بهینه سازی ساختار شبکه های عصبی توسط اتوماتانهای یادگیر

حمید بیگی محمدرضا میبدی

دانشکده مهندسی کامپیوتر
دانشگاه صنعتی امیر کبیر
تهران، ایران

چکیده

ثابت بودن ساختار شبکه در حین یادگیری، یکی از مشکلات آموزش شبکه های عصبی است. طراح شبکه ابتدا ساختار مناسب شبکه را مشخص میکند و سپس شبکه را آموزش میدهد. جستجوی کامل در فضای ساختارها ، تنها راه پیدا نمودن ساختار بهینه است و هزینه بالایی را در بر دارد. برای کاهش هزینه جستجو، روشهای تقریبی از جمله، الگوریتم های سازنده، الگوریتم های هرس، الگوریتم های ترکیبی و الگوریتم های تکاملی پیشنهاد شده اند که ساختار نزدیک به بهینه را تولید میکنند. در اغلب این روشها برای پیدا نمودن ساختار شبکه از الگوریتم های کوهنوردی استفاده میشود که مشکل گرفتاری در حداقل های محلی را دارند. در این مقاله از اتوماتان یادگیر که یکی از روشهای جستجوی عمومی است برای تعیین ساختار شبکه استفاده شده است. برای این منظور الگوریتمی بنام الگوریتم بقا معرفی شده است که از یک اتوماتان یادگیر و الگوریتم انتشار خطا به عقب استفاده میکند و در حین آموزش، ساختار مناسب شبکه را تعیین مینماید. در این روش آموزش را با 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 قرارداد زیرا واحد های مخفی هم هرس و هم اضافه میشوند، اما هدف استفاده از الگوریتم های سازنده یا الگوریتم های هرس یا ترکیبی از آنها نیست. در این روش تعیین ساختار شبکه بعنوان مسئله افراز مجموعه ها تعریف شده است. این الگوریتم روی مسائل متنوعی پیاده سازی شده است و نتایج رضایت بخشی بدست آمده است.

کلمات کلیدی: بهینه سازی ساختار شبکه عصبی، آموزش شبکه عصبی، اتوماتان یادگیر

۱- مقدمه

شبکه های عصبی بصورت گسترده درکاربردهای زیادی مورد استفاده قرار گرفته است. بیشتر این کاربرد ها از شبکه های عصبی پیش خور[1] که بوسیله الگوریتم انتشار خطا به عقب[2] آموزش داده می شود استفاده میکنند. کارایی این الگوریتم ، علیرغم کاربردهای موفق آن بمیزان زیادی به ساختار[3] شبکه وابسته است. مسئله اصلی در تعیین ساختار شبکه عصبی، تعیین تعداد لایه های مخفی، تعداد واحدهای مخفی در هر لایه و اتصالات بین واحدها میباشد. مسئله طراحی یک شبکه با ساختار بهینه یک مسئله NP-Hard است[۱۳] و بهمین جهت بیشتر الگوریتم های تعیین ساختار شبکه، الگوریتم های تقریبی هستند .

یک شبکه عصبی را میتوان توسط یک گراف $N = (V, E)$ نمایش داد که V مجموعه واحدها و E مجموعه وزنهای شبکه میباشد. روشهای بهینه سازی ساختار شبکه از وضعیت[4] موجود شبکه $N_1 = (V_1, E_1)$ بدنبال یافتن وضعیت مناسبتر شبکه $N_2 = (V_2, E_2)$ هستند. الگوریتم بهینه سازی ساختار شبکه را میتوان بعنوان یک نگاشت بصورت $\Delta: N \rightarrow N$ تعریف نمود که از روی وضعیت N_1 وضعیت N_2 را پیدا میکند. فضای جستجو را نیز میتوان بعنوان یک گراف $A = (N, \delta)$ در نظر گرفت که N مجموعه گره ها و δ مجموعه کمانهای این گراف است. هر گره از این گراف یک ساختار از شبکه را نشان میدهد و کمانهای این گراف نمایانگر نگاشت ها هستند. هدف الگوریتم های تعیین ساختار شبکه، پیمایش این گراف بمنظور پیدا کردن مناسبترین گره (شبکه) با حداقل هزینه میباشد.

برای بهینه سازی ساختار شبکه های عصبی، الگوریتم های متعددی پیشنهاد شده اند که قبل یا در حین یا بعد از یادگیری ساختار مناسب برای شبکه را تعیین میکنند. بعضی از این الگوریتم ها از اطلاعات محلی و

1- Feed forward neural networks
2- Error backpropagation algorithm
3- Topology
4- State

بعضی دیگر از اطلاعات عمومی برای یافتن ساختار مناسب استفاده میکنند. این الگوریتم ها را میتوان به چهار گروه زیر تقسیم کرد.

الف) الگوریتم های هرس[1]: این الگوریتم ها از یک شبکه بزرگ شروع نموده و بتدریج در حین آموزش یا بعد از آن واحد ها یا وزنهای اضافی را از شبکه هرس می کنند. در این الگوریتم ها نیاز است که تعداد واحد های مخفی را در ابتدای آموزش مشخص نماییم. البته این مشکل حادی را ایجاد نمیکند زیرا کران بالا برای یک مسئله معین مشخص است[۲][۱۰][۳۵][۴۰]. این الگوریتم ها هم از مزایای شبکه های بزرگ (پیچیدگی آموزش کم و دوری از حداقل های محلی) و هم از مزایای شبکه های کوچک (قدرت تعمیم بالا) بهره میبرند. در این الگوریتم ها برای دو وضعیت متوالی $N_1 = (V_1, E_1)$ و $N_2 = (V_2, E_2)$ دوشرط $V_2 \subseteq V_1$ و $E_2 \subset E_1$ برقرار میباشد [۵][۱۱][۱۲][۳۰][۳۱].

ب) الگوریتم های سازنده[2]: این الگوریتم ها با یک شبکه کوچک شروع به آموزش میکنند و بتدریج در حین آموزش شبکه، واحد یا لایه مخفی به شبکه افزوده می شود. این الگوریتم ها معمولا شبکه های کوچکی تولید می کنند اما پیچدگی آموزش این شبکه ها معمولا بالا است[۱۳]. در این الگوریتم ها 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 $E_1 \subset E_2$ برقرار میباشد[۲][۶][۷][۱۶][۱۹][۲۲][۳۲][۳۹].

ج) الگوریتم های ترکیبی: این گروه از الگوریتم ها، از ترکیبی از الگوریتم های سازنده و الگوریتم های هرس برای تعیین ساختار شبکه استفاده میکنند. در این الگوریتم ها برای رسیدن به شبکه مطلوب میتوان واحد یا لایه را کم یا زیاد نمود[۹][۲۴].

د) الگوریتم های تکاملی[3]: دراین الگوریتم ها طراحی ساختار بهینه شبکه بصورت جستجو در فضای ساختارها در نظر گرفته می شود که هر نقطه از این فضا نماینده یک ساختار شبکه است. الگوریتم جستجو با استفاده از یک معیار کارایی مانند حداقل خطا، سرعت آموزش و پیچیدگی آموزش بدنبال مناسبترین ساختار میباشد[۱][۱۵][۳۳][۳۷][۳۸].

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

در این مقاله روشی بر اساس اتوماتانهای یادگیر[4] برای تعیین ساختار نزدیک به بهینه شبکه عصبی سه لایه (ساختار با حداقل تعداد واحد های 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 الگوریتم های ترکیبی قرار داد زیرا واحد های مخفی هم هرس و هم اضافه میشوند اما هدف استفاده از الگوریتم های سازنده یا الگوریتم های هرس یا ترکیبی از آنها نیست بلکه تعیین ساختار شبکه بعنوان مسئله افراز[5] مجموعه تعریف شده است.

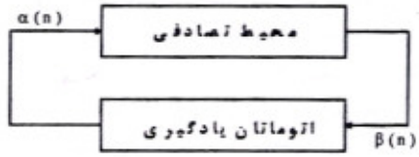
بخش های بعدی مقاله بصورت زیر سازماندهی شده است. مقدمه ای بر اتوماتانهای یادگیر در بخش ۲ آمده است. اتوماتان پیشنهادی و الگوریتم بقا در بخش های ۳ و ۴ بیان شده اند. نتایج آزمایشها برای مسائل مختلف در بخش ۵ آورده شده است ودر پایان نتیجه گیری آمده است.

1- Pruning algorithms
2- Constructive algorithms
3- Evolutionary algorithms
4- Learning automata
5- Partitioning

۲- اتوماتانهای یادگیر

یادگیری در اتوماتانهای یادگیر، انتخاب یک اقدام[6] بهینه از میان یک مجموعه از اقدام های مجاز میباشد. این اقدام روی یک محیط تصادفی اعمال میشود و محیط به این اقدام اتوماتان بوسیله یک پاسخ از مجموعه پاسخ های مجاز جواب میدهد. پاسخ محیط بصورت آماری به اقدام اتوماتان وابسته است. اصطلاح محیط شامل اجتماع تمام شرایط خارجی و تاثیرات آنها روی عملکرد اتوماتان میباشد.

یک محیط بصورت یک سه تایی $\langle \underline{\alpha}, \underline{C}, \underline{\beta} \rangle$ 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 ۱ نشان داده شده است.



شکل ۱: اتصال اتوماتان یادگیری با محیط

اتوماتان های یادگیری به دو خانواده اتوماتان یادگیری با ساختار ثابت[7] و اتوماتان یادگیری با ساختار متغیر[8] دسته بندی میشوند. اتوماتا نهای ***کرینسکی***[9] و ***کرایلوف***[10] مثالهایی از خانواده اتوماتانهای با ساختار ثابت هستند. یک اتوماتان یادگیری با ساختار ثابت را میتوان با یک پنج تایی $\langle \underline{\alpha}, \underline{\Phi}, \underline{\beta}, F, G \rangle$ نشان داد. $\underline{\alpha}=\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ مجموعه اقدام های مجاز برای اتوماتان یادگیر، $\underline{\Phi}=\{\Phi_1, \Phi_2, \ldots, \Phi_s\}$ مجموعه وضعیت های اتوماتان، $\underline{\beta} = \{0, 1\}$ مجموعه ورودیها (در این مجموعه **یک** نمایانگر شکست[11] و **صفر** نمایانگر موفقیت[12] میباشد)، $F: \Phi \times \beta \to \Phi$ تابع نگاشت وضعیت ها و $G: \Phi \to \alpha$ تابع نگاشت خروجی میباشد.

اقدام اتوماتان بعنوان ورودی به محیط داده میشود و محیط بعد از اعمال اقدام داده شده توسط اتوماتان یک پاسخ تصادفی که میتواند موفق یا نا موفق باشد را تولید میکند که بعنوان ورودی به اتوماتان داده میشود. اتوماتان با توجه به پاسخ محیط اقدام مربوطه را جریمه میکند و یا به آن پاداش میدهد. اگر احتمال تغییر وضعیت ها در اتوماتان ثابت باشد آنرا اتوماتان یادگیری با ساختار ثابت و در غیر اینصورت آنرا اتوماتان یادگیری با ساختار متغیر مینامند.

اتوماتانهای یادگیر دارای کاربردهای فراوانی میباشد بعضی از این کاربردها عبارتند از: مسیر یابی در شبکه های ارتباطی[۱۸]، فشرده سازی

6- Action
7- Fixed Structure Learning Automata (FSLA)
8- Variable Structure Learning Automata (VSLA)
9- Krinsky Automata
10- Krylov Automata
11- Unfavorable
12- Favorable

تصاویر[۸]، شناسایی الگو [۳۶]، برنامه ریزی فرایندها[1] در یک شبکه کامپیوتری[۲۵]، تئوری صف [۲۱] ، کنترل دسترسی در شبکه های انتقال ناهمزمان[2] [۲۱]، کمک به آموزش شبکه های عصبی [۲۰] دسته بندی و افراز اشیاء [۲۶] .

در صورتیکه از اتوماتانهای یادگیر برای دسته بندی یا افراز اشیاء استفاده شوند علیرغم سرعت همگرایی خوب دارای تعداد خروجی های بسیار بالا خواهند بود[۲۶]. برای پایین آوردن تعداد خروجی ها، اتوماتان مهاجرت اشیاء[3] توسط **اومن**[4] و **ما**[5] پیشنهاد شده است[۲۶]. تعداد خروجی های این اتوماتان بمراتب کمتر از اتوماتانهای قبلی است ولی در عوض سرعت همگرایی آن پایین میباشد.

اتوماتان مهاجرت اشیاء: اتوماتان مهاجرت اشیاء توسط پنج تایی $\langle \underline{\alpha}, \underline{\Phi}, \underline{\beta}, F, G \rangle$ نشان داده میشود[۲۶]. $\underline{\alpha}=\{\alpha_1, \alpha_2, \ldots, \alpha_r\}$ مجموعه اقدام های مجاز، $\underline{\Phi}=\{\Phi_1, \Phi_2, \ldots, \Phi_s\}$ مجموعه وضعیت های اتوماتان، $\underline{\beta}=\{0, 1\}$ مجموعه ورودیها، $F: \Phi \times \beta \rightarrow \Phi$ تابع نگاشت وضعیت ها و $G: \Phi \rightarrow \alpha$ تابع نگاشت خروجی میباشند. این نوع اتوماتان برای دسته بندی اشیاء[۲۶]، انتساب حروف به کلیدها[۲۷] و افراز گراف[6] [۲۸] مورد استفاده قرار گرفته است.

در این اتوماتان هر اقدام یک دسته را نشان میدهد. در اتوما تا نهای با ساختار ثابت پاسخ محیط به اتوماتان سبب میشود که اتوماتان از یک وضعیت به وضعیت دیگر منتقل شود در صورتیکه در اتوماتان مهاجرت اشیاء، اشیاء به وضعیت ها انتساب داده میشوند و پاسخ محیط به اتوماتان سبب گردش اشیاء در بین وضعیت های اتوماتان میگردد و از طریق این گردش دسته بندی اشیاء انجام میگیرد.

اگر شی W_j درخروجی i ام اتوماتان مهاجرت اشیاء قرار داشته باشد این شی متعلق به دسته شماره i است. برای خروجی α_k مجموعه وضعیت $\{\Phi_{(k-1)N+1}, \ldots, \Phi_{kN}\}$ در نظر گرفته میشود که N عمق حافظه را نشان میدهد. بدون از دست دادن عمومیت بحث میتوان $\Phi_{(k-1)N+1}$ را داخلی ترین وضعیت و Φ_{kN} راخارجی ترین وضعیت این خروجی در نظر گرفت. اگر دو شی W_i و W_j بترتیب در وضعیت های $\Phi_{(k-1)N+1}$ و $\Phi_{(k-1)N+m}$ (برای $m > 1$) قرار داشته باشند در اینصورت قطعیت تعلق شی W_i به این دسته از قطعیت تعلق شی W_j بیشتر است. بنابر این برای خروجی α_k، وضعیت $\Phi_{(k-1)N+1}$، وضعیت با بیشترین قطعیت و وضعیت Φ_{kN}، وضعیت با کمترین قطعیت نامیده میشود.

۳- اتوماتان یادگیر تعیین تعداد واحدهای لایه مخفی[7]

در این قسمت یک اتوماتان از نوع مهاجرت اشیاء برای تعیین تعداد واحد های لایه مخفی یک شبکه سه لایه که توسط الگوریتم انتشار خطا به عقب آموزش داده میشود معرفی میگردد. وظیفه این اتوماتان دسته بندی واحد های لایه مخفی به دو دسته واحد های مناسب و واحد های نا مناسب میباشد. این اتوماتان بصورت یک شش تایی $\langle \underline{\alpha}, \underline{H}, \underline{\Phi}, \underline{\beta}, F, G \rangle$ نشان داده میشود که در آن

۱- $\underline{\alpha}=\{\alpha_1, \alpha_2\}$ مجموعه اقدام های مجاز برای اتوماتان یادگیر است. این اتوماتان دو خروجی دارد که خروجی شماره یک آن خروجی مناسب یا واحد های روشن نام دارد. واحد هایی که در این خروجی قرار دارند برای آموزش شبکه عصبی مورد استفاده قرار میگیرند. خروجی شماره دو آن خروجی نامناسب یا واحد های خاموش نام دارد. واحد هایی که در این خروجی قرار میگیرند برای آموزش شبکه عصبی مورد استفاده قرار نمی گیرند.

۲- $\underline{H}=\{H_1, H_2, \ldots, H_n\}$ مجموعه واحدهای مخفی موجود در خروجی اتوماتان میباشد. اگر واحد H_i در خروجی شماره یک اتوماتان ظاهر شود این واحد بعنوان واحد مخفی مناسب(روشن) و در غیر اینصورت این واحد بعنوان واحد مخفی نامناسب (خاموش) درنظر گرفته میشود.

۳- $\underline{\Phi}=\{\Phi_1, \Phi_2, \ldots, \Phi_{2N}\}$ مجموعه وضعیت ها وN عمق حافظه برای اتوماتان میباشد. مجموعه وضعیت های این اتوماتان به دو زیر مجموعه $\{\Phi_1, \ldots, \Phi_N\}$ و $\{\Phi_{N+1}, \ldots, \Phi_{2N}\}$ افراز میشود و واحد های مخفی براساس اینکه در کدام وضعیت قرار داشته باشند دسته بندی میشوند. براین اساس واحد های روشن با مجموعه $ON = \{H_i \mid 1 \le State(H_i) \le N\}$ و واحد های خاموش با مجموعه $OFF = \{H_i \mid N+1 \le State(H_i) \le 2N\}$ نشان داده میشوند و $State(H_i)$ نشاندهنده وضعیتی است که واحد H_i در آن قرار دارد.

۴- $\underline{\beta}=\{0, 1\}$ مجموعه ورودیهای اتوماتان میباشد. در این مجموعه **یک** نمایانگر شکست و **صفر** نمایانگر موفقیت میباشد.

۵- $F: \Phi \times \beta \rightarrow \Phi$ تابع نگاشت وضعیت ها میباشد. این تابع از روی وضعیت فعلی و ورودی اتوماتان وضعیت بعدی آنرا تولید می نماید. در واقع این تابع چگونگی گردش واحد های مخفی رادر وضعیت های اتوماتان مشخص میکند. شرح کارکرد این تابع در قسمت بعد خواهد آمد.

۶- $G: \Phi \rightarrow \alpha$ تابع نگاشت خروجی میباشد. این تابع تصمیم میگیرد که به ازای هر وضعیت ، اتوماتان چه اقدامی را انجام دهد. اگر واحد H_i متعلق به مجموعه وضعیت های $\{\Phi_1, \Phi_2, \ldots, \Phi_N\}$ باشد این واحد روشن در نظر گرفته میشود. اگر واحد در وضعیت Φ_1 قرار داشته باشد مناسبترین واحد است و بیشترین اهمیت را دارا میباشد. اگر در وضعیت Φ_N قرار داشته باشد دارای کمترین اهمیت میباشد. خروجی شماره دو نیز بهمین صورت میباشد. اگر واحد متعلق به مجموعه وضعیت های $\{\Phi_{N+1}, \ldots, \Phi_{2N}\}$ باشد واحد را خاموش در نظر میگیریم. 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 میباشد.

برای سهولت نمایش در ارائه مطالب، اتوماتان یادگیر تعیین تعداد واحدهای لایه مخفی با K اقدام ، عمق حافظه N و انتساب M واحد مخفی در شروع آموزش توسط HULA (K, N, M) نشان داده میشود.

برای تشریح تابع نگاشت وضعیت ها چهار حالت زیر را در نظر میگیریم.

۱- واحد مخفی H_i روشن است ودر وضعیت Φ_j قرار دارد و بدلیل عملکرد مناسبش پاداش میگیرد. در اینصورت اهمیت این واحد بیشتر شده و بسمت وضعیت های داخلی تر این اقدام حرکت میکند. نحوه حرکت چنین واحدی در شکل زیر نشان داده شده است.

1- Process
2- Asynchronous transfer mode (ATM)
3- Object Migrating Automata (OMA)
4- Oommen
5- Ma
6- Graph Partitioning
7- Hidden unit learning automata (HULA)

الف) وضعیت واحد H_i قبل از دادن پاداش

ب) وضعیت واحد H_i بعد از دادن پاداش

شکل۲ : نحوه دادن پاداش به یک واحد روشن

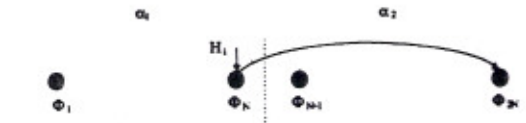
اگرواحد مخفی H_i در وضعیت Φ_1 قرار داشته باشد و پاداش بگیرد در همان وضعیت باقی میماند.

۲- واحد H_i روشن است ودر وضعیت Φ_j قرار دارد و بدلیل عملکرد نامناسبش جریمه میشود. در اینصورت از اهمیت این واحد کاسته شده و بسمت وضعیت های بیرونی تر حرکت میکند. نحوه حرکت چنین واحدی برای دو حالت مختلف در شکل زیر نشان داده شده است.

الف) وضعیت واحد H_i قبل ازجریمه شدن

ب) وضعیت واحد H_i بعد از جریمه شدن

شکل۳ :نحوه دادن جریمه به یک واحد روشن برای یک وضعیت غیر مرزی



الف) وضعیت واحد H_i قبل ازجریمه شدن

ب) وضعیت واحد H_i بعد از جریمه شدن

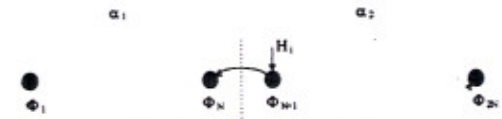
شکل ۴ : نحوه دادن جریمه به یک واحد روشن برای یک وضعیت مرزی

۳- واحد مخفی H_i خاموش است ودر وضعیت Φ_j قرار دارد و بدلیل عملکرد خوبش در گذشته پاداش میگیرد. در اینصورت به اهمیت این واحد افزوده میشود و به واحد های روشن نزدیکتر میشود یعنی به وضعیت های بیرونی تر منتقل میشود. نحوه حرکت چنیین واحدی برای دو حالت مختلف در شکل زیر نشان داده شده است.

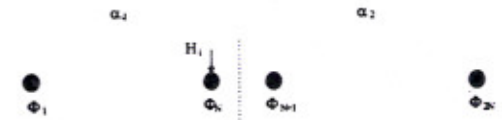
الف) وضعیت واحد H_i قبل از دادن پاداش

ب) وضعیت واحد H_i بعد از دادن پاداش

شکل۵ نحوه دادن پاداش به یک واحد خاموش برای یک وضعیت غیرمرزی



الف) وضعیت واحد H_i قبل ازدادن پاداش



ب) وضعیت واحد H_i بعد از دادن پاداش

شکل ۶نحوه دادن پاداش به یک واحد روشن برای یک وضعیت مرزی

۴- واحد مخفی H_i خاموش است ودر وضعیت Φ_j قرار دارد و بدلیل عملکرد نامناسبش در گذشته جریمه میشود. در اینصورت اهمیت این واحد کمتر میشود و از واحد های روشن دورتر میشود یعنی به وضعیت های داخلی تر اتوماتان منتقل میشود. نحوه حرکت چنین واحدی در شکل زیر نشان داده شده است.

الف) وضعیت واحد H_i قبل از جریمه شدن

ب) وضعیت واحد H_i بعد از جریمه شدن

شکل ۷ : نحوه دادن جریمه یک واحد خاموش

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

۴- الگوریتم تعیین تعداد واحد های لایه مخفی

در این قسمت الگوریتمی معرفی میشود که با استفاده از اتوماتان یا دگیر معرفی شده درقسمت قبل و الگوریتم انتشارخطا به عقب، ساختار شبکه را در حین آموزش مشخص میکند. طرز کار این الگوریتم بصورت زیر است.

در ابتدا همه واحد ها در مجموعه واحد های روشن و در وضعیت Φ_1 قرار میگیرند. به همه واحد های روشن مدتی اجازه داده میشود تا درآموزش شبکه شرکت نمایند. واحد هایی را که عملکرد آنها در این مدت مناسب نبوده جریمه میشوند، واحد هایی که عملکرد آنها خیلی خوب بوده پاداش پاداش میگیرند و واحدهایی که در مورد آنها نمیتوان تصمیم گیری قطعی

نمود بدون تغییروضعیت باقی میمانند. برای تشخیص خوب یا بد بودن عملکرد یک واحداز 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 را دارای عملکرد خوب میگوییم اگر به ازای همه الگوهای ورودی، مقدار فعالیت این واحد تغییرات زیادی داشته باشد. این بدان معنی است که اطلاعات مهمی در وزنهای این واحد ذخیره شده است.

عملکرد بد یک واحد: واحدی را دارای عملکرد بد میگوییم اگر به ازای همه الگوهای ورودی، مقدار فعالیت این واحد تغییرات کم داشته باشد. یعنی اطلاعات مهمی در وزنهای این واحد ذخیره نشده است.

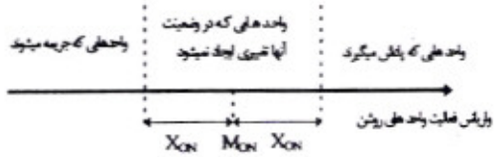
تشخیص نحوه عملکرد یک واحد روشن: واحدی را که مقدار فعالیت آن به ازای همه الگوهای آموزشی از یک مقدار آستانه ای کمتر باشد واحد بد و اگر مقدار فعالیت آن به ازای همه الگوهای آموزشی از یک مقدار آستانه ای دیگر بیشتر باشد واحد خوب مینامیم. برای تعیین مقادیر آستانه ای ابتدا واریانس فعالیت واحد به ازای همه الگوهای آموزشی بصورت زیر محاسبه میشود

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

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 Iام است که بصورت زیر تعریف میشود.

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

مطابق شکل ۸، 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 دیگر بیشتر باشد پاداش میگیرند و واحد هایی که واریانس فعالیت آنها بین این دو مقدار آستانه ای قرار میگیرند تغییری در وضعیت آنها ایجاد نمیشود.



شکل ۸: تشخیص مقدار آستانه واحد های روشن

مقدارM_{ON} که عبارتست از میانگین واریانس واحد های روشن بصورت زیر محاسبه میشود.

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

پهنایX_{ON} در شکل فوق بصورت زیر محاسبه میشود.

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

در رابطه بالا ثابت λ_{ON} ضریب پهنای روشنی نامیده میشود. $Max\,(\sigma_{ON})$ و $Min\,(\sigma_{ON})$ بترتیب بیشترین و کمترین مقدار واریانس فعالیت واحدهای روشن میباشند و بصورت زیر محاسبه میگردند.

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

واحد هایی که مقدار واریانس فعالیت آنها از $M_{ON} - X_{ON}$ کمتر باشدجریمه میشوند. واحدهایی را که مقدار واریانس فعالیت آنها از $M_{ON} + X_{ON}$ بیشتر باشند پاداش میگیرند و واحد هایی که مقدار واریانس آنها در بازه $[M_{ON} - X_{ON}, M_{ON} + X_{ON}]$ قراردارند تغییری در وضعیت آنها ایجاد نمیشود.

تشخیص نحوه عملکرد یک واحد خاموش: واحد هایی که خاموش هستند، در آموزش شبکه شرکت نمیکنند بنابراین فعالیت و واریانس فعالیت در مورد این واحد ها معنا ندارد. دراینصورت از گذشته این واحد ها برای روشن 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 نقطه ای که هم اکنون روی سطح خطا محاسبه میشود به احتمال زیاد به نقطه جواب بسیار نزدیکتر از نقطه ایست که در زمان روشن بودن آن واحد محاسبه شده است. لذا فعالیت یک واحد خاموش بصورت زیر محاسبه میشود.

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

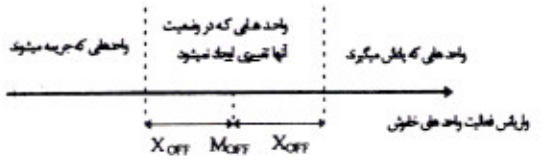
در رابطه فوق ثابت λ_d ضریب کاهش فعالیت نامیده میشود و n شاخص زمان را نشان میدهد. به این ترتیب مقدار فعالیت یک واحد خاموش بتدریج کم میشود. واریانس واحد های خاموش بصورت زیر محاسبه میشود

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

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

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

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



شکل ۹: تشخیص مقدار آستانه واحد های خاموش

و مقدار M_{OFF} که عبارتست از میانگین واریانس واحد های خاموش و بصورت زیر محاسبه میشود.

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

پهنای X_{OFF} در شکل فوق بصورت زیر محاسبه میشود.

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

در رابطه بالا ثابت λ_{OFF} ضریب پهنای خاموشی نامیده میشود. $Max\ (\sigma_{OFF})$ و $Min\ (\sigma_{OFF})$ بترتیب بیشترین و کمترین مقدار واریانس فعالیت واحدهای خاموش هستند و بصورت زیر محاسبه میشوند.

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

واحدهایی که مقدار واریانس فعالیت آنها از $M_{OFF} - X_{OFF}$ کمتر باشد جریمه میشوند. واحدهایی را که مقدار واریانس فعالیت آنها از $M_{OFF} + X_{OFF}$ بیشتر باشد پاداش میگیرند و واحد هایی که مقدار واریانس آنها دربازه $[M_{OFF} - X_{OFF}, M_{OFF} + X_{OFF}]$ قرار دارد تغییری در وضعیت آنها داده نمی شود.

برای استفاده از مزایای در حین آموزش شبکه های بزرگ، در ابتدای آموزش شبکه حساسیت واحدها کم در نظر گرفته میشود تا رقابت بین واحد ها کمتر باشد و پیچیدگی آموزش کاهش یابد و شبکه براحتی بتواند از کمینه های محلی عبور نماید. این عمل با کوچک انتخاب نمودن شیب تابع فعالیت انجام میشود. اما زمانیکه شبکه جواب تقریبی را تولید نمود حساسیت واحد ها را بتدریج افزایش داده و رقابت بین واحدها زیاد میشود. این عمل با بزرگ انتخاب نمودن شیب تابع فعالیت انجام میشود. برای اینکه الگوریتم بتواند تطبیق شیب تابع فعالیت را براحتی انجام دهد مقدار شیب تابع فعالیت بصورت زیر تنظیم میشود.

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

و γ شیب تابع سیگموئید $e^{-\gamma \times X}$ ، λ_s ضریب تغییر شیب تابع فعالیت و MSE میانگین مربع خطا می باشد.

زمانی که یک واحد از حالت خاموش به حالت روشن میرود برای تعیین مقادیر اولیه وزنهای این واحد دو روش وجود دارد. روش اول استفاده از مقادیر تصادفی برای وزنهای این واحد میباشد و روش دوم استفاده از وزنهایی است که در آخرین زمان روشن بودن محاسبه شده اند میباشد. البته این مقادیر نیز میتوانند بنوعی تصادفی تلقی شوند زیرا ابعاد فضای وزنها تغییر نموده است. در این مقاله از روش دوم برای تعیین مقادیر اولیه وزنها استفاده شده است. الگوریتم بقا برای تعیین ساختار مناسب شبکه در شکل ۱۰ 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 PenalizeOFFUnit (I)
        end if
        if σI > (MOFF + XOFF) then
          call RewardOFFUnit (I)
        end if
      end if
    end for
  until Termination Condition is Satisfied.
  return (W, ON)
end Algorithm

procedure PenalizeOnUnit (I)
  if State (I) < N then
    inc (State (I))
  else
    State (I) := 2 * N
  end if
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))
  end if
end procedure

procedure RewardOffUnit (I)
  dec (State (I))
end procedure
```

شکل ۱۰: الگوریتم بقا

مثالی از عملکرد اتوماتان یادگیر تعیین تعداد واحد هـای مخفـی:
برای روشن تر شدن نحوه کارکرد این اتوماتـان مثـالی در زیـر آورده شـده است. در این مثال ازاتوماتان HULA (2, 6, 6) بـا ضرائـب $\lambda_{ON} = 0.05$ و $\lambda_{OFF} = 0.01$ استفاده شده است. واریـانس فعـالیت واحـد هـای مخفـی و نحوه کارکرد اتوماتان در هشت مرحله درجدول زیر آمده شده است.

جدول ۱: واریا نس فعالیت واحد ها

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	۰/۰۳	۰/۱۸	۰/۴۳	۰	۰/۴	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
۰/۱۵	۰/۶	۰/۲	۰/۷	۰/۳۵	۰/۲۵	۰/۴۳	۰	۰/۱۷	۰
۰/۷	۰/۷	۰/۱۵	۰/۳	۰/۷	۰/۲	۰/۵۲	۰/۱۵	۰/۴۱	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 اول : در ابتدای آموزش، همه واحد های مخفی در وضعیت ۱ قرار میگیرند. در این مرحله وضعیت اتوماتان بصورت زیر است.

مرحلهٔ دوم : در ایـن مرحلـه، واحـد۳ جریمـه میشـود و واحـد ۶ پـاداش میگیرد و بقیه واحد ها تغییر وضعیت نمیدهند. بعـد از اتمـام ایـن مرحلـه وضعیت اتوماتان بصورت زیر است.

مرحله سوم : در این مرحله، واحد های۳ و ۴ جریمه میشوند و واحـد ۶ پاداش میگیرد و بقیه واحد ها تغییر وضعیت نمیدهنـد. بعـد از اتمـام ایـن مرحله وضعیت اتوماتان بصورت زیر است.

مرحله چهارم : در این مرحله اتوماتانْ تغییر وضعیت نمیدهد.

مرحله پنجم : در این مرحله، واحد های۳ و ۶ جریمه میشوند و واحـد ۴ پاداش میگیرد و بقیه واحد ها تغییر وضعیت نمیدهنـد. بعـد از اتمـام ایـن مرحله وضعیت اتوماتان بصورت زیر است.

مرحله ششم : در این مرحله، واحدهای۴ و ۶ جریمه و بقیـه واحـد هـا تغییر وضعیت نمیدهند. بعد از اتمام این مرحله وضعیت اتوماتـان بصـورت زیر است.

مرحله هفتم : در این مرحله، واحد های ۲و ۴ و ۶ جریمه و بقیـه واحـد ها تغییر وضعیت نمیدهنـد. بعـد از اتمـام ایـن مرحلـه وضعیـت اتوماتـان بصورت زیر است.

مرحله هشتم : در این مرحله، واحـد ۵ جریمـه و بقیـه واحـد هـا تغییـر وضعیت نمیدهند. بعد از اتمام این مرحله وضعیـت اتوماتـان بصـورت زیـر است.

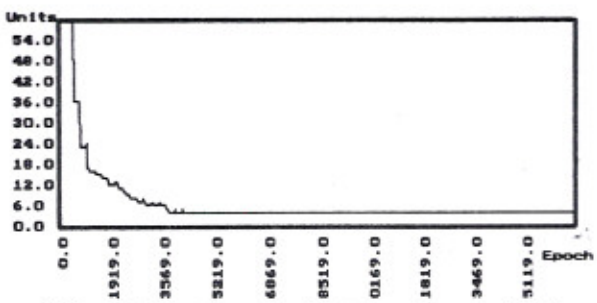
۵- نتایج شبیه سازی

برای نشان دادن کارایی های الگوریتم پیشنهاد شـده، ایـن الگوریتـم روی پنج مسئله Parity سه بیتی ، تشخیص اعداد لاتین ، Encoding، تقـارن و XOR که تعداد تقریبی واحد های مخفی برای آنها مشخص است آزمـایش شده است. تعدادی از این آزمایشات در این بخش آورده شده است. بـرای تمامی این مسائل از اتوماتان HULA (2, 7, 60) استفاده شـده اسـت. لازم بذکر است که برای آموزش شبکه هـا از الگوریتـم انتشـار خطـا بـه عقـب استاندارد استفاده شده است.

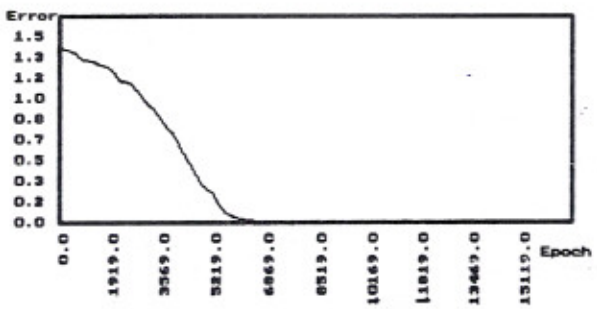
۱- مسئله Parity سه بیتی: نشان داده شده است که برای تولید Parity الگوهای n بیتی نیاز به n واحد مخفی میباشد [۲۹]. البته شبکه هـای بـا سـاختار هـای دیگـر میتواننـد ایـن مسـئله را بـا دو واحـد مخفـی حـل نمایند[۲۳]. الگوریتم بقا روی ۲۰ شبکه با وزنهای اولیه تصادفی برای این مسئله آزمایش شده است و نتایج آن در جدول زیر آمده است. لازم بذکـر است که در تمامی موارد آزمایش شده، الگوریتـم قـادر بـه آمـوزش شـبکه بوده است. شکل های ۱۱ و ۱۲ منحنی تغییرات تعداد واحدهای مخفـی و میانگین مربع خطا براساس تکرار ارائه الگو(Epoch) بـه شـبکه بـرای یـک 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
۱۳	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 ۱۱: منحنی تغییرات تعداد واحد های مخفی بر اساس تکرار ارئه الگو(Parity)

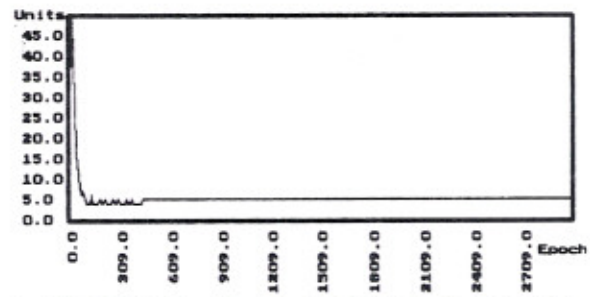


شکل ۱۲ : منحنی میانگین مربع خطا بر اساس تکرار ارئه الگو (Parity)

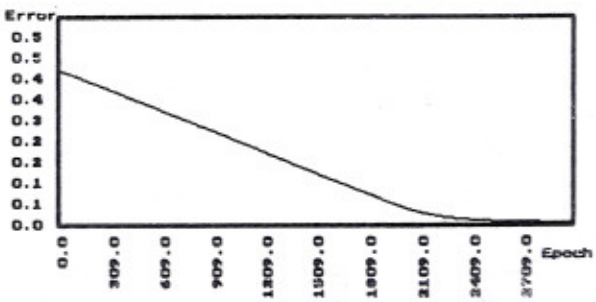
۲- مسئله تشخیص اعداد لاتین: در این مسئله ده عدد وجود دارد که هر کدام از آنها توسط یک ماتریس ۸×۸ از نقطه های سیاه و سفید نشان داده میشود[۳۴]. برای حل این مسئله حداقل چهار واحد مخفی در لایه میانی مورد نیاز میباشد. این الگوریتم روی ۲۰ شبکه با وزنهای اولیه تصادفی آزمایش شده اند و نتایج آن در جدول زیر نشان داده شده است. لازم بذکر است که در تمامی موارد آزمایش شده، الگوریتم قادر به آموزش شبکه بوده است. شکل های ۱۳ و ۱۴ 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 اعدادلاتین

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	۰/۰۴۹۲۸	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



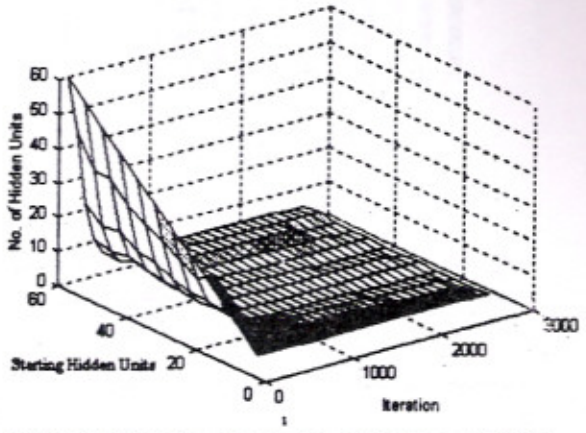
شکل ۱۳: منحنی تغییرات تعداد واحد های مخفی بر اساس تکرار ارئه الگو (اعدادلاتین)



شکل۱۴ : منحنی میانگین مربع خطا بر اساس تکرار ارئه الگو (اعدادلاتین)

شکل ۱۵ تعداد واحد های مخفی تعیین شده بر اساس 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 واحد هایی است که در شروع آموزش شبکه در نظر گرفته میشود. هر چند ساختار های تولید شده اختلاف جزئی با هم دارند اما همه بسیار نزدیک به مقدار بهینه هستند. هر نقطه از این شکل متوسط ۱۰ اجرای مختلف میباشد.

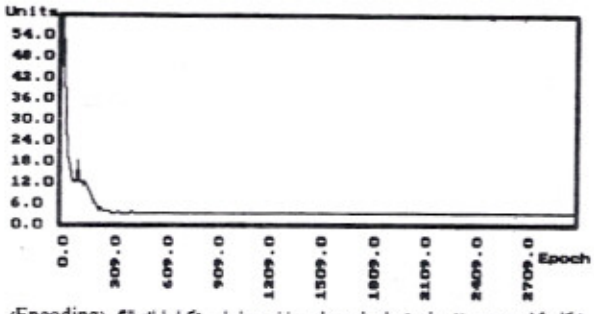


شکل ۱۵: منحنی تعداد واحد های مخفی تعیین شده بر اساس تعداد واحد های اولیه

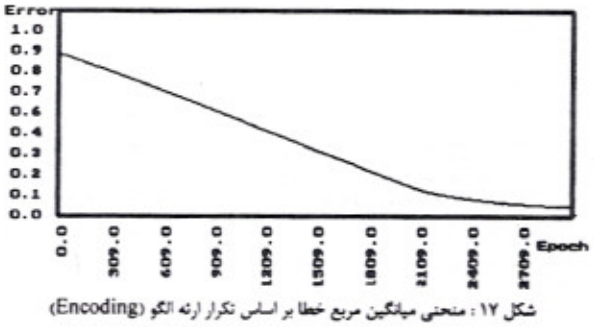
۳- مسئله Encoding : مسئله Encoding برای بردارهای n بیتی ورودی ، نیاز به $\log_2^n$ واحد مخفی دارد[۲۹]. این الگوریتم روی ۲۰ شبکه با وزنهای اولیه تصادفی برای بردارهای هشت بیتی ورودی آزمایش شده اند و نتایج آن در جدول زیر نشان داده شده است . در تمامی موارد آزمایش شده، الگوریتم قادر به آموزش شبکه بوده است. 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 ۳: 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
۱۳	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	۰/۱۲۲۶۶	۱۰۰
۲۰	۳	۰/۰۰۹۹۹	۱۰۰
متوسط	۳/۰۵	۰/۰۱۸۵۶	۱۰۰



شکل ۱۶ : منحنی تغییرات تعداد واحد های مخفی بر اساس تکرار ارائه الگو (Encoding)



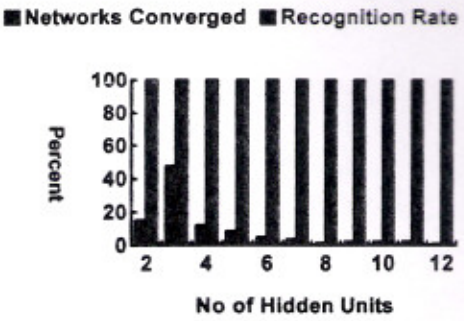
شکل ۱۷ : منحنی میانگین مربع خطا بر اساس تکرار ارئه الگو (Encoding)

۴- مسئله XOR : برای پیاده سازی 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 زیر آورده شده است.

جدول۵ : نتایج شبیه سازی برای مسئله XOR

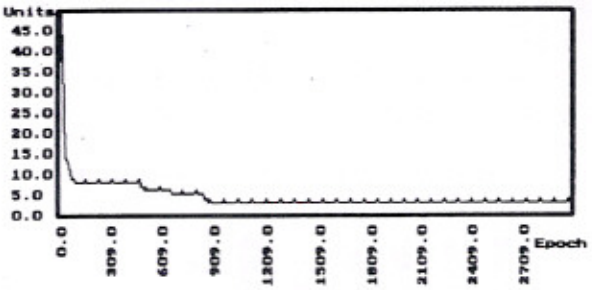
شبکه	تعداد واحدهای مخفی	متوسط مربع خطا	درصد تشخیص
۲۲-۱	۲	۰/۰۲۲۵۸۲	۱۰۰
۹۱-۲۳	۳	۰/۰۰۴۱۳۲	۱۰۰
۱۰۹-۹۲	۴	۰/۰۰۰۴۹۰	۱۰۰
۱۲۱-۱۱۰	۵	۰/۰۰۰۴۸۷	۱۰۰
۱۲۸-۱۲۲	۶	۰/۰۰۰۴۸۰	۱۰۰
۱۳۳-۱۲۹	۷	۰/۰۰۰۴۶۰	۱۰۰
۱۳۵-۱۳۴	۸	۰/۰۰۰۴۲۵	۱۰۰
۱۳۸-۱۳۶	۹	۰/۰۰۰۴۲۷	۱۰۰
۱۴۱-۱۳۹	۱۰	۰/۰۰۰۴۸۵	۱۰۰
۱۴۴-۱۴۲	۱۱	۰/۰۰۰۴۸۲	۱۰۰
۱۴۵	۱۲	۰/۰۰۰۴۸۲	۱۰۰
متوسط	۳/۹۸	۰/۰۰۷۲۶	۱۰۰

از بین ۱۷۰ شبکه آزمایش شده، این الگوریتم قادر به آموزش ۲۵ شبکه نبوده است. تعداد واحدهای مخفی درصد تشخیص آنها برای مواردی که الگوریتم همگرا شده است توسط هیستوگرام شکل ۱۸ داده شده است. برای این مسئله ۱۵/۲ درصد از شبکه ها 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.

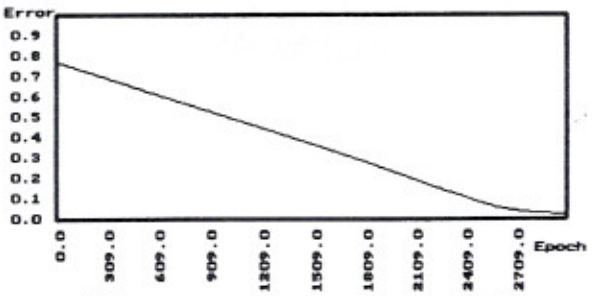


شکل۱۸ : هیستوگرام تعداد واحد های مخفی تعیین شده توسط الگوریتم بقا

شکل های 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.



شکل ۱۹: منحنی تغییرات تعداد واحد های مخفی بر اساس تکرار ارائه الگو(XOR)



شکل ۲۰: منحنی میانگین مربع خطا بر اساس تکرار ارائه الگو (XOR)

۵- مسئله تقارن: برای حل این مسئله حداقل دو واحد مخفی مورد نیاز میباشد[۲۶]. دو گروه آزمایش برای مسئله تقارن انجام گرفته است: تقارن چهار بیتی و تقارن شش بیتی. برای بردارهای چهار بیتی ۶۵ شبکه و برای بردار های شش بیتی ۱۲ شبکه توسط الگوریتم پیشنهادی آموزش داده شده اندکه نتایج آن در ذیل آمده است.

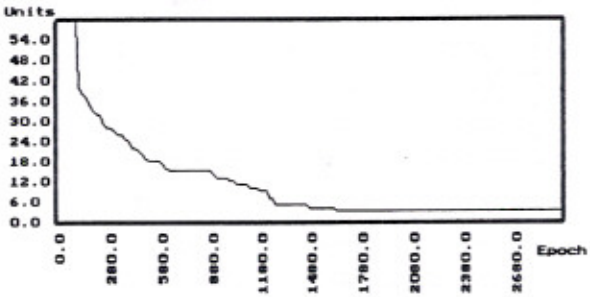
جدول ۶ : نتایج شبیه سازی برای مسئله تقارن چهار بیتی

شبکه	تعداد واحدهای مخفی	متوسط مربع خطا	درصد تشخیص
۹-۱	۲	۰/۰۱۳۱۷۶	۱۰۰
۱۸-۱۰	۳	۰/۰۵۶۲۱۸	۱۰۰
۲۷-۱۹	۴	۰/۰۸۰۳۹۲	۱۰۰
۴۱-۲۸	۵	۰/۰۰۰۸۲۲۵	۱۰۰
۴۶-۴۲	۶	۰/۰۰۰۲۹۱۸	۱۰۰
۵۰-۴۷	۷	۰/۰۰۰۹۴۲۵	۱۰۰
متوسط	۴/۱۸	۰/۰۲۷۴۳	۱۰۰

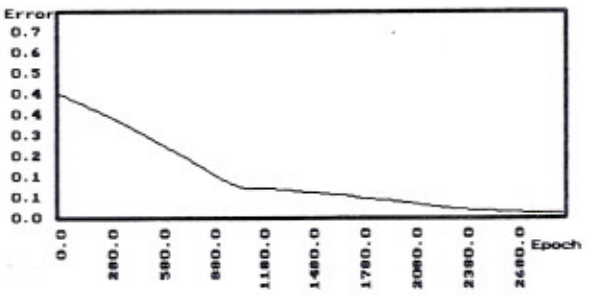
جدول ۷ : نتایج شبیه سازی برای مسئله تقارن شش بیتی

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

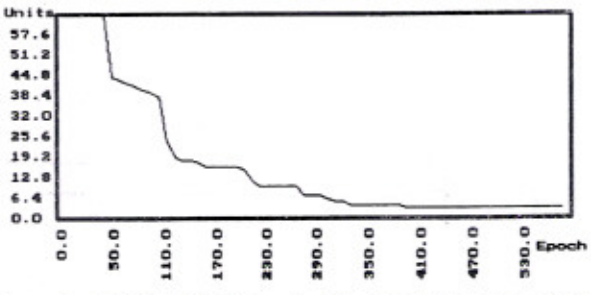
شکل های ۲۱ الی ۲۴ منحنی تغییرات تعداد واحد های مخفی و میانگین مربع خطا براساس تکرار ارائه الگو به شبکه را برای یک شبیه سازی نمونه نشان میدهند.



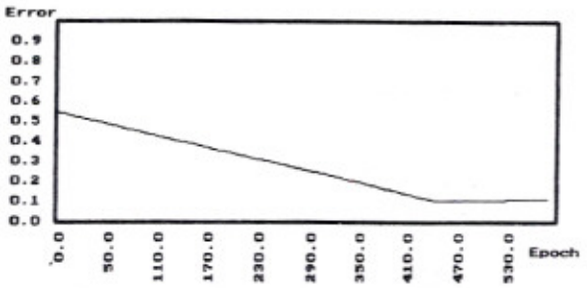
شکل ۲۱: منحنی تغییرات تعداد واحد های مخفی بر اساس تکرار ارائه الگو (تقارن چهاربیتی)



شکل ۲۲: منحنی میانگین مربع خطا بر اساس تکرار ارائه الگو (تقارن چهاربیتی)



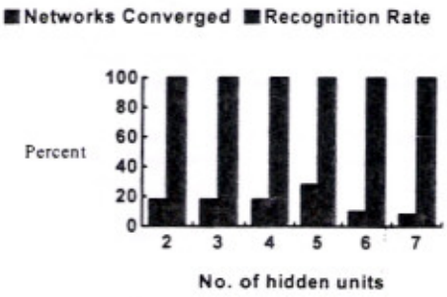
شکل۲۳: منحنی تغییرات تعداد واحد های مخفی بر اساس تکرار ارائه الگو (تقارن شش بیتی)



شکل ۲۴: منحنی میانگین مربع خطا بر اساس 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 ۲۵ 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 چها واحد، ۲۸ درصد به پنج واحد و کمتر از ۱۰ درصد به شش واحد و ۸ درصد به هفت واحد همگرا میشوند.

برای مسائل XOR و تقارن زمانی، که تعداد واحدهای مخفی به مقدار بهینه نزدیک میشود سرعت آموزش شبکه بسیار کند میگردد.



شکل ۲۵: هیستوگرام تعداد واحد های مخفی تعیین شده توسط الگوریتم بقا

۶- نتیجه گیری

در این مقاله الگوریتمی برای تعیین تعداد واحد های مخفی شبکه عصبی سه لایه بنام الگوریتم بقا که از اتوماتان های مهاجرت اشیاء استفاده میکند 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 بصورت مسئله افراز مجموعه ها تعریف شده است. نتایج آزمایشات نشان میدهند که تعداد واحد های تعیین شده توسط این الگوریتم خیلی نزدیک به مقدار بهینه میباشد وهمچنین مستقل از تعداد واحد هایی است که در شروع آموزش شبکه در نظر گرفته میشود. بعنوان 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 (جداول ۸ الی ۱۱) مقایسه گردیده است. مقایسه این جداول با جداول داده شده در این مقاله کارایی الگوریتم بقا را نشان میدهد.

جدول ۸ : نتایج هرس بوسیله روش محاوره ای 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 ۹: نتایج هرس بوسیله 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 ۱۰: نتایج هرس بوسیله 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 ۱۱ : نتایج هرس بوسیله 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	۳	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

اغلب روشهای گزارش شده برای تعیین ساختار شبکه از الگوریتم های کوهنوردی استفاده میکنند و مشکل گرفتاری در حداقل محلی را دارند. در این روش بدلیل استفاده از روشهای جستجوی عمومی، امکان گرفتاری در حداقل های محلی فضای ساختارها کم میشود.

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. 1998 (In Persian).

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

[5] Castellano, G., Fanelli, A. M., and Pelillo, M. (1997). A 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 Cascade - 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. 198-209.

[8] Hashim, A. A., Amir, S., and Mars, p. (1986). Application of Learning Automata 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 Bottelnecks 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. 8, No. 3 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 Systems 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 (To appear).

[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] Nabhan, T. M. and Zomaya, A. Y. (1994). Toward Neural Networks Structures for Function Appriximation, 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, No. 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. V. de St. (1996). Graph Partioning Using Learning Automata, IEEE Trans. on Computers, No. 45, No. 2, pp. 195-208.

[29] Rumelhart, D. E., Hinton, G. E., and Williams, R. J.(1986). Learning Internal Representations by Error Backropagation, In Parallel distributed processing, Cambridge, MA: MIT Press.

[30] Reed, R. (1993). Prunning 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 Artifitial 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 Starita, 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 Layers 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 Throught a Cooperative Game of Automata, IEEE Trans. Syst., Man and Cybern., Vol. SMC-27, pp.73-85.

[37] Whitley, D. amd 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 Artifitial 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.