

۳ ـ اتوماتان یادگیر تعیین تعداد واحدهای لایه مخفی[۲۸]

در این قسمت یک اتومـاتان از نوع مـهاجرت اشـیاء برای تعیین تعداد واحدهای لایه مخفی یک شبکه سه لایه که توسط الگوریتم انتشـار خطا به عـقب آمـوزش داده می شود معرفی می گردد. وظیفه این اتوماتان دسته بندی واحدهـای لایه مـخـفی به دو دسـتـه واحدهـای مناسب و واحدهای نامناسب می باشد. این اتومـاتان به صورت یک شش تایی $(\underline{\alpha}, \underline{H}, \underline{\Phi}, \underline{\beta}, \underline{F}, \underline{G})$ نشان داده می شود که در آن

۱ ـ $\underline{\alpha} = \{\alpha_1, \alpha_2\}$ مـجـمـوعـه اقـدام های مـجـاز برای اتوماتان یادگیر است. این اتوماتان دو اقدام دارد که اقدام شماره یک آن اقدام مناسب یا واحدهای روشن نام دارد. واحدهایی که در این اقدام قـرار دارند برای آموزش شبکه عصبی مـورد استفاده قرار می گیرند. اقدام شماره دو آن اقدام نامناسب یا واحدهای خـاموش نام دارد. واحـدهایی که در این اقدام قرار می گیرند برای آموزش شبکه عصبی مورد استفاده قرار نمی گیرند.

۲ ـ $\underline{H} = \{H_1, H_2, ..., H_n\}$ مجموعه واحدهای مخفی موجود در وضـعیت های اتوماتان می باشد. اگر واحد H_i در اقدام شـماره یک اتومـاتان ظاهر شـود، این واحد به عنوان واحد مخفی مناسب (روشن) و در غیر این صورت این واحد به عنوان واحد مـخـفی نامناسب (خـامـوش) در نظر گرفته می شود.

۳ ـ $\underline{\Phi} = \{\Phi_1, \Phi_2, ..., \Phi_{2N}\}$ مجموعه وضـعیت ها و N عـمق حـافظه برای اتومـاتان می بـاشـد. مـجـمـوعـه وضـعیت هـای این اتومـاتان به دو زیر مـجـمـوعـــــــه $\{\Phi_1, ..., \Phi_N\}$ و $\{\Phi_{N+1}, ..., \Phi_{2N}\}$ افـراز مـی شـود و براسـاس اینکه واحدهای مـخـفی در چه وضـعـیتی قـرار داشـتـه باشند، دسـتـه بندی می شـــــــوند. بر این اسـاس واحدهای روشن بـا مـجـمـوعـــــــــــه $ON = \{H_i \mid 1 \le State(H_i) \le N\}$ و واحدهای خـاموش با مـجـمـوعـه $OFF = \{H_i \mid N+1 \le State(H_i) \le 2N\}$ نشـان داده می شوند و $State(H_i)$ نشاندهنده وضـعیتی است که واحد H_i در آن قرار دارد.

۴ ـ $\underline{\beta} = \{0, 1\}$ مـجـمـوعـه ورودی هـای اتومـاتان می باشـد. در این مـجـمـوعـه یک نمایانگر شکسـت و صفر نمایانگر موفقیت می باشد.

۵ ـ $F: \underline{\Phi} \times \underline{\beta} \rightarrow \underline{\Phi}$ تابع نگاشت وضعیت ها می باشد. این تابع از روی وضعیت فعلی ورودی اتوماتان وضعیت بعـدی آن را تولیـد مـی نماید. در واقع این تابع چگونگی گردش واحـدهای مـخـفی را در وضـعیت های اتومـاتان مشخص می کند. شرح کار کرد این تابع در قسمت بعدی خواهد آمد.

۶ ـ $G: \underline{\Phi} \rightarrow \underline{\alpha}$ تابع نگاشت خـروجی می بـاشـد. این تابع تصـمیم می گیرد که به ازای هر وضعیت، اتومـاتان

چه اقدامی را انجام دهد. اگر واحد H_i متعلق به مجموعه وضعیت های $\{\Phi_1, \Phi_2, ..., \Phi_N\}$ باشد، این واحد روشن در نظر گرفته می شود. اگر واحد در وضعیت Φ_1 قرار داشته این مناسبترین واحد است و بیشترین اهمیت را دارا می باشد. اگر در وضعیت Φ_N قرار داشته باشد، دارای کمترین اهمیت می باشد. اقدام شماره دو نیز بهمین صورت می باشد. اگر واحد متعلق به مجموعه وضعیت های $\{\Phi_{N+1}, ..., \Phi_{2N}\}$ باشد واحد را خاموش در نظر می گیریم. اگر واحد در وضعیت Φ_{N+1} قرار داشته باشد، بیشترین اهمیت را دارد و اگر در وضعیت Φ_{2N} قرار داشته باشد، دارای کمترین اهمیت می باشد.

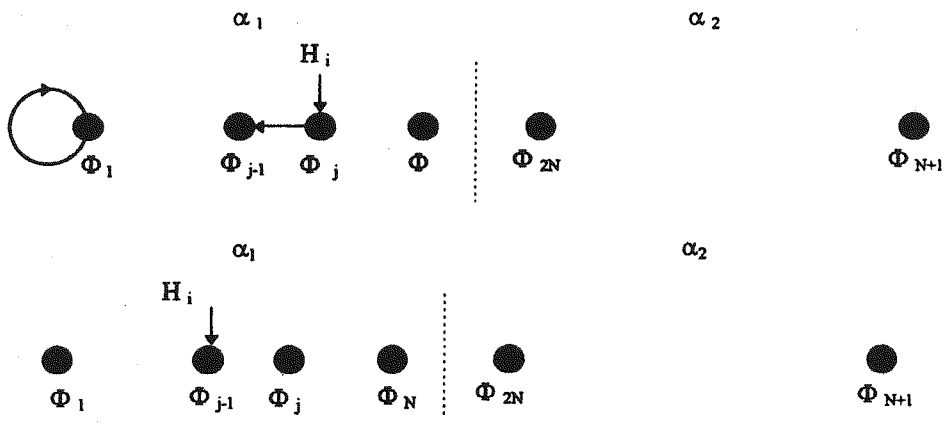
برای سهولت نمایش در ارائه مطالب، اتوماتان یادگیر تعیین تعداد واحدهای لایه مخفی با K اقدام، عمق حافظه N و انتساب M واحد مخفی توسط HULA (K, N, M) نشان داده می شود. برای تشریح تابع نگاشت وضعیت ها چهار حالت زیر را در نظر می گیریم.

۱ ـ واحد مخفی H_i روشن است و در وضعیت Φ_j قرار دارد و به دلیل عملکرد مناسبش پاداش می گیرد. در این صورت اهمیت این واحد بیشتر شده و به سمت وضعیت های داخلی تر این اقدام حرکت می کند. نحوه حرکت چنین واحدی در شکل (۷) نشان داده شده است.

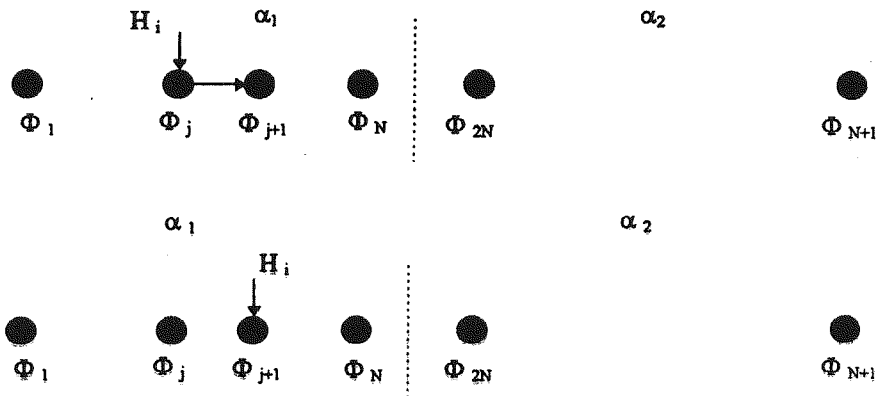
اگر واحد مخفی H_i در وضعیت Φ_1 قرار داشته باشد و پاداش بگیرد در همان وضعیت باقی می ماند.

۲ ـ واحد H_i روشن است و در وضعیت Φ_j قرار دارد و به دلیل عملکرد نامناسبش جریمه می شود. در این صورت از اهمیت این واحد کاسته شده وبه سمت وضعیت های بیرونی تر حرکت می کند. نحوه حرکت چنین واحدی برای دو حالت مختلف در اشکال (۸) و (۹) نشان داده شده است.

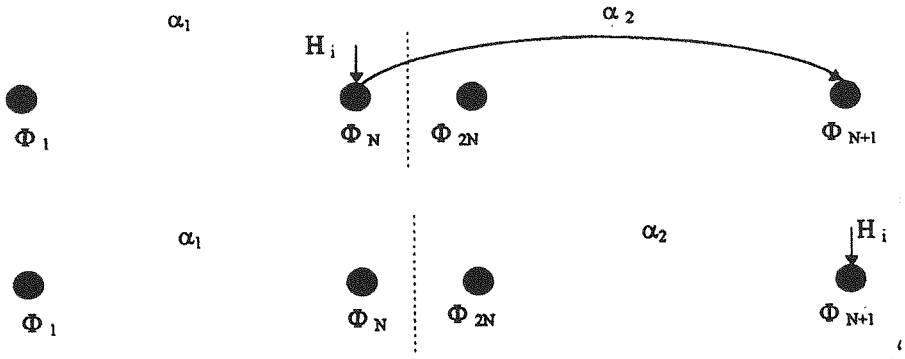
۳ ـ واحد مخفی H_i خاموش است و در وضعیت Φ_j قرار دارد و به دلیل عملکرد خوبش در گذشته جریمه می شود. در این صورت به واحدهای روشن نزدیکتر می شود یعنی به وضعیت های بیرونی تر منتقل می شود. نحوه حرکت چنین واحدی برای دو حالت مختلف در اشکال (۱۰) و (۱۱) نشان داده شده است.



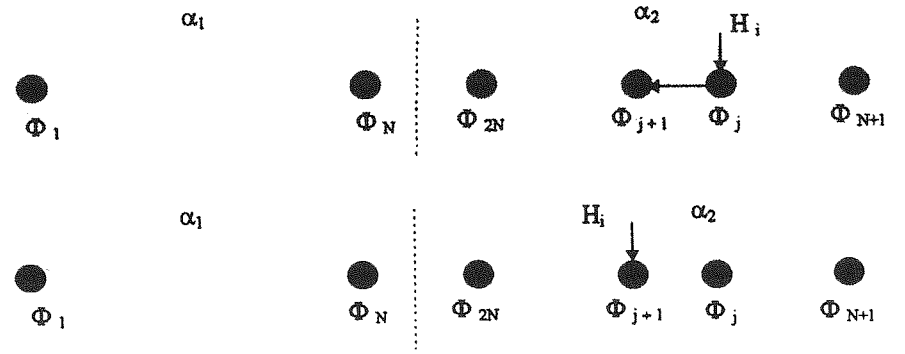
شکل (۷) نحوه دادن پاداش به یك واحد روشن.



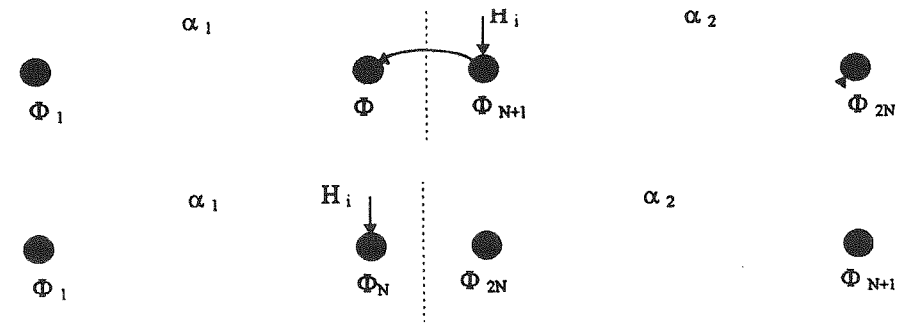
شکل (۸) نحوه دادن جریمه به یك واحد روشن برای یك وضعیت غیرمرزی.



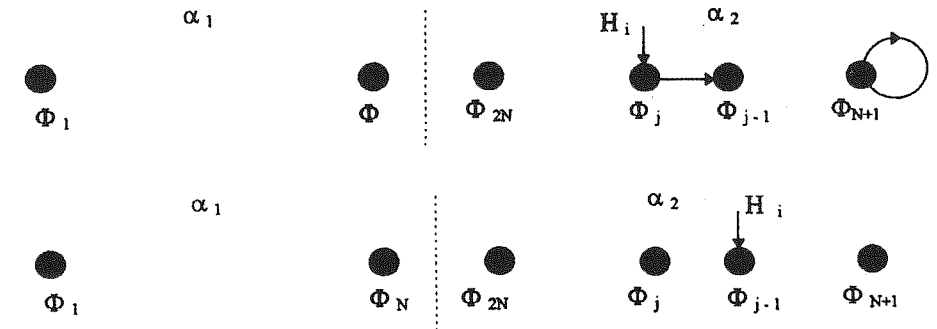
شکل (۹) نحوه دادن جریمه به یك واحد روشن برای یك وضعیت مرزی.



شکل (۱۰) نحوه جریمه شدن یك واحد خاموش برای یك وضعیت غیرمرزی.



شکل (۱۱) نحوه جریمه شدن یك واحد روشن برای یك وضعیت مرزی.



شکل (۱۲) نحوه دادن پاداش یك واحد خاموش.

۴ ـ واحد مخفی H_i خاموش است و در وضعیت Φ_j قرار دارد و به دلیل عملکرد نامناسبش در گذشته پاداش می‌گیرد، در این صورت اهمیت این واحد کمتر می‌شود و از واحدهای روشن دورتر می شود، یعنی به وضعیت های داخلی تر اتوماتان منتقل می‌شود. نحوه حرکت چنین واحدی در شکل (۱۲) نشان داده شده است.

اگر واحد مخفی H_i در وضعیت Φ_{2N} قرار داشته باشد و جریمه شود در همان وضعیت باقی می‌ماند.

٤ ـ الگوریتم تعیین تعداد واحدهای لایه مخفی

در این قسمت الگوریتمی معرفی می‌شود که با استفاده از اتوماتان یادگیر معرفی شده در قسمت قبل و الگوریتم انتشار خطا به عقب، ساختار بهینه شبکه را در حین آموزش مشخص می‌کند. طرز کار این الگوریتم به صورت زیر است.

در ابتدا همه واحدها در مجموعه واحدهای روشن و در وضعیت Φ_1 قرار می‌گیرند. به همه واحدهای روشن مدتی اجازه داده می‌شود تا در آموزش شبکه شرکت نمایند. واحدهایی که عملکرد آنها در این مدت مناسب نبوده جریمه می‌شوند، واحدهایی که عملکرد آنها خیلی خوب بوده پاداش می‌گیرند و واحدهایی که در مورد آنها نمی‌توان تصمیم‌گیری قطعی نمود بدون تغییر وضعیت باقی می‌مانند. برای تشخیص خوب یا بد بودن عملکرد یک واحد از متوسط انرژی مصرف شده آن واحد استفاده می‌شود. چگونگی تغییر فعالیت یک واحد به ازای همه الگوهای آموزشی، انرژی مصرف شده یک واحد نامیده می‌شود و توسط دو قانون مکاشفه ای زیر بیان می‌شود.

عملکرد خوب یک واحد: واحدی دارای عملکرد خوب می‌باشد، اگر به ازای همه الگوهای ورودی، مقدار فعالیت این واحد تغییرات زیادی داشته باشد. این بدان معنی است که اطلاعات ذخیره شده در وزن‌های این واحد مهم می‌باشد.

عملکرد بد یک واحد: واحدی دارای عملکرد بد می‌باشد، اگر به ازای همه الگوهای ورودی، مقدار فعالیت این واحد تغییرات کم داشته باشد. یعنی اطلاعات ذخیره شده در وزن‌های این واحد مهم نیستند.

تشخیص نحوه عملکرد یک واحد روشن

واحدی را که مقدار فعالیت آن به ازای همه الگوهای آموزشی از یک مقدار آستانه ای کمتر باشد واحد بد واگر از مقدار فعالیت آن به ازای همه الگوهای آموزشی ازیک مقدار آستانه ای دیگر بیشتر باشد واحد خوب می‌نامیم. برای تعیین مقادیر آستانه ای ابتدا واریانس فعالیت واحد به ازای همه الگوهای آموزشی به صورت زیر محاسبه می‌شود

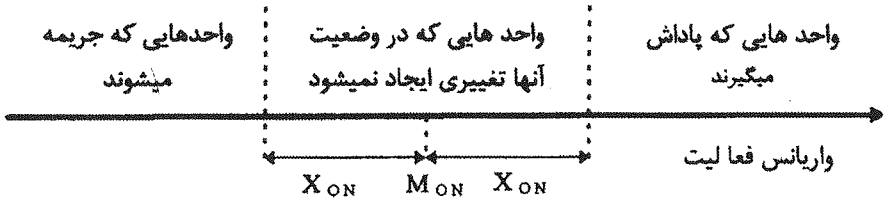
$$\sigma_I = \sqrt{\frac{\sum_{K=1}^{P} \left(|U_{IK}| - \mu_I\right)^2}{P}} \qquad I \in ON$$

که در آن U_{IK} مقدار فعالیت واحد شماره I به ازای الگوی شماره K و P تعداد الگوهای آموزشی می‌باشد. μ_I نشاندهنده میانگین فعالیت واحد I ام است که به صورت زیر تعریف می‌شود.

$$\mu_I = \frac{\sum_{K=1}^{P} |U_{IK}|}{P} \qquad I \in ON$$

پس از محاسبه واریانس فعالیت واحدهای روشن، واحدهای روشنی که واریانس فعالیت آنها از یک مقدار آستانه ای کمتر باشند، جریمه می‌شوند و واحدهای روشنی که واریانس فعالیت آنها از مقدار آستانه ای دیگری بیشتر باشند پاداش می‌گیرند. واحدهایی که واریانس فعالیت آنها بین این دو مقدار آستانه ای قرار می‌گیرند، تغییری در وضعیت آنها ایجاد نمی‌شود (شکل ۱۳).

مقدار M_{ON} که عبارتست از میانگین واریانس واحدهای روشن به صورت زیر محاسبه می‌شود.



شکل (۱۳) تشخیص مقدار آستانه واحدهای روشن.

$$M_{on} = \frac{\sum_{K \in ON} \sigma_K}{|ON|}$$

که |ON| تعداد اعضای مجموعه ON را نشان می‌دهد. پهنای X_{ON} در شکل فوق به صورت زیر محاسبه می‌شود.

$$X_{on} = \lambda_{on} \frac{|ON| + |OFF|}{|ON|} \times \frac{Max\,(\sigma_{ON})}{Min\,(\sigma_{ON})}$$

در رابطه بالا ثابت $\lambda_{ON} \geq 0$ ضریب پهنای روشنی نامیده می‌شود. $Max\,(\sigma_{ON})$ و $Min\,(\sigma_{ON})$ به ترتیب بیشترین و کمترین مقدار واریانس فعالیت واحدهای روشن می‌باشند وبه صورت زیر محاسبه می‌گردند.

$$Max\,(\sigma_{ON}) = \underset{I \in ON}{Max}\{\sigma_I\} \quad , \quad Min\,(\sigma_{ON}) = \underset{I \in ON}{Min}\{\sigma_I\}$$

واحدهایی که مقدار واریانس فعالیت آنها از $M_{on} - X_{on}$ کمتر باشد، جریمه می‌شوند. واحدهایی را که مقدار واریانس فعالیت آنها از $M_{ON} + X_{ON}$ بیشتر باشند، پاداش می‌گیرند و واحدهایی که مقدار واریانس آنها در فاصله $[M_{ON} - X_{ON}, M_{ON} + X_{ON}]$ قرار دارند، وضعیت آنها تغییر نمی کند.

تشخیص نحوه عملکرد یک واحد خاموش

واحدهایی که خاموش هستند، در آموزش شبکه شرکت نمی‌کنند، بنابراین فعالیت و واریانس فعالیت برای این واحدها وجود ندارد. در این صورت از فعالیت و واریانس فعالیت قبلی این واحدها برای مشخص شدن وضعیت کنونی آنها استفاده می‌شود. فعالیت یک واحد خاموش برای یک الگو براساس آخرین مقدار فعالیت این واحد در زمان روشن بودن برای آن الگو محاسبه می‌شود. اگر یک واحد برای مدت طولانی خاموش باشد از ارزش فعالیت آن کاسته می‌شود، زیرا نقطه‌ای که هم اکنون روی سطح خطا محاسبه می‌شود به احتمال زیاد به نقطه جواب بسیار نزدیکتر از نقطه ای است که در زمان روشن بودن آن واحد محاسبه شده است. لذا فعالیت یک واحد خاموش را می توان به صورت زیر محاسبه نمود.

$$U_{IK}(n+1) = U_{IK}(n)\, e^{-\lambda_d |U_{IK}(n)|}$$

در رابطه فوق ثابت $\lambda_d \geq 0$ ضریب کاهش فعالیت نامیده می‌شود و n شاخص زمان را نشان می‌دهد. به این ترتیب مقدار فعالیت یک واحد خاموش به تدریج کم می‌شود. واریانس واحدهای خاموش به صورت زیر محاسبه می‌شود.

$$\sigma_I = \sqrt{\frac{\sum_{K=1}^{P} \left(|U_{IK}| - \mu_I\right)^2}{P}} \qquad I \in OFF$$

که در آن U_{IK} مقدار فعالیت واحد خاموش شماره I به ازای الگوی شماره K و μ_I نشاندهنده میانگین فعالیت واحد خاموش I است که به صورت زیر تعریف می‌شود.

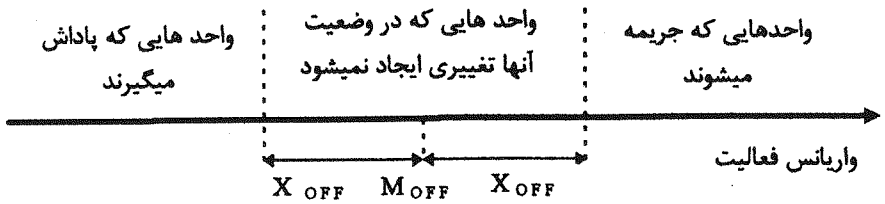
$$\mu_I = \sqrt{\frac{\sum_{K=1}^{P} U_{IK}}{P}} \qquad I \in OFF$$

پس از محاسبه واریانس فعالیت واحدهای خاموش، واحدهای خاموشی که واریانس فعالیت آنها از یک مقدار آستانه‌ای کمتر باشند پاداش می‌گیرند و واحدهای خاموشی که واریانس فعالیت آنها از مقدار آستانه‌ای دیگر بیشتر باشد جریمه می‌شوند. واحد هایی که واریانس فعالیت آنها بین این دو مقدار آستانه ای قرار می‌گیرند تغییری در وضعیت آنها ایجاد نمی‌شود (شکل ۱۴).

و مقدار M_{OFF} که عبارتست از میانگین واریانس واحدهای خاموش و به صورت زیر محاسبه می‌شود.

$$M_{OFF} = \frac{\sum_{K \in OFF} \sigma_K}{|OFF|}$$

پهنای X_{OFF} در شکل (۱۴) به صورت زیر محاسبه می‌شود.



شکل (۱۴) تشخیص مقدار آستانه واحدهای خاموش.

$$X_{OFF} = \lambda_{OFF} \frac{|ON| + |OFF|}{|OFF|} \times \frac{Max\,(\sigma_{OFF})}{Min\,(\sigma_{OFF})}$$

در رابطه بالا ثابت $\lambda_{OFF} \geq 0$ ضریب پهنای خاموشی نامیده می‌شود $Max\,(\sigma_{OFF})$ و $Min\,(\sigma_{OFF})$ به ترتیب بیشترین و کمترین مقدار واریانس فعالیت واحدهای خاموش هستند و به صورت زیر محاسبه می‌شوند.

$$Max\,(\sigma_{OFF}) = \underset{I \in OFF}{Max}\,\{\sigma_I\} \quad , \quad Min\,(\sigma_{OFF}) = \underset{I \in OFF}{Min}\{\sigma_I\}$$

واحدهایی که مقدار واریانس فعالیت آنها از $M_{OFF} - X_{OFF}$ کمتر باشد، پاداش می‌گیرند. واحدهایی که مقدار واریانس فعالیت آنها از $M_{OFF} + X_{OFF}$ بیشتر باشد جریمه می‌شوند و واحدهایی که مقدار واریانس آنها در فاصله $[M_{OFF} - X_{OFF}, M_{OFF} + X_{OFF}]$ قرار دارد، تغییری در وضعیت آنها داده نمی‌شود.

برای استفاده از مزایای در حین آموزش شبکه ای بزرگ، در ابتدای آموزش شبکه حساسیت واحدها کم در نظر گرفته می‌شود تا رقابت بین واحدها کمتر باشد و پیچیدگی آموزش کاهش یابد و شبکه به راحتی بتواند از کمینه های محلی عبور نماید. این عمل با انتخاب مقدار کوچک برای شیب تابع فعالیت انجام می‌شود. اما زمانی که شبکه جواب تقریبی را تولید نمود حساسیت واحدها را به تدریج افزایش داده و از این طریق رقابت بین واحدها افزایش می‌یابد. این عمل با افزایش شیب تابع فعالیت انجام می‌شود. برای اینکه الگوریتم بتواند تطبیق شیب تابع فعالیت را به راحتی انجام دهد، مقدار شیب تابع فعالیت به صورت زیر تنظیم می‌شود.

$$\gamma = e^{-\gamma_s \times MSE}$$

که γ شیب تابع سیگموئید $e^{-\lambda x}$ ، $\gamma_s \geq 0$ ضریب تغییر شیب تابع فعالیت و MSE میانگین مربع خطا می‌باشد. زمانی که یک واحد از حالت خاموش به حالت روشن می‌رود برای تعیین مقادیر اولیه وزن های این واحد دو روش وجود دارد. روش اول استفاده از مقادیر تصادفی برای وزن های این واحد و روش دوم استفاده از وزن هایی است که در آخرین زمان روشن بودن محاسبه شده اند می‌باشد. به دلیل تغییر ابعاد فضای وزن ها ، این مقادیر نیز می‌توانند به نوعی تصادفی تلقی شوند. در این مقاله از روش دوم برای تعیین مقادیر اولیه وزن ها استفاده شده است. الگوریتم بقا برای تعیین ساختار مناسب شبکه در شکل ۱۵ نشان داده شده است.

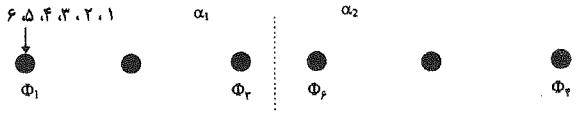
```
Algorithm Survival
  input:
    Training Set (X, T)
    Maximum No. of Hidden Units: Hmax
  output:
    Network Weight Vector: W
    Network Topology: Set of ON
  repeat
    for m := 1 To K do
      call BP      // After K steps hidden units are examined
    end for
    // Decrease the activation of off units
    for all I ∈ OFF do
      for k := 1 To P do
        Uik := exp (- λd | Uik| )
      end for
    end for
    for I := 1 To Hmax do
      Compute σI
    end for
    Compute MON, MOFF, XOFF, XON
    // Move the hidden units among the automatas' states
    for I := 1 to Hmax do
      if I ∈ ON then
        if σI < (MON - XON) then
          call PenalizeOnUnit (I)
        end if
        if σI > (MON + XON) then
          call RewardOnUnit (I)
        end If
      end If
      if I ∈ OFF then
        if σI < (MOFF - XOFF) then
          call RewardOFFUnit (I)
        end if
        if σI > (MOFF + XOFF) then
          call PenalizeOFFUnit (I)
        end if
      end if
    end for
  until Termination Condition is Satisfied.
  return (W, ON)
end Algorithm

procedure PenalizeOnUnit (I)
  inc (State (I))
end procedure

procedure RewardOnUnit (I)
  if State (I) > 1 then
    dec (State (I))
  end if
end procedure

procedure PenalizeOffUnit (I)
  if State (I) ≠ 2 * N then
    inc (State (I))
  else
    State (I) := N
  end if
end procedure

procedure RewardOffUnit (I)
  if State (I) ≠ N + 1 then
    dec (State (I))
  end if
end procedure
```

شکل (۱۵) الگوریتم بقا.

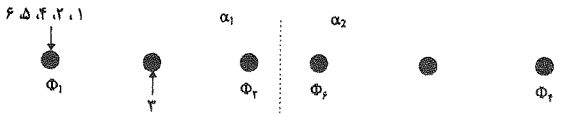
مثالی از عملکرد اتوماتان یادگیر تعیین تعداد واحدهای مخفی

برای روشن تر شدن نحوه کارکرد این اتوماتان مثالی در زیر آورده شده است. در این مثال از اتوماتان HULA (2, 6, 6) با ضرایب $\lambda_{ON} = 0.05$ و $\lambda_{OFF} = 0.01$ استفاده شده است. واریانس فعالیت واحدهای مخفی و نحوه کارکرد اتوماتان در هشت مرحله در جدول زیر آورده شده است.

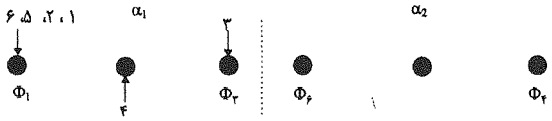
مرحله اول: در ابتدای آموزش، همه واحدهای مخفی در وضعیت ۱ قرار می‌گیرند. در این مرحله وضعیت اتوماتان به صورت زیر است.



مرحله دوم: در این مرحله، واحد ۳ جریمه می‌شود و واحد ۶ پاداش می‌گیرد و بقیه واحدها تغییر وضعیت نمی‌دهند. در انتهای این مرحله وضعیت اتوماتان به صورت زیر می‌باشد.

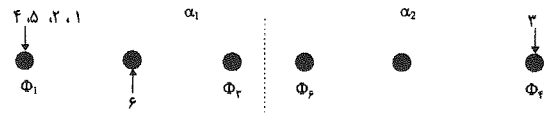


مرحله سوم: دراین مرحله، واحدهای ۳ و ۴ جریمه می‌شوند و واحد ۶ پاداش می‌گیرد و بقیه واحدها تغییر وضعیت نمی‌دهند. بعد از اتمام این مرحله وضعیت اتوماتان به صورت زیر است.

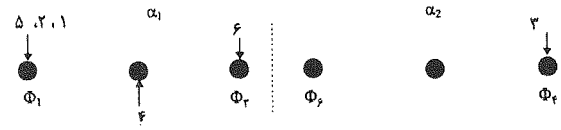


مرحله چهارم: در این مرحله اتوماتان تغییر وضعیت نمی‌دهد.

مرحله پنجم: در این مرحله، واحدهای ۳ و ۶ جریمه می‌شوند و واحد ۴ پاداش می‌گیرد و بقیه واحدها تغییر وضعیت نمی‌دهند. در انتهای این مرحله وضعیت اتوماتان به صورت زیر می‌باشد.



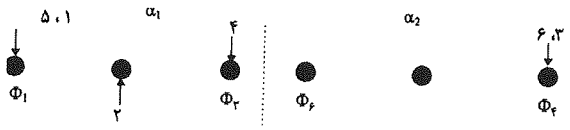
مرحله ششم: در این مرحله، واحدهای ۴ و ۶ جریمه می‌شوند و بقیه واحدها تغییر وضعیت نمی‌دهد. بعد از اتمام این مرحله وضعیت اتوماتان به صورت زیر است.



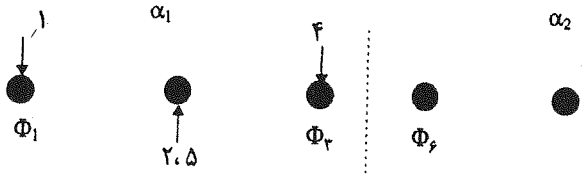
جدول (۱) واریانس فعالیت واحدها.

واریانس فعالیت واحدها						میانگین		پهنا	
۱	۲	۳	۴	۵	۶	M_{ON}	M_{OFF}	X_{ON}	X_{OFF}
۰/۶	۰/۴	۰/۳	۰/۲	۰/۳	۰/۸	۰/۴۳	۰	۰/۲	۰
۰/۶	۰/۳	۰/۲	۰/۳	۰/۴	۰/۶	۰/۳۸	۰	۰/۱۵	۰
۰/۳	۰/۲	۰/۲	۰/۳	۰/۲	۰/۳	۰/۲۵	۰	۰/۰۷	۰
۰/۵	۰/۶	۰/۲	۰/۷	۰/۳۵	۰/۲۵	۰/۴۳	۰	۰۱۷.	۰
۰/۷	۰/۷	۰/۱۵	۰/۳	۰/۷	۰/۲	۰/۵۲	۰/۱۵	۰/۲۱	۰/۰۶
۰/۷	۰/۲۵	۰/۱۲	۰/۲۵	۰/۷	۰/۲۵	۰/۴۳	۰/۱۲	۰/۱۷	۰/۰۶
۰/۷	۰/۷	۰/۱	۰/۴	۰/۳	۰/۲	۰/۵۲	۰/۱۵	۰/۱۷	۰/۰۶

مرحله هفتم: در این مرحله، واحدهای ۲ و ۴ و ۶ جریمه می‌شوند و بقیه واحدها تغییر وضعیت نمی‌دهند. در انتهای این مرحله وضعیت اتوماتان به صورت زیر می‌باشد.



مرحله هشتم: در این مرحله، واحد ۵ جریمه می‌شود و بقیه واحدها تغییر وضعیت نمی‌دهند. بعد از اتمام این مرحله وضعیت اتوماتان به صورت زیر است.



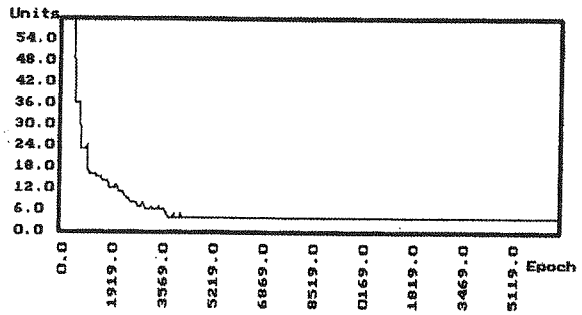
۵ـ نتایج شبیه سازی

برای نشان دادن کارایی های الگوریتم پیشنهاد شده، این الگوریتم روی پنج مسئله Parity سه بیتی، تشخیص اعداد لاتین، Encoding، تقارن و XOR که تعداد تقریبی واحدهای مخفی برای آنها مشخص است، آزمایش شده است. تعدادی از این آزمایشات در این بخش آورده شده است. برای تمامی این مسائل از اتوماتان HULA (2, 7, 60) استفاده شده است. لازم به ذکر است که برای آموزش شبکه ها از الگوریتم انتشار خطا به عقب استاندارد استفاده شده است.

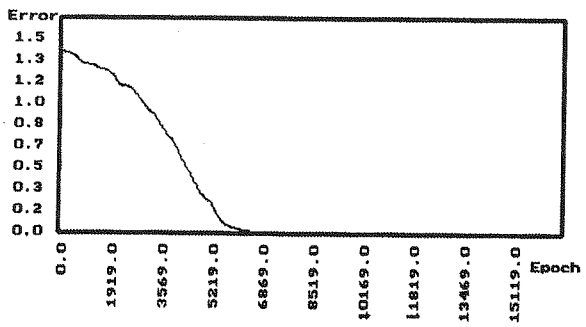
مسئله Parity سه بیتی: نشان داده شده است که برای تولید Parity الگوهای n بیتی، یک شبکه عصبی سه لایه که توسط الگوریتم انتشار خطا به عقب آموزش می‌بیند، نیاز به n واحد مخفی دارد [۲۹]. البته شبکه های با ساختارهای دیگر می‌توانند این مسئله را با دو واحد مخفی حل نمایند [۲۳]. الگوریتم بقا روی ۲۰ شبکه با وزن های اولیه تصادفی برای این مسئله آزمایش شده است و نتایج آن در جدول ۲ آمده است. لازم به ذکر است که در تمامی موارد آزمایش شده، الگوریتم بقا قادر به آموزش شبکه بوده است. شکل های ۱۶ و ۱۷ منحنی تغییرات تعداد واحدهای مخفی و میانگین مربع خطا براساس تکرار ارائه الگو (Epoch) به شبکه برای یک شبیه سازی نمونه را نشان می‌دهند.

جدول (۲) نتایج شبیه سازی برای مسئله Parity سه بیتی.

شبکه	تعداد واحدهای مخفی	متوسط مربع خطا	درصد تشخیص
۱	۵	۰/۰۰۴۸۵۹	۱۰۰
۲	۴	۰/۰۰۴۹۱۹	۱۰۰
۳	۳	۰/۰۰۴۹۷۹	۱۰۰
۴	۳	۰/۰۰۴۹۸۴	۱۰۰
۵	۵	۰/۰۰۴۷۰۸	۱۰۰
۶	۴	۰/۰۰۴۸۴۹	۱۰۰
۷	۴	۰/۰۰۴۷۲۱	۱۰۰
۸	۳	۰/۰۰۴۷۷۸	۱۰۰
۹	۴	۰/۰۰۴۸۶۲	۱۰۰
۱۰	۳	۰/۰۰۴۹۹۳	۱۰۰
۱۱	۴	۰/۰۰۴۸۸۷	۱۰۰
۱۲	۴	۰/۰۰۴۸۳۷	۱۰۰
۱۴	۳	۰/۰۰۴۱۵۰	۱۰۰
۱۵	۵	۰/۰۰۴۷۸۸	۱۰۰
۱۶	۳	۰/۰۰۴۹۹۴	۱۰۰
۱۷	۴	۰/۰۰۴۸۹۲	۱۰۰
۱۸	۴	۰/۰۰۴۹۲۰	۱۰۰
۱۹	۴	۰/۰۰۴۷۷۰	۱۰۰
۲۰	۵	۰/۰۰۴۷۹۹	۱۰۰
متوسط	۳/۷۲	۰/۰۰۴۵۸۴	۱۰۰

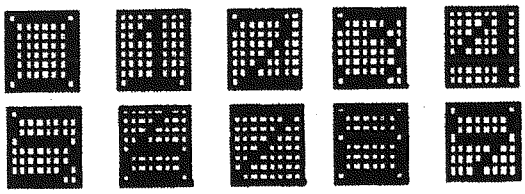


شکل (۱۶) منحنی تغییرات تعداد واحدهای مخفی براساس تکرار ارائه الگو.



شکل (۱۷) منحنی میانگین مربع خطا براساس تکرار ارائه الگو.

مسئله تشخیص اعداد لاتین: در این مسئله ده عدد وجود دارد که هر کدام از آنها توسط یک ماتریس ۸×۸ از نقطه های سیاه و سفید نشان داده می‌شود [۳۴]. الگوهای آموزشی برای این مسئله در شکل (۱۸) نشان داده شده است.

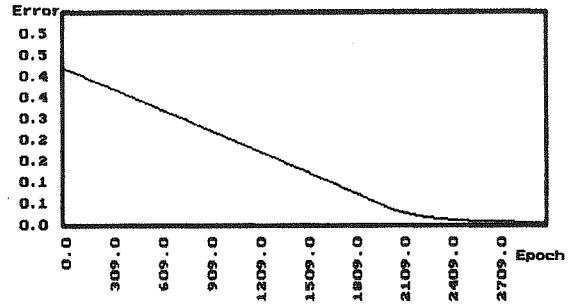


شکل (۱۸) الگوهای آموزشی برای مسئله تشخیص اعداد لاتین.

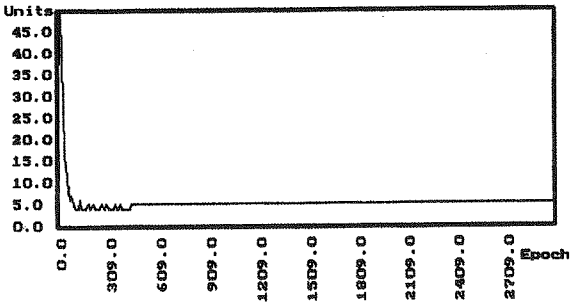
برای حل این مسئله توسط یک شبکه عصبی سه لایه که با الگوریتم انتشار خطا به عقب آموزش می بیند حداقل چهار واحد مخفی در لایه میانی مورد نیاز می‌باشد. این الگوریتم روی ۲۰ شبکه با وزن‌های اولیه تصادفی آزمایش شده‌اند و نتایج آن در جدول (۳) نشان داده شده است. لازم به ذکر است که در تمامی موارد آزمایش شده، الگوریتم پیشنهادی قادر به آموزش شبکه بوده است. شکل‌های ۱۹ و ۲۰ منحنی تغییرات تعداد واحدهای مخفی و میانگین مربع خطا براساس تکرار ارائه الگو (Epoch) به شبکه با استفاده از HULA (2, 7, 50) برای یک شبیه سازی نمونه را نشان می‌دهند.

جدول (۳) نتایج شبیه سازی برای مسئله اعداد لاتین.

شبکه	تعداد واحدهای مخفی	متوسط مربع خطا	درصد تشخیص
۱	۵	۰/۰۴۹۲۷	۹۰
۲	۵	۰/۰۴۹۴۹	۹۰
۳	۵	۰/۰۴۸۵۸	۹۰
۴	۵	۰/۰۴۹۸۲	۹۰
۵	۵	۰/۰۴۶۲۸	۹۰
۶	۵	۰/۰۴۹۲۶	۹۰
۷	۵	۰/۰۴۸۰۹	۹۰
۸	۵	۰/۰۴۹۰۸	۹۰
۹	۵	۰/۰۴۹۱۹	۹۰
۱۰	۵	۰/۰۴۹۴۷	۹۰
۱۱	۵	۰/۰۴۸۶۰	۹۰
۱۲	۵	۰/۰۴۹۹۰	۹۰
۱۴	۵	۰/۰۴۸۷۰	۹۰
۱۵	۵	۰/۰۴۹۳۸	۹۰
۱۶	۵	۰/۰۰۴۹۵	۹۰
۱۷	۵	۰/۰۰۴۹۷	۹۰
۱۸	۴	۰/۰۰۴۹۷	۹۰
۱۹	۶	۰/۰۰۴۹۷	۹۰
۲۰	۵	۰/۰۰۴۸۳	۹۰
متوسط	۵	۰/۰۳۸۰	۹۰

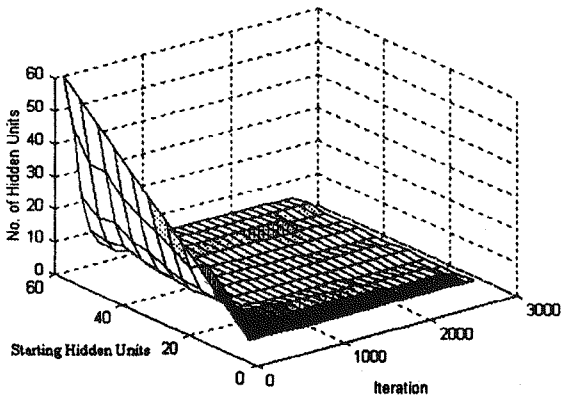


شکل (۱۹) منحنی تغییرات تعداد واحدهای مخفی براساس تکرار ارائه الگو.



شکل (۲۰) منحنی میانگین مربع خطا براساس تکرار ارائه الگو.

شکل ۲۱ تعداد واحدهای مخفی تعیین شده براساس تعداد واحد های استفاده شده در شروع آموزش را نشان می‌دهد. این منحنی بیانگر این مطلب است که تعداد واحدهای تعیین شده توسط الگوریتم مستقل از تعداد واحدهایی است که در شروع آموزش شبکه در نظر گرفته می‌شود. هر چند ساختارهای تولید شده اختلاف جز ئی با هم دارند، اما همه بسیار نزدیک به ساختار بهینه هستند. هر نقطه از این شکل متوسط ۱۰ اجرای مختلف می‌باشد.

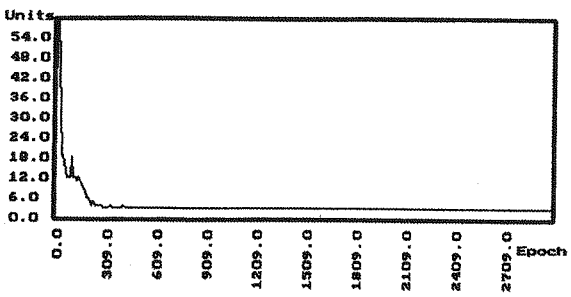


شکل (۲۱) منحنی تعداد واحدهای مخفی تعیین شده براساس تعداد واحدهای اولیه.

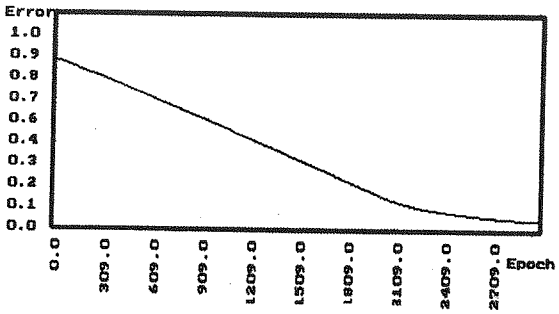
مسئله Encoding: حل مسئله Encoding برای بردارهای n بیتی ورودی، به وسیله یک شبکه عصبی سه لایه که توسط الگوریتم انتشار خطا به عقب آموزش می‌بیند، نیاز به $\log_2^n$ واحد مخفی دارد [۲۹]. این الگوریتم روی ۲۰ شبکه با وزن‌های اولیه تصادفی برای بردارهای هشت بیتی ورودی آزمایش شده‌اند و نتایج آن در جدول (۴) نشان داده شده است. در تمامی موارد آزمایش شده، الگوریتم قادر به آموزش شبکه بوده است. شکل‌های ۲۲ و ۲۳ منحنی تغییرات تعداد واحدهای مخفی و میانگین مربع خطا براساس تکرار ارائه الگو (Epoch) به شبکه برای یک شبیه‌سازی نمونه را نشان می‌دهند.

جدول (۴) نتایج شبیه سازی برای مسئله Encoding

شبکه	تعداد واحدهای مخفی	متوسط مربع خطا	درصد تشخیص
۱	۳	۰/۰۲۱۹۳	۱۰۰
۲	۳	۰/۰۰۹۹۸	۱۰۰
۳	۴	۰/۰۵۴۰۵	۱۰۰
۴	۳	۰/۰۰۹۹۳	۱۰۰
۵	۴	۰/۰۰۹۹۶	۱۰۰
۶	۳	۰/۰۰۹۹۵	۱۰۰
۷	۳	۰/۰۲۱۹۰	۱۰۰
۸	۳	۰/۰۰۹۹۸	۱۰۰
۹	۳	۰/۰۰۹۹۸	۱۰۰
۱۰	۳	۰/۰۰۹۹۹	۱۰۰
۱۱	۳	۰/۰۰۹۹۹	۱۰۰
۱۲	۳	۰/۰۰۹۹۸	۱۰۰
۱۴	۳	۰/۰۰۹۹۹	۱۰۰
۱۵	۳	۰/۰۰۹۹۸	۱۰۰
۱۶	۳	۰/۰۰۹۹۹	۱۰۰
۱۷	۳	۰/۰۰۹۹۵	۱۰۰
۱۸	۳	۰/۰۰۹۹۸	۱۰۰
۱۹	۲	۰/۱۲۳۶۶	۱۰۰
۲۰	۳	۰/۰۰۹۹۹	۱۰۰
متوسط	۳/۰۵	۰/۰۱۸۵۶	۱۰۰



شکل (۲۲) منحنی تغییرات تعداد واحدهای مخفی براساس تکرار ارائه الگو.



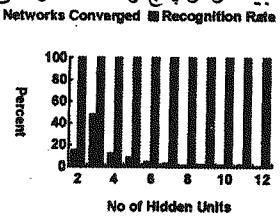
شکل (۲۳) منحنی میانگین مربع خطا براساس تکرار ارائه الگو.

مسئله XOR: برای پیاده‌سازی این تابع توسط شبکه سه لایه، حداقل به دو واحد نیاز می‌باشد [۲۹]. الگوریتم پیشنهادی روی ۱۷۰ شبکه با وزن‌های اولیه تصادفی آزمایش شده است و نتایج آن در جدول (۵) آورده شده است.

جدول (۵) نتایج شبیه سازی برای مسئله XOR.

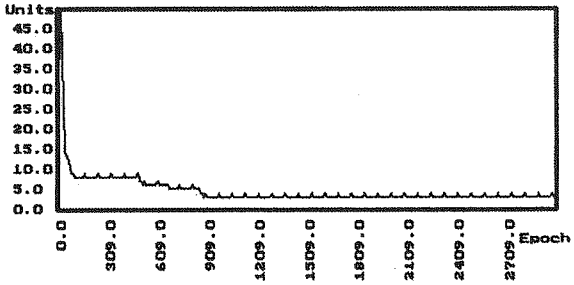
شبکه	تعداد واحدهای مخفی	متوسط مربع خطا	درصد تشخیص
۱-۲۲	۲	۰/۰۳۳۵۸۴	۱۰۰
۲۳-۹۱	۳	۰/۰۰۴۱۳۲	۱۰۰
۹۲-۱۰۹	۴	۰/۰۰۰۴۹۰	۱۰۰
۱۱۰-۱۲۱	۵	۰/۰۰۰۴۸۷	۱۰۰
۱۲۲-۱۲۸	۶	۰/۰۰۰۴۸۰	۱۰۰
۱۲۹-۱۳۳	۷	۰/۰۰۰۴۶۰	۱۰۰
۱۳۴-۱۳۵	۸	۰/۰۰۰۴۴۵	۱۰۰
۱۳۶-۱۳۸	۹	۰/۰۰۰۴۷۷	۱۰۰
۱۳۹-۱۴۱	۱۰	۰/۰۰۰۴۸۵	۱۰۰
۱۴۲-۱۴۴	۱۱	۰/۰۰۰۴۸۴	۱۰۰
۱۴۵	۱۲	۰/۰۰۰۴۸۲	۱۰۰
متوسط	۳/۹۸	۰/۰۰۷۲۶	۱۰۰

از بین ۱۷۰ شبکه آزمایش شده، این الگوریتم قادر به آموزش ۲۵ شبکه نبوده است. تعداد واحدهای مخفی و درصد تشخیص آنها برای مواردی که الگوریتم همگرا شده است، توسط هیستوگرام شکل ۲۴ داده شده است. برای این مسئله ۱۵/۲ درصد از شبکه‌ها به دو واحد مخفی (مقدار بهینه)، ۴۷/۶ درصد به سه واحد، ۱۲ درصد به چهار واحد، ۸/۳ درصد به پنج واحد و کمتر از ۱۷ درصد به بیشتر از پنج واحد همگرا می‌شوند.

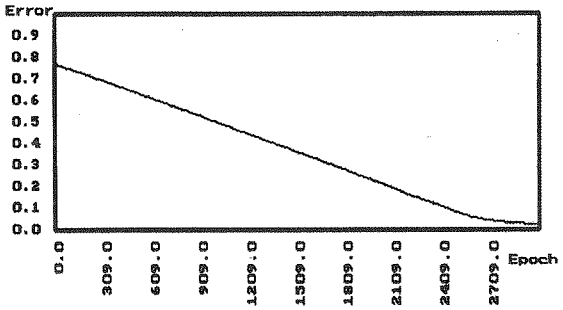


شکل (۲۴) هیستوگرام تعدادو احدهای مخفی تعیین شده توسط الگوریتم بقا.

شکل های ۲۵ و ۲۶ منحنی تغییرات تعداد واحدهای مخفی و میانگین مربع خطا براساس تکرار ارائه الگو (Epoch) به شبکه با استفاده از HULA (2, 7, 50) برای یک شبیه سازی نمونه را نشان می دهند.



شکل (۲۵) منحنی تغییرات تعداد واحدهای مخفی براساس تکرار ارائه الگو.



.شکل (۲۶) منحنی میانگین مربع خطا براساس تکرار ارائه الگو.

۵ ـ مسئله تقارن: برای حل این مسئله توسط یک شبکه عصبی سه لایه و آموزش توسط الگوریتم انتشار خطا به عقب، حداقل دو واحد مخفی مورد نیاز می باشد [۲۹]. دو گروه آزمایش برای مسئله تقارن انجام گرفته است: تقارن چهار بیتی و تقارن شش بیتی. برای بردارهای چهاربیتی ۶۵ شبکه و برای بردار های شش بیتی ۱۲ شبکه توسط الگوریتم پیشنهادی آموزش داده شده اند که نتایج آن در جداول (۶) و (۷) آمده است.

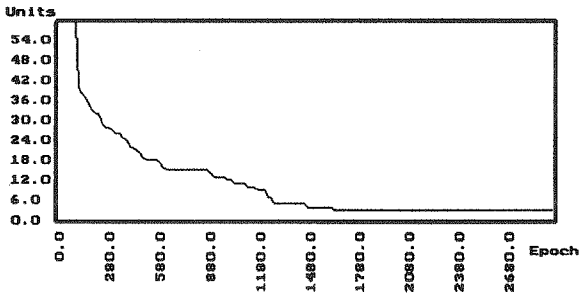
جدول (۶) نتایج شبیه سازی برای مسئله تقارن چهار بیتی.

شبکه	تعداد واحدهای مخفی	متوسط مربع خطا	درصد تشخیص
۱-۹	۲	۰/۰۱۳۱۷۶	۱۰۰
۱۰-۱۸	۳	۰/۰۵۶۷۱۸	۱۰۰
۱۹-۲۷	۴	۰/۰۸۰۳۹۴	۱۰۰
۲۸-۴۱	۵	۰/۰۰۰۸۲۲۵	۱۰۰
۴۲-۴۶	۶	۰/۰۰۰۷۹۱۸	۱۰۰
۴۷-۵۰	۷	۰/۰۰۰۹۴۲۵	۱۰۰
متوسط	۴/۱۸	۰/۰۲۷۴۳	۱۰۰

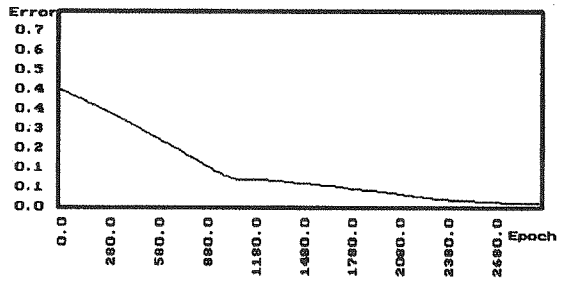
جدول (۷) نتایج شبیه سازی برای مسئله تقارن شش بیتی.

شبکه	تعداد واحدهای مخفی	متوسط مربع خطا	درصد تشخیص
۱	۵	۰/۰۰۰۰۳	۱۰۰
۲	۲	۰/۰۰۱۱۳	۱۰۰
۳	۳	۰/۰۱۰۸۷	۱۰۰
۴	۲	۰/۱۰۵۷۹	۸۷/۵
۵	۲	۰/۰۹۴۴۲	۸۷/۵
۶	۵	۰/۱۰۹۰۹	۸۷/۵
۷	۲	۰/۰۹۳۸۰	۸۷/۵
۸	۵	۰/۱۰۱۷۸	۸۷/۵
۹	۴	۰/۰۹۷۷۹	۸۷/۵
۱۰	۳	۰/۱۰۷۹۳	۸۷/۵
۱۱	۳	۰/۰۹۵۳۵	۸۷/۵
۱۲	۲	۰/۰۹۳۷۵	۸۷/۵
متوسط	۳/۱۶	۰/۰۷۵۹۸	۸۹/۶

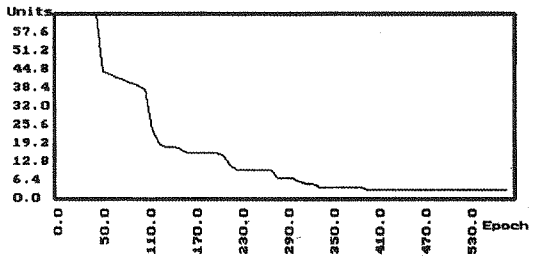
شکل های ۲۷ الی ۳۰ منحنی تغییرات تعداد واحدهای مخفی و میانگین مربع خطا براساس تکرار ارائه الگو به شبکه را برای یک شبیه سازی نمونه نشان می دهند.



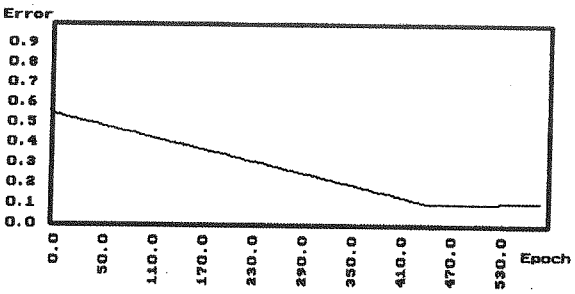
شکل (۲۷) منحنی تغییرات تعداد واحدهای مخفی براساس تکرار ارائه الگو (تقارن چهار بیتی).



شکل (۲۸) منحنی میانگین مربع خطا براساس تکرار ارائه الگو (تقارن چهار بیتی).



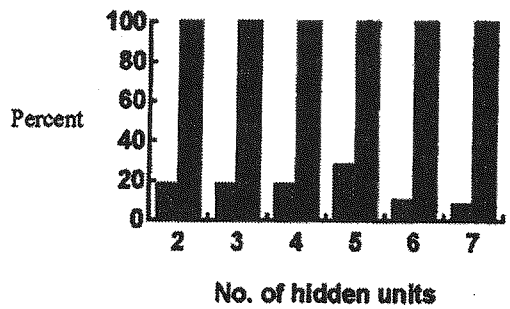
شکل (۲۹) منحنی تغییرات تعداد واحدهای مخفی براساس تکرار ارائه الگو (تقارن شش بیتی).



شکل (۳۰) منحنی میانگین مربع خطا براساس تکرار ارائه الگو (تقارن شش بیتی).

از بین ۶۵ شبکه آزمایش شده برای تقارن چهار بیتی، این الگوریتم قادر به آموزش ۱۵ شبکه نبوده است. تعداد واحدهای مخفی و درصد تشخیص آنها برای مواردی که الگوریتم همگرا شده است ،توسط هیستوگرام شکل ۳۱ داده شده است. در این مسئله ۱۸ درصد از شبکه ها به دو واحد مخفی (مقدار بهینه)، ۱۸ درصد به سه واحد، ۱۸ درصد به چهار واحد، ۲۸ درصد به پنج واحد و کمتر از ۱۰ درصد به شش واحد و ۸ درصد به هفت واحد همگرا می شوند.

لازم به ذکر است که برای مسائل XOR و تقارن، هر چه تعداد واحدهای مخفی به مقدار بهینه نزدیک می شود، سرعت آموزش شبکه کندتر می گردد.



شکل (۳۱) هیستوگرام تعداد واحدهای مخفی تعیین شده توسط الگوریتم بقا.

۶ـ بررسی رفتار اتوماتان یادگیر تعیین تعداد واحدهای مخفی

در این بخش مصلحت گرایی و همگرایی اتوماتان یادگیر تعیین تعداد واحدهای مخفی را در یک محیط ایستا بررسی می کنیم. فرض ایستا بودن محیط که باعث ساده تر شدن تحلیل ریاضی مدل می گردد بدین دلیل در نظر گرفته شده است که با گذشت زمان و عبور از حالت گذرا و نزدیک شدن به حالت ماندگار، تغییرات در شرایط محیط کم می شود و محیط تقریباً همانند یک محیط ایستا عمل می کند. قضیه ۱ مصلحت گرایی اتوماتان HULA (K, N, M) و قضیه ۴ همگرایی اتوماتان HULA (2, N, 1) را اثبات می کند.

لم ۱: ماکزیمم تابع $M = \sum_{n=1}^{K} d_n (1 - d_n)$ با شرط $\sum_{n=1}^{K} d_n = 1$ در نقطه $(d_1, d_2, \ldots, d_K) = (\frac{1}{K}, \frac{1}{K}, \ldots, \frac{1}{K})$ اتفاق می افتد و مقدار آن برابر است با $\frac{K-1}{K}$.

اثبات: تابع M را به صورت زیر در نظر بگیرید.

$$M = f(d_1, d_2, \ldots, d_{K-1}) = \sum_{n=1}^{K-1} d_n (1 - d_n) + d_K (1 - d_K)$$

با توجه به شرط $\sum_{n=1}^{K} d_n = 1$ مقدار تابع $f(d_1, d_2, \ldots, d_{k-1})$ برابر است با

$$f(d_1, d_2, \ldots, d_{K-1}) = \sum_{n=1}^{K-1} d_n (1 - d_n) + d_k (1 - d_k)$$

$$= \sum_{n=1}^{K-1} d_n (1 - d_n) + \sum_{n=1}^{K-1} d_n - \left[\sum_{n=1}^{K-1} d_n\right]^2 = 2 \sum_{n=1}^{K-1} d_n - \left[\sum_{n=1}^{K-1} d_n^2 + \left(\sum_{n=1}^{K-1} d_n\right)^2\right]$$

ماکزیمم تابع $f(d_1, d_2, \ldots, d_{k-1})$ با مساوی صفر قرار دادن مشتقات جزئی آن نسبت به $d_1, d_2, \ldots, d_{k-1}$ بدست می آید.

$$\frac{\partial f}{\partial d_n} = 2 - 2d_n - 2 \sum_{m=1}^{K-1} d_m = 0 \qquad \text{for } n = 1, 2, \ldots, K-1$$

دستگاه معادلات فوق را به فرم ماتریسی می توان به صورت زیر نشان داد.

$$\begin{bmatrix} 2 & 1 & 1 & \cdots & 1 \\ 1 & 2 & 1 & \cdots & 1 \\ 1 & 1 & 2 & \cdots & 1 \\ \vdots & & & & \\ 1 & 1 & 1 & \cdots & 2 \end{bmatrix} \begin{bmatrix} d_1 \\ d_2 \\ d_3 \\ \vdots \\ d_{K-1} \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \\ \vdots \\ 1 \end{bmatrix}$$

باتوجه به ماتریس ضرایب، دستگاه معادلات فوق یک جواب منحصر بفرد $(d_1, d_2, \ldots, d_{K-1}) = (\frac{1}{K}, \frac{1}{K}, \ldots, \frac{1}{K})$ دارد [۴۳]. تابع $f(d_1, d_2, \ldots, d_{K-1})$ در این نقطه

دارای حداکثر مقدار می باشد، زیرا مشتق دوم این تابع در تمامی محورها کوچکتر از صفر می باشد. باتوجه به شرط $\sum_{n=1}^{K} d_n = 1$ مقدار تابع $f(d_1, d_2, ..., d_{K-1})$ در این نقطه برابر است با

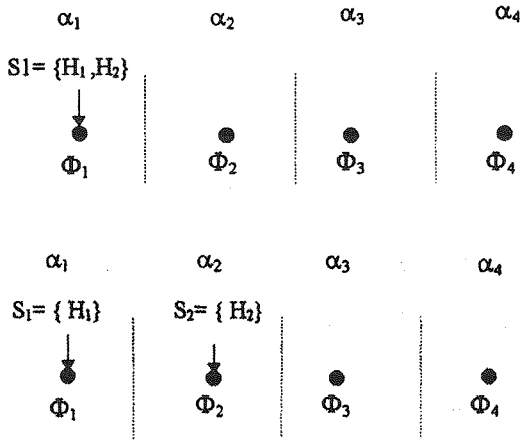
$$f(d_1, d_2, ..., d_{K-1}) = \sum_{n=1}^{K-1} \frac{1}{K}\left(1 - \frac{1}{K}\right) + \frac{1}{K}\left(1 - \frac{1}{K}\right)$$

$$= \sum_{n=1}^{K} \frac{1}{K}\left(1 - \frac{1}{K}\right) = \frac{K-1}{K}$$

بنابر این لم اثبات می گردد.

قضیه ۱: اتوماتان HULA (K, N, M) مصلحت گرا است.

اثبات: فرض کنید مجموعه $W = \{w_1, w_2, ...\}$ نشان دهنده وضعیت های زنجیره مارکوف توصیف کننده اتوماتان HULA (K, N, M) باشند. دو نمونه از این وضعیت ها برای اتوماتان HULA (4, 1, 2) در شکل (۳۲) نشان داده شده اند.



الف ـ وضعیت w_1 در زنجیره مارکوف توصیف کننده اتوماتان HULA (4, 1, 2)

ب ـ وضعیت w_2 در زنجیره مارکوف توصیف کننده اتوماتان HULA (4, 1, 2)

شکل (۳۲) دو وضعیت نمونه در زنجیره مارکوف توصیف کننده اتوماتان HULA (4, 1, 2).

حالت های مختلف زنجیره مارکوف توصیف کننده این اتوماتان در جدول ۸ آمده است. به سادگی می توان نشان داد که زنجیره مارکوف توصیه کننده این اتوماتان دارای $k = (KN)^M$ وضعیت مختلف $w_1, w_2, ..., w_k$ می باشد.

جدول (۸) وضعیت های مختلف زنجیره مارکوف توصیف کننده اتوماتان HULA (4, 1, 2).

اقدام اتوماتان				
اقدام ۱	اقدام ۲	اقدام ۳	اقدام ٤	وضعیت زنجیره مارکوف
$\{H_1, H_2\}$	Φ	Φ	Φ	w_1
$\{H_1\}$	$\{H_2\}$	Φ	Φ	w_2
$\{H_2\}$	$\{H_1\}$	Φ	Φ	w_3
Φ	$\{H_1, H_2\}$	Φ	Φ	w_4
Φ	Φ	$\{H_1, H_2\}$	Φ	w_5
Φ	Φ	$\{H_1\}$	$\{H_2\}$	w_6
Φ	Φ	$\{H_2\}$	$\{H_1\}$	w_7
Φ	Φ	Φ	$\{H_1, H_2\}$	w_8
$\{H_1\}$	Φ	$\{H_2\}$	Φ	w_9
$\{H_1\}$	Φ	Φ	$\{H_2\}$	w_{10}
$\{H_2\}$	Φ	$\{H_1\}$	Φ	w_{11}
$\{H_2\}$	Φ	Φ	$\{H_1\}$	w_{12}
Φ	$\{H_1\}$	$\{H_2\}$	Φ	w_{13}
Φ	$\{H_1\}$	Φ	$\{H_2\}$	w_{14}
Φ	$\{H_2\}$	$\{H_1\}$	Φ	w_{15}
Φ	$\{H_2\}$	Φ	$\{H_1\}$	w_{16}

احتمال انتخاب وضعیت d_n, w_n می باشد که به صورت زیر تعریف می شود.

$$d_1 = Pr[S_1 = \{H_1, ..., H_M\}] + Pr[S_2 = \emptyset] + \cdots + Pr[S_k = \emptyset]$$

$$d_2 = Pr[S_1 = \{H_2, ..., H_M\}] + Pr[S_2 = \{H_1\}] + \cdots + Pr[S_k = \emptyset]$$

$$\vdots$$

$$d_k = Pr[S_1 = \emptyset] + Pr[S_2 = \emptyset] + \cdots + Pr[S_k = \{H_1, ..., H_M\}]$$

که در آن S_j نشاندهنده مجموعه واحدهای مخفی است که در اقدام j اتوماتان HULA (K, N, M) یا در وضعیت w_j زنجیره مارکوف توصیف کننده این اتوماتان قرار دارند. ماتریس انتقال این زنجیره به صورت زیر می باشد.

$$F = \begin{bmatrix} 1 - \sum_{n=2}^{k} d_n & d_2 & \cdots & d_k \\ d_1 & 1 - d_1 - \sum_{n=2}^{k} d_n & & d_k \\ \vdots & & & \\ d_1 & d_2 & & 1 - \sum_{n=1}^{k-1} d_n \end{bmatrix}$$

چون زنجیره مارکوف توصیف کننده این اتوماتان یک زنجیره Ergodic می‌باشد، در حالت تعادل معادله $F^T \Pi = \Pi$ برقرار می‌باشد، که $\Pi = [\pi_1, ..., \pi_k]^T$ و π_n، احتمال قرار گرفتن زنجیره در وضعیت w_n می‌باشد. با بسط معادله $F^T\Pi = \Pi$، دستگاه معادلات زیر بدست می‌آید.

$$-(1 - d_1)\pi_1 + d_1 \times \sum_{n=2}^{K} \pi_n = 0$$

$$d_2 \pi_1 - (1 - d_2)\pi_2 + d_2 \times \sum_{n=3}^{K} \pi_n = 0$$

$$\vdots$$

$$d_m \times \sum_{n=1}^{m-1} \pi_n - (1 - d_m)\pi_m + d_m \times \sum_{n=m+1}^{K} \pi_n = 0$$

با بدست آوردن مقدار π_1 از معادله اول و قرار دادن آن در معادله دوم و ادامه دادن این روش جواب عمومی زیر بدست می‌آید.

$$\pi_m = \frac{d_m}{1 - \sum_{n=1}^{m} d_n} \sum_{n=m+1}^{K} d_n \qquad m = 1, 2, ..., K - 1$$

باتوجه به اینکه شرط $\sum_{n=1}^{K} \pi_n = 1$ همیشه برقرار است، جواب دستگاه معادلات فوق برای $1 \le n \le K$ برابر است با $\pi_n = d_n$. در نتیجه مقدار متوسط جریمه (M) در حالت تعادل برابر خواهد بود با

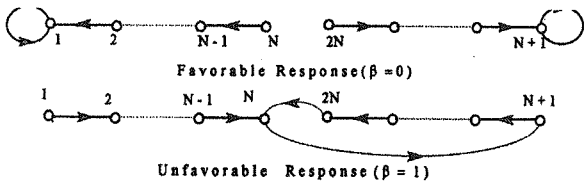
$$M = \sum_{n=1}^{K} c_n \pi_n = \sum_{n=1}^{K} d_n (1 - d_n)$$

و مقدار متوسط جریمه (M_0) برای یک اتوماتان شانس برابر است با

$$M_0 = \sum_{n=1}^{k} \frac{1}{K}\left(1 - \frac{1}{K}\right) = \frac{K - 1}{K}$$

با استفاده از لم (۱) و این حقیقت که در زنجیره مارکوف توصیف کننده اتوماتان HULA (K, N, M) مقادیر d_n ها (برای $1 \le n \le K$) یکسان نیستند شرط $M < M_0$ همیشه برقرار است و بنابراین اتوماتان HULA (K, N, M) مصلحت گرا می‌باشد.

عملکرد اتوماتان HULA (2, N, M) را می‌توان به صورت زیر نیز بیان نمود. هر یک از M واحد مخفی در اتوماتان HULA (2, N, M) به یکی از دو اقدام این اتوماتان وابسته می‌شوند. بنابر این می‌توان به جای یک اتوماتان HULA (2, N, M) اتوماتان HULA (2, N, 1) که گراف تغییر وضعیت آن در شکل (۳۳) نشان داده شده است را در نظر گرفت که در هر کدام از آنها یک واحد مخفی قرار دارد. هر یک از این M اتوماتان یک اتوماتان با ساختار ثابت می‌باشد که اقدام گرفته شده توسط آنها نشان دهنده روشن یا خاموش بودن یک واحد است. این M اتوماتان در یک بازی به منظور تعیین تعداد واحدهای مخفی شرکت می‌کنند. در این بازی، اتوماتان شماره n وضعیت واحد مخفی شماره n را تعیین می‌کند. اگر اتوماتان شماره n در وضعیت Φ_m (شکل ۳۳) قرار داشته باشد این بدان معنی است که واحد مخفی شماره n در وضعیت Φ_m قرار دارد. اگر اتوماتان شماره n در یکی از حالت های $\{\Phi_1, ..., \Phi_N\}$ باشد، در این صورت واحد شماره n روشن واگر در یکی از حالت های $\{\Phi_{N+1}, ..., \Phi_{2N}\}$ باشد، در این صورت واحد شماره n خاموش است. گراف تغییر حالت اتوماتان HULA (2, N, 1) در شکل (۳۳) نشان داده شده است.



شکل (۳۳) اتوماتان HULA (2, N, 1).

قضیه ۱: اگر $\theta = \frac{c_1}{c_2} < 1$ باشد اتوماتون HULA (2, N, 1) یک اتوماتان Optimal - ε است.

اثبات: با توجه به شکل ۳۳ ماتریس های انتقال این اتوماتان برابر است با

$$F(0) = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 1 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 1 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & & & \vdots & & \vdots & & & & \vdots \\ 0 & 0 & \cdots & 1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & & & \vdots & \vdots & & & & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 1 & 0 \end{bmatrix}$$

$$F(1) = \begin{bmatrix} 0 & 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 1 & \cdots & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & & & & \vdots & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 1 & \vdots & & \cdots & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 1 & 0 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 1 & \cdots & 0 & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & & & & 0 & \vdots & & & & \vdots \\ 0 & 0 & \cdots & 0 & 0 & 0 & 0 & \cdots & 0 & 1 \\ 0 & 0 & \cdots & 0 & 1 & 0 & 0 & \cdots & 0 & 0 \end{bmatrix}$$

و ماتریس احتمال انتقال برابر است با

$$P = \begin{bmatrix} d_1 & c_1 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ d_1 & 0 & c_1 & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & d_1 & 0 & \cdots & 0 & 0 & 0 & 0 & 0 & \cdots & 0 & 0 \\ \vdots & & & & & \vdots & \vdots & & & & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & c_1 & 0 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & d_1 & 0 & c_1 & 0 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & d_2 & c_2 & 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & d_2 & 0 & c_2 & \cdots & 0 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & d_2 & 0 & \cdots & 0 & 0 \\ \vdots & & & & & \vdots & \vdots & & & & & \vdots \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 & d_2 & 0 & c_2 \\ 0 & 0 & 0 & \cdots & 0 & c_2 & 0 & \cdots & 0 & 0 & d_2 & 0 \end{bmatrix}$$

عملکرد این اتوماتان در محیط ایستا را می توان توسط یک زنجیره مارکوف توصیف نمود . به دلیل اینکه زنجیره مارکوف توصیف کننده این اتوماتان Ergodic است، بردار احتمال نهایی وضعیت ها که توسط $\Pi = [\pi_1 , \dots , \pi_{2N}]^T$ نمایش داده می‌شود را می توان از رابطه زیر بدست آورد.

$$P^T \Pi = \Pi$$

به دلیل اینکه این اتوماتان دو اقدام دارد، در نتیجه داریم $c_1 = d_2$ و $c_2 = d_1$. با بسط معادله فوق داریم:

$$c_1 \pi_1 + c_2 \pi_2 = \pi_1 \qquad (1)$$
$$\vdots$$
$$c_1 \pi_{k-1} + c_2 \pi_{k+1} = \pi_k \qquad (k)$$
$$\vdots$$
$$c_1 \pi_{N-1} + c_2 \pi_{2N} = \pi_N \qquad (N)$$
$$c_1 \pi_N + c_1 \pi_{N+1} + c_1 \pi_{N+2} = \pi_{N+1} \qquad (N+1)$$
$$\vdots$$
$$c_1 \pi_{N+k-1} + c_2 \pi_{N+k+1} = \pi_{N+k} \qquad (N+k)$$
$$\vdots$$
$$c_2 \pi_{2N-1} = \pi_{2N} \qquad (2N)$$

معادلات شماره (k) و (N + k) برای $k > 1$، معادلات تفاضلی مرتبه دوم هستند و معادلات با شماره های (1) , (N) , (N + 1) و (2N) شرایط مرزی را مشخص می‌کنند. برای حل معادلات فوق فرض می‌کنیم که شکل جواب به صورت زیر باشد.

$$\pi_k = a_1 \lambda_1^{k-1}$$

$$\pi_{N+k} = a_2 \lambda_2^{k-1}$$

با جایگزاری دو معادله فوق در معادلات (k) و (N + k) برای $k > 1$ معادله مشخصه زیر به دست می‌آید:

$$(1 - c_m) \lambda_m^2 - \lambda_m + c_m = 0 \quad m = 1, 2$$

جواب های معادله فوق عبارتند از:

$$\lambda_1^{(1)} = 1$$

$$\lambda_1^{(2)} = \frac{c_1}{c_1} = \theta$$

$$\lambda_1^{(2)} = \frac{c_1}{1 - c_1} = \theta$$

$$\lambda_2^{(2)} = \frac{1 - c_1}{c_1} = \frac{1}{\theta}$$

و شکل جواب معادلات (k) و (N + k) برابر است با

$$\pi_k = A_1 \theta^{k-1} + B_1 \qquad \text{for all } k \geq 1$$
$$\pi_{N+k} = A_2 \theta^{k-1} + B_2 \qquad \text{for all } k \geq 1$$

با استفاده از معادلات شماره (1) و (2N) خواهیم داشت $B_1 = 0$ و $B_2 = -\frac{A_2}{\theta^N}$ و با استفاده از معادله شماره (N) داریم $A_2 = A_1 \frac{\theta^{2N}}{\theta - 1}$. بنابر این جواب معادله $P^T \Pi = \Pi$ برابر است با

$$\pi_k = A_1 \theta^{k-1} \qquad k = 1, 2, \dots, N$$

$$\pi_{N+k} = \frac{A_1}{\theta - 1} \theta^N \left[\theta^{N-k+1} - 1\right] \qquad k = 1, 2, \dots, N$$

در این صورت مقادیر p_1 و p_2 به صورت زیر محاسبه می‌شوند.

$$p_1 = \sum_{k=1}^{N} \pi_k = A_1 \frac{\theta^N - 1}{\theta - 1}$$

$$p_2 = \sum_{k=1}^{N} \pi_{N+k} = \frac{A_2}{\theta^N} \left[\frac{\theta (1 - \theta^N) N (1 - \theta)}{(1 - \theta)}\right]$$

با توجه به مقادیر p_1 و p_2 و اینکه شرط $p_1 + p_2 = 1$ بر قرار است خواهیم داشت.

$$A_1 = \frac{(\theta - 1)^2}{(\theta - 1)(\theta^N - 1) + N\theta^N (1 - \theta) - \theta^{N+1} (1 - \theta^N)}$$

$$A_2 = \frac{\theta^{2N} (\theta - 1)}{(\theta - 1)(\theta^N - 1) + N\theta^N (1 - \theta) - \theta^{N+1} (1 - \theta^N)}$$

در این صورت

$$P_1 = \frac{(\theta^N - 1)(\theta - 1)}{(\theta - 1)(\theta^N - 1) + N\theta^N(1 - \theta) - \theta^{N+1}(1 - \theta^N)}$$

$$P_2 = \frac{\theta^N[N(1 - \theta) - \theta(1 - \theta^N)]}{(\theta - 1)(\theta^N - 1) + N\theta^N(1 - \theta) - \theta^{N+1}(1 - \theta^N)}$$

در نتیجه مقدار متوسط جریمه برای اتوماتان HULA (2, N, 1) برابر است با

$$M_{HULA\,(2,N,1)} = c_1 p_1 + c_2 p_2$$

$$= c_1 p_1 + (1 - c_1) p_2$$

$$= \frac{\theta}{1+\theta} p_1 + \frac{\theta}{1+\theta} p_2$$

و یا

$$M_{HULA(2,N,1)} = \frac{1}{1+\theta} \frac{\theta(\theta - 1)(\theta^N - 1) + N\theta^N(1 - \theta) - \theta^{N+1}(1 - \theta^N)}{(\theta - 1)(\theta^N - 1) + N\theta^N(1 - \theta) - \theta^{N+1}(1 - \theta^N)}$$

اگر θ کوچکتر از یک باشد در این صورت

$$\lim_{N \to \infty} M_{HULA\,(2,N,1)} = \frac{\theta}{1+\theta} = \min\{c_1, c_2\} = c_1$$

بنابر این HULA (2, N, 1) یک اتوماتان ε - Optimal است.

قضیه فوق برای اتوماتان HULA (2, N, M) هنوز اثبات نشده است ولی نتایج شبیه سازی های انجام گرفته ε - Optimal بودن این اتوماتان را نشان می دهد.

۷ ـ نتیجه گیری

در این مقاله الگوریتمی برای تعیین تعداد واحدهای مخفی شبکه عصبی سه لایه به نام الگوریتم بقا که از اتوماتان های مهاجرت اشیاء استفاده می کند، ارائه گردیده است. در این الگوریتم آموزش را از یک شبکه بزرگ شروع نموده و اتومان یادگیر با افزودن و کاستن واحدها سعی در پیدا نمودن ساختار مناسب برای شبکه را دارد. هر چند ممکن است این روش را در گروه الگوریتم های ترکیبی قرار داد، اما در این روش از الگوریتم های سازنده یا الگوریتم های هرس یا ترکیبی از آنها استفاده نشده است، بلکه تعیین ساختار شبکه به صورت مسئله افراز مجموعه ها تعریف شده است. نتایج آزمایشات نشان می دهند که تعداد واحدهای تعیین شده توسط این الگوریتم خیلی نزدیک به مقدار بهینه می باشد و همچنین مستقل از تعداد واحدهایی است که در شروع آموزش شبکه در نظر گرفته می شود. به عنوان نمونه الگوریتم ارائه شده در این مقاله با نتایج حاصله توسط دو الگوریتم هرس محاوره ای [۳۱] و هرس تکراری [۵] برای مسائل Parity و تقارن چهار بیتی که از مقاله [۵] گرفته شده است (جدول ۹ الی ۱۲) مقایسه گردیده است. مقایسه این جداول با جداول داده شده در این مقاله کارایی الگوریتم بقا را نشان می دهد. اغلب روش های گزارش شده برای تعیین ساختار شبکه از الگوریتم های کوهنوردی استفاده می کنند و مشکل گرفتاری در حداقل های محلی را دارند. در روش پیشنهادی در این مقاله به دلیل استفاده از روش های جستجوی عمومی، امکان گرفتاری در حداقل های محلی فضای ساختارها کاهش می یابد.

جدول (۹) نتایج هرس به وسیله روش محاوره ا ی برای مسئله Parity چهار بیتی.

شبکه	تعداد واحدهای مخفی	متوسط مربع خطا	درصد تشخیص
۱	۳	۰/۱۲	۸۷/۵
۲	۵	۰/۰۶	۹۳/۷
۳	۴	۰/۰۶	۹۳/۷
۴	۶	۰/۰۰	۱۰۰
۵	۶	۰/۰۰	۱۰۰
۶	۷	۰/۰۰	۱۰۰
۷	۵	۰/۰۰	۱۰۰
۸	۴	۰/۰۶	۹۳/۷
۹	۶	۰/۰۰	۱۰۰
۱۰	۵	۰/۱۲	۸۸
متوسط	۵/۱	۰/۰۴۴	۹۵/۶۶

جدول (۱۰) نتایج هرس به وسیله روش محاوره ای برای مسئله تقارن.

شبکه	تعداد واحدهای مخفی	متوسط مربع خطا	درصد تشخیص
۱	۶	۰/۰۰	۱۰۰
۲	۹	۰/۰۰	۱۰۰
۳	۶	۰/۰۱	۱۰۰
۴	۸	۰/۰۰	۱۰۰
۵	۷	۰/۰۰	۱۰۰
۶	۴	۰/۰۰	۱۰۰
۷	۶	۰/۱	۱۰۰
۸	۶	۰/۰۲	۸۷/۵
۹	۷	۰/۰۱	۹۳/۷
۱۰	۷	۰/۰۴	۱۰۰
متوسط	۶/۶	۰/۰۱۸	۹۷/۴۹

جدول (۱۱) نتایج هرس به وسیله روش تکراری برای مسئله Parity چهار بیتی.

شبکه	تعداد واحدهای مخفی	متوسط مربع خطا	درصد تشخیص
۱	۵	۰/۰۰	۱۰۰
۲	۵	۰/۰۲	۱۰۰
۳	۴	۰/۰۰	۱۰۰
۴	۵	۰/۰۰	۱۰۰
۵	۵	۰/۰۱	۱۰۰
۶	۵	۰/۰۰	۱۰۰
۷	۵	۰/۰۰	۱۰۰
۸	۵	۰/۰۰	۱۰۰
۹	۵	۰/۰۰	۱۰۰
۱۰	۵	۰/۰۰	۱۰۰
متوسط	۵/۱	۰/۰۰۳	۱۰۰

جدول (۱۲) نتایج هرس به وسیله روش تکراری برای مسئله تقارن.

شبکه	تعداد واحدهای مخفی	متوسط مربع خطا	درصد تشخیص
۱	۳	۰/۰۱	۱۰۰
۲	۳	۰/۰۱	۱۰۰
۳	۴	۰/۰۰	۱۰۰
۴	۴	۰/۰۰	۱۰۰
۵	۲	۰/۰۲	۱۰۰
۶	۳	۰/۰۰	۱۰۰
۷	۳	۰/۰۲	۱۰۰
۸	۴	۰/۰۱	۱۰۰
۹	۴	۰/۰۲	۱۰۰
۱۰	۵	۰/۰۱	۱۰۰
متوسط	۳/۶	۰/۰۸	۱۰۰

زیرنویس ها

1- Multi-Layer Preceptrons
2- Feedforward Nets
3- Rosenblatt
4- Step Function
5- Activation Function
6- Error Backpropagation
7- Topology
8- Time Complexity
9- Memorization Capability
10- Generalization Capability
11- Interpolation
12- State
13- Pruning Algorithms
14- Constructive Algorithms
15- Evolutionary Algorithms
16- Learning Automata
17- Partitioning
18- Action
19- Fixed Structure Learning Automata (FSLA)
20- Variable Structure Learning Automata (VSLA)
21- Unfavorable
22- Favorable
23- Norms of Behavior
24- Expedient
25- Optimal
26- Process
27- Asynchronous Transfer Mode (ATM)
28- Object Migrating Automata (OMA)
29- Oommen
30- Ma
31- Graph Partitioning
32- Graph Isomorphism
33- Hidden Unit Learning Automata (HULA)

مراجع

[1] Angeline, P. J., Saunders, G. M., and Pollack, J. B. (1994). Evolutionary Algorithm that Construct Recurrent Neural Networks, IEEE Trans. on Neural Networks, Vol. 5, No. 1, pp. 54-65.

[2] Arai, M. (1993). Bounds on the Number of Hidden Units in Binary-Valued Three-Layer Neural Networks, Neural Networks, Vol. 6, pp. 855-860.

[3] Beigy, H. and Meybodi, M. R. (1998). A Fast Method for Determining the Number of Hidden Units in Feedforward Neural Networks, Proc. of CSIC-97, Tehran, Iran, pp. 414-420 (In Persian).

[4] Beigy, H. and Meybodi, M. R. (1998). Optimization of Topology of Neural Networks: A Survey, Technical Reports, Computer Eng. Dept. Amirkabir Univeristy (In Persian).

[5] Castellano, G., Fanelli, A. M., and Pelillo, M. (1997). An Iterative Pruning Algorithm for Feedforward Neural Networks, IEEE Trans. on Neural Networks, Vol. 8, No. 3, pp. 519-531.

[6] Fahlman, S. E. and Lebier, C. (1990). The Cas-

cade-Correlation Learning Architecture, Advances in Neural Information Processing System, Vol. II, pp. 524-532.

[7] Frean, M. (1990). The Upstart: A Method for Constructing and Training Feedforward Neural Networks, Neural Computation, pp. 198-29.

[8] Hashim, A. A., Amir, S., and Mars, p. (1986). Application of Learning Auromata to Data Compression, In Adaptive and Learning Systems, K. S. Narendra (Ed)., New York: Plenum Press, pp. 229-234.

[9] Hirose, Y., Yamashita, K., and Hijya, S. (1991). Back-Propagation Algorithm Which Varies the Number of Hidden Units, Neural Networks, Vol. 4, No. 1, pp. 61-66.

[10] Huang, S. C. and Huang, Y. F. (1991), Bounds on the Number Hidden Neurons in Multilayer Preceptrons, IEEE Trans. on Neural Networks, Vol. 2, No. 1, pp. 47-56.

[11] Kruschke, J. H. (1988), Creating Local and Distributed Bottlenecks in Hidden Layer of Backpropagation Networks, Proc. of Connectionist Models, Summer School, Eds. D. Tourestzky, G. Hinton, and T. Sejnowski, pp. 120-126.

[12] Kruschke, J. h. (1989), Improving Generalization in Backpropagation Networks, Proc. of Int. Joint Conf. on Neural Networks, Vol. i, pp. 443-447.

[13] Kwok, T. Y. & Yeung, D. Y. (1997). Constructive Algorithms for Structure Learning in Feedforward Neural Networks for Regression Problems, IEEE Trans. on Neural Networks, Vol. I, No. pp. 630-645.

[14] Lin, J. H. & Vitter, J. S. (1991). Complexity Results on Learning by Neural Nets, Machine Learning, Vol. 6, pp. 211-230.

[15] Maniezzo, V. (1994). Genetic Evolution of the Topology and Weight Distribution of Neural Networks, IEEE Trans on Neural Networks, Vol. 5, No. 1, pp. 39-53.

[16] Marchand, M., Golea, M., and Rujan, R (1990). A Convergence Theorem for Sequential Learning in Two-Layer Perceptrons, Europhysics Letters 11, pp. 487-492.

[17] Mars, P., chen, J., R., and Nambiar, R. (1998). Learning Algorithms: Theory and Application in Signal Processing Control, and Communications, CRC press, New York.

[18] Mars. P. and Narendra. K. S., and Chrystall, M. (1983). Learning Automata Control of Computer Communication Networks Proc. of Third Yale Workshop on Applications of Adaptive Sytems Theory, Yale University.

[19] Meltser , M., Shoham, M., and Manevitz, L. M. (1996). Approximating Function by Neural Networks: A Constructive Solution in the Uniform Norm, Neural Networks, Vol 9, No. 6, pp. 965-978.

[20] Meybodi, m. R. and Beigy , H. (1998). New Class of Learning Automata Based Scheme for Adaptation of Backpropagation Algorithm Parameters, Proc. of EUFIT-98, Achen, Germany, pp. 339-344.

[21] Meybodi. M. R. and Lakshmivarhan, S. (1983). A Learning Approach to Priority Assignment in a Two Class M/M/1 Queuing System with Unknown Parameters, Proc. of Third Yale Workshop on Applications of Adaptive Systems Theory, Yale University, pp. 106-109.

[22] Mezard, M. and Nadal, J. P. (1989). Learning in Feedforward Neural Networks: The Tiling Algorithm, Journal of Physics, pp. 1285-1296.

[23] Minor, J. M. (1993). Parity With Two Layer Feedforward Nets, Neural Networks, Vol. 6, No. 5, pp. 705-707.

[24] Nabban, T. m. and Zomaya, A. Y. (1994). Toward Neural Networks Structures for Function Approximation, Neural Networks, Vol. 7, No. 1, pp. 89-99.

[25] Narendra, k. S. and Thathachar, M. A. L. (1989). Learning Automata: An Introduction, Prentice-hall, Englewood cliffs.

[26] Oommen,. B. J. and Ma, D. C. Y. (1988). Deterministic Learning Automata Solutions to the Equipartitioning Problem, IEEE Trans. on Computers, Vol. 37, No. 1, pp. 2-13.

[27] Oommen, B. J., Valiveti, R. S., and Zgierski, J. R. (1991). An Adaptive Learning Solution to the Keyboard Optimization Problem, IEEE Trans. on Systems, Man, and Cybernetics, Vol. 21, No. 6, pp. 1608-1618.

[28] Oommen, B. J. and Croix, E. (1996). Graph Partitioning Using Learning Automata, IEEE Trans. on Computers, Vol. 45, No. 2, pp. 195-208.

[29] Rumelhart, D. E., Jinton, G. E., and Williams, R. J. (1986). Learning Internal Representations by Error Backpropagation, In Parallel Distributed Processing, Cambridge, MA: MIT Press.

[30] Reed, R. (1993). Pruning Algorithms-A Survey, IEEE Trans. on Neural Networks, Vol. 4, No. 5, pp. 740-747.

[31] Sietsma, J. and Dow, R. J. F. (1991). Creating Artificial Neural Networks that Generalize, Neural Networks, Vol. 4, No. 1, pp. 67-79.

[32] Sirat, J. A. and Nadal, J. P. (1990), Neural Trees: A New Tool for Classification, Preprint, Laboratories d'Electronique, Philips, Limeil Brevannes, France.

[33] Schaffer, J. D., Whitely, D., and Eshelman, L. J. (1992). Combinations of Genetic Algorithms and Neural Networks: A Survey of the State of the Art, IEEE Proc. COGANN-92, pp. 1-37.

[34] Sperduti, A. and Startia, A. (1993). Speed Up Learning and Network Optimization with Extended Backpropagation, Neural Networks, Vol. 6, pp. 365-383.

[35] Tamura, S. and Tateishi, M. (1997). Capabilities of a Four-Layered Feedforward Neural Network: Four Layer Versus Three, IEEE Trans. on Neural Networks, Vol. 8, No. 2, pp. 251-255.

[36] Thathachar, M. A. L. and Sastry, P. S. (1987). Learning Optimal Discriminant Functions Through a Cooperative Game of Automata, IEEE Trans. Syst., Man and Cybern., Vol. SMC-27, pp. 73-85.

[37] Whitley, D. and Bogart, C. (1990). The Evolution of Connectivity: Pruning Neural Networks Using Genetic Algorithms, Proc. of Int. Joint Conf. on Neural Networks, Vol. I, pp. 134.

[38] Yao, X. and Liu, Y. (1997). A New Evolutionary System for Evolving Artificial Neural Networks, IEEE Trans. on Neural Networks, Vol. 8, No. 3, pp. 694-713.

[39] Yeung, D. Y. (1991). Automatic Determination of Network Size for Supervised Learning, IEEE Int. joint Conf. on Neural Networks, pp. 158-164.

[40] Yu, X. H. (1992). Can Backpropagation Error Surface Not Have Local Minima, IEEE Trans. on Neural Networks, Vol. 3, No. 6, pp. 1019-1021.

[41] Lippman, R. P. (1987). An Introduction to Computing with Neural Nets, IEEE ASSP Mag., Vol. 4, pp. 4-22.

[42] Beigy, H. and Meybodi, M. R. (1999). Graph Isomorphism Using Learning Automata, In proc. of 5th Annual Int. CSI Computer Conf., CSICC' 2000, pp. 402-415, March. 2000.

[43] Gilbert, J. and Gilbert, L. (1994), Linear Algebra and Matrix Theory, Academic Press, New York.

[44] Meybodi, M. R. and Lakshmivarahan S. (1982), Optimality of a General Class of Learning Algorithm, Information Science, Vol. 28, pp. 1-20.

[45] Meybodi, M. R. and Lakshmivarahan S. (1984), On a Class of Learning Algorithms which Have a Symmetric Behavior Under Success and Failure, Springer Verlag Lecture Notes in Statistics, pp. 145-155.

[46] Meybodi, M. R. (1987), Results on Strongly Absolutely Expedient Learning Automata, Proc. of OU Inference Conf. 86, ed. D. R. Mootes and R. Butrick, Athens, Ohio: Ohio University Press, pp. 197-204.