مهندسی شبکه های عصبی توسط اتوماتانهای یادگیر:
ارائهٔ یک الگوریتم تطبیقی برای تعیین تعدا د وزنهای ورودی نرونهای لایهٔ مخفی برای شبکه های عصبی سه لایه

بهبود مشعوفی
دانشجوی دکتری دانشکدهٔ مهندسی برق
سید احمد معتمدی
دانشیار دانشکدهٔ مهندسی برق

محمد رضا میبدی
دانشیار دانشکدهٔ مهندسی کامپیوتر
محمد باقر منهاج
دانشیار دانشکدهٔ مهندسی برق

دانشگاه صنعتی امیر کبیر
تهران، ایران

چکیده

الگوریتم پس انتشار خطای استاندارد، فقط در فضای وزنهای شبکه، با توپولوژی ثابت، عمل جستجو را انجام میدهد. تعداد لایه ها، نرونها و وزنهای شبکه، تأ ثیر بسزائی بر روی کارائی شبکه دارد. بنابر این نیاز به الگوریتمهایی داریم که بتوانند بطور اتوماتیک ساختار مناسب شبکه را تعیین کنند. برای تعیین اندازهٔ مطلوب برای شبکه های عصبی الگوریتمهای گوناگونی توسط افراد مختلف ارائه شده است. توسط آقایان میبدی و بیگی الگوریتم جدیدی با استفاده از اتوماتان یادگیری مهاجرت اشیا برای تعیین تعداد وزنها طراحی گردیده است. این الگوریتم با حذف وزنهایی که دارای تأثیر کمتری هستند نه تنها باعث کاهش پیچیدگی شبکه میشود. بلکه قدرت تعمیم شبکه را نیز افزایش میدهد. در الگوریتم تعیین تعداد وزن، در ابتدا همهٔ وزنهای موجود در شبکه روشن بوده و به همهٔ وزنها مدتی اجازه داده میشود تا در آموزش شبکه شرکت نمایند. وزنهای روشنی که قدر مطلق مقدار آنها از یک مقدار آستانه ای کمتر باشد جریمه شده و وزنهای روشنی که قدر مطلق آنها از یک مقدار آستانه ای دیگر بیشتر باشد پاداش میگیرند. وزنهای روشنی که قدر مطلق مقدار آنها بین دو مقدار آستانه ای قرار گیرد تغییری در وضعیت آنها ایجاد نمیشود. با انتخاب مقادیر بهینه برای این بازه ها میتوانیم در کمترین زمان به شبکه هایی با حداقل تعداد وزن که بتواند با خطای قابل قبولی الگوهای آموزش را یاد گرفته و همچنین از قدرت تعمیم قابل قبولی برخوردار باشد برسیم. در این مقاله با استفاده از اتوماتان یادگیر، بازه های تصمیم گیری را تطبیق داده ایم. درالگوریتم پیشنهادی برای پاسخ دادن به اتوماتان مربوط به تنظیم بازه های تصمیم گیری، از معیار خطا کمک گرفته ایم. مالگوریتم پیشنهادی بر روی مسائل مختلفی پیاده سازی شده است. نتایج مشابه سازیها نشان میدهد شبکه های تولید شده توسط الگوریتم جدید از پیچیدگی پائین تری نسبت به الگوریتم تعیین وزن غیر تطبیقی برخوردار میباشند.

کلمات کلیدی: **مهندسی شبکه های عصبی، شبکه های عصبی چندلایه، ساختار شبکه های عصبی، الگوریتم پس انتشار خطا، اتوماتانهای یادگیر**

۱- مقدمه

در سالهای اخیر مدلهای شبکهٔ عصبی زیادی، برای مسائل طبقه بندی الگو، تقریب تابع ، بازشناسی گفتار و ... ارائه گردیده است. در بین اینها، شبکه های عصبی جلو روندهٔ چند لایه، مهمترین آنها میباشد. روشهایی که از الگوریتم پس انتشار خطای استاندارد استفاده میکنند فقط در فضای وزنهای شبکه با توپولوژی ثابت، عمل جستجو را انجام میدهند[۱]. این روشها در حالت کلّی فقط موقعی مناسب میباشد که ساختار شبکه درست انتخاب شده باشد. تعداد لایه ها، نرونها و وزنهای شبکه، تأ ثیر بسزائی بر روی کارائی شبکه دارد. شبکه های خیلی کوچک، قادر به یادگیری مسئله نبوده و شبکه های با ابعاد بزرگ دچار Overfitting شده و قدرت تعمیم[1] پائینی خواهد داشت. علاوه براین بسیارکند بوده و هزینهٔ بالائی خواهد داشت. حالت مشابه را در مسئلهٔ برازش منحنی[2] با استفاده از چند جمله ایها شاهد هستیم. داده هایی را در نظر میگیریم که توسط یک تابع که نویز جمع شونده نیز در خروجی آن قرار دارد ایجاد شده است. یک چند جمله ای با تعداد پائین ضرائب قادر نخواهد بود از روی داده ها تابع را مدل سازی کند از طرف دیگر یک چند جمله ای با تعداد بالای ضرائب، نویز موجود در داده ها را نیز مدلسازی کرده در نتیجه مدل خوبی را برا ی تابع ارائه نخواهد کرد. ولی

1) Generalization
2) Curvefitting

اگر تعداد ضرائب بطور مناسب انتخاب شود، چند جمله ای، نمایش خوبی را برای تابع، همچنین پیشگویی دقیقی را برا ی داده های جدید ارائه خواهد کرد. در شبکه های عصبی نیز همین مسئله اتفاق می افتد. در این حالت نیز پیچیدگی مدل بایستی با مسئله تطبیق داده شود. بنابر این نیاز به الگوریتمهایی داریم که بتوانند بطور اتوماتیک ساختار مناسب شبکه را تعیین کنند. الگوریتمهائی را که تاکنون توسط افراد مختلف بمنظور تعیین اندازهٔ مطلوب برای شبکه های عصبی ارائه شده است میتوان به پنج گروه عمدهٔ زیرتقسیم کرد[۴۲].

الف)الگوریتمهای هرس[1]: در این دسته از الگوریتمها، از یک شبکهٔ بزرگ شروع کرده و آنرا آموزش میدهیم سپس وزنها و نرونهای زائد را حذف میکنیم. اندازهٔ اولیهٔ بزرگ، این اجازه را میدهد که شبکه سریعاً، با حداقل حساسیت به شرایط اولیهٔ، آموزش ببینند. و پیچیدگی پائین شبکهٔ هرس شده نیز قدرت تعمیم آنرا افزایش میدهد. این الگوریتمها بد و گروه عمده تقسیم میشوند. گروه اول، حساسیت تابع خطا نسبت به حذف یک عنصر را تخمین زده، عناصر با حداقل تأثیر را حذف میکند[۶]-[۳] . گروه دوم، عباراتی را به تابع هدف اضافه میکند که به شبکه برای انتخاب جوا بهای کارآمد پاداش میدهد[۱۴]-[۷] .

ب)الگوریتمهای سازنده[2]: این الگوریتمها از یک شبکهٔ کوچک شروع کرده سپس تا حصول یک جواب رضایت بخش، واحدهای مخفی و وزنهایی را به شبکه اضافه میکنند[۱۹]-[۱۵],[۱] .

ج)الگوریتمهای ترکیبی: این دسته از الگوریتمها در حقیقت ترکیبی از الگوریتمهای هرس و الگوریتمهای سازنده میباشد. این الگوریتمها در طول آموزش با اضافه و حذف کردن وزنها و نرونها سعی در رسیدن به یک شبکه با ساختار اپتیمال را دارند[۲۱],[۲۰] .

د)الگوریتمهای تکاملی[3]: این الگوریتمها با استفاده از یک معیار کارآیی مثل حداقل خطا و یا پیچیدگی آموزش، در فضای ساختارها، بدنبال مناسبترین ساختار میباشند. در این روش هر نقطه از فضای جستجو، متناظر با یک ساختار شبکه میباشد[۲۶] ,[۲۲] .

هـ)الگوریتمهای بر اساس اتوماتانهای یادگیر[4]: برای اولین بار توسط آقایان میبدی و بیگی الگوریتمهایی بر اساس اتوماتانهای یادگیر برای دستیابی به ساختارهای بهینه با پیچیدگی آموزش کم و قدرت تعمیم بالا ارائه گردیده است [۲۷],[۲] . در اولین الگوریتم ارائه شده که تحت عنوان الگوریتم بقا می باشد از یک اتوماتان یادگیر مهاجرت اشیا[5]بعنوان یک ابزار جستجوی عمومی استفاده شده است. این الگوریتم حین آموزش، ساختار مناسبی برای شبکهٔ عصبی سه لایه از حیث پائین بودن پیچیدگی آموزش و قدرت تعمیم بالا تعیین میکند. الگوریتم بقا آموزش را با یک شبکهٔ عصبی سه لایهٔ بزرگ شروع کرده و با افزودن و کاستن نرونهای مخفی، تعداد نرونهای لایهٔ مخفی شبکه را تعیین میکند. در ادامهٔ کار در مقالهٔ [۴۲] سه الگوریتم دیگر توسط آقایان میبدی و بیگی بر اساس اتوماتانهای یادگیر مهاجرت اشیا و الگوریتم یادگیری پس انتشار خطا ارائه گردیده است. این الگوریتمها شامل دو بخش میباشند. در بخش اول تعداد نرونهای مخفی مورد نیاز شبکه، و در بخش دوم تعداد وزنهای هر نرون مشخص میگردد. در الگوریتم ۱ برای تعیین تعداد واحدهای مخفی از الگوریتم بقا استفاده شده است. برای تعیین تعداد وزنها الگوریتم جدیدی با استفاده از اتوماتان یادگیری مهاجرت اشیا طراحی گردیده است. این الگوریتم با حذف وزنهایی که دارای تأثیر کمتری هستند نه تنها باعث کاهش پیچیدگی شبکه می شود. بلکه افزایش تعمیم شبکه را هم بدنبال دارد. الگوریتم تعیین وزنها با خاموش و روشن کردن وزنها سعی در پیدا کردن وزنهای مناسب دارد. در دو الگوریتم دیگر هدف فقط تعیین تعداد وزنهای لایهٔ مخفی شبکه است و تعداد واحدهای لایهٔ مخفی از طریق تعیین تعداد وزنهای شبکه مشخص میگردد. در الگوریتم کاهش تعداد وزن، [۴۲] از یک اتوماتان برای تعیین تعداد وزنهای لایهٔ مخفی استفاده شده است. در ابتدا همهٔ وزنهای موجود در شبکه روشن بوده و به همهٔ وزنها مدتی اجازه داده میشود تا در آموزش شبکه شرکت نمایند. وزنهای روشنی که قدر مطلق مقدار آنها از یک مقدار آستانه ای کمتر باشد جریمه شده و وزنهای روشنی که قدر مطلق آنها از یک مقدار آستانه ای دیگر بیشتر باشد پاداش میگیرند. وزنهای روشنی که قدر مطلق مقدار آنها بین دو مقدار آستانه ای قرار بگیرد تغییری در وضعیت آنها ایجاد نمیشود. در مورد وزنهای خاموش نیز وضع بهمین منوال است وزنهای خاموشی که قدر مطلق آنها کمتر از یک مقدار آستانه ای باشد پاداش میگیرند و وزنهای خاموشی که قدر مطلق آنها بیشتر از یک مقدار آستانه ای باشد جریمه میشو ند. وزنهایی که بین دو مقدار آستانه ای قرار میگیرند تغییری در وضعیت آنها صورت نمیگیرد. مقادیرآستانه ای که در این تصمیم گیریها مورد استفاده قرار میگیرد بسیار تعیین کننده بوده و تأثیر چشمگیری در عملکرد شبکهٔ عصبی دارد این مطلب در بخشهای بعدی مقاله بطور مبسوط توضیح داده خواهد شد. با انتخاب مقادیر بهینه برای این بازه هامیتوانیم در کمترین زمان به شبکه ه هایی با حداقل تعداد وزن که بتواند با خطای قابل قبولی الگوهای آموزش را یاد گرفته و از قدرت تعمیم قابل قبولی برخوردار باشد برسیم. در این مقاله با استفاده از اتوماتان یادگیر، بازه های تصمیم گیری را تطبیق داده ایم. در الگوریتم پیشنهادی برای پاسخ دادن به اتوماتان مربوط به تنظیم بازه های تصمیم گیری، از معیار خطا کمک گرفته ایم. مینیمم مقدار خطا در تکرار فعلی را با مینیمم مقدار خطا در تکرار قبلی مقایسه میکنیم اگر با یک ضریبی خطا افزایش پیدا کرده بود اتوماتان را جریمه میکنیم در غیر اینصورت به آن پاداش میدهیم.

1) Pruning Algorithms
2) Constructive Algorithms
3) Evolutionary Algorithms
4) Learning Automata
5) Object Migrating Learning Automata

بخش های بعدی مقاله بصورت زیر سازماندهی شده است. در بخش ۲ الگوریتم یادگیری پس انتشار خطا که برای آموزش شبکه های عصبی چند لایه مورد استفاده قرار می گیرد توضیح داده شده است. موضوع Overtraining و قدرت تعمیم شبکه در بخش ۳ مورد بررسی قرار گرفته است. مقدمه ای بر اتوماتانهای یادگیر و انواع مهم آن در بخش ۴ آورده شده است. در بخش ۵ الگوریتم پیشنهادی را توضیح داده ایم. نتایج آزمایشها برای مسا ئل مختلف در بخش ۶ ، قدر دانی و تشکر در بخش ۷ و در پایان نتیجه گیری آورده شده است.

۲ - الگوریتم پس انتشار خطا

الگوریتم BP یک روش سیستماتیک برای آموزش شبکه های عصبی چند لایه می باشد. الگوریتم BP بطور مبسوط در مرجع [۲۸] توضیح داده شده است. الگوریتم BP ، دو مسیر محاسباتی دارد. مسیر اول مسیر پیشخور یا رفت و مسیر دوم مسیر پس خور یا مسیر برگشت نامیده می شود.

مسیر رفت: این مسیر با معادلات زیر توصیف می شود:

$$\begin{aligned} &\underline{a}^0 = \underline{p}(k) \\ &\underline{a}^{l+1}(k) = \underline{F}^{l+1}\left(W^{l+1}(k)\underline{a}^l + \underline{b}^{l+1}(k)\right), \quad l = 0,1,\ldots,L-1 \\ &\underline{a} = \underline{a}^L(k) \end{aligned} \tag{۱}$$

در این مسیر وزنها و بایاسهای شبکه تغییر داده نمی شوند. توابع محرک، روی تمامی نرونها عمل می کند، یعنی:

$$\underline{F}^{l+1}(\underline{n}(k)) = \left[f^{l+1}\left(n_1(k)\ldots f^{l+1}\left(n_{s_{l+1}}(k)\right)\right)\right]^T \tag{۲}$$

مسیر برگشت: در این مسیر بردارهای حساسیت از لایه آخر به لایه اول برگشت داده می شوند. معادلات زیر، دینامیک مسیر برگشت را بیان می کند:

$$\begin{aligned} &\underline{\delta}^L(k) = -2F^L(\underline{n})\underline{e}(k) \\ &\underline{\delta}^l(k) = F^l(\underline{n}^l)(W^{l+1})^T\underline{\delta}^{l+1}, \quad l = L-1,\ldots,1 \\ &\underline{e}(k) = \underline{t}(k) - \underline{a}(k) \end{aligned} \tag{۳}$$

در مسیر برگشت ابتدا با در دست بودن بردار هدف، بردار خطا محاسبه می شود. سپس بردار خطا از سمت راست به چپ و از لایه آخر به لایهٔ ا ول توزیع شده، گرادیان محلی نرون به نرون با الگوریتم بازگشتی محاسبه میشود.

تنظیم پارامترها: در این مرحله ماتریسهای وزن و بردارهای بایاس شبکه بصورت زیر تنظیم می شوند.

$$\begin{aligned} &W^l(k+1) = W^l(k) - \alpha\underline{\delta}^l(k)\left(\underline{a}^{l-1}(k)\right)^T \\ &b^l(k+1) = b^l(k) - \alpha\underline{\delta}^l(k), \qquad l = 1,2,\ldots,L \end{aligned} \tag{۴}$$

توقف: اگر میانگین مربعات خطا در هر e poch (جمع مربعات خطا برای تمامی الگوهای یادگیری) کمتر از مقدار از پیش تعیین شده ای بوده و یا اینکه فرم تغییرات در پارامترهای شبکه پس از هر سیکل خیلی کوچک باشد الگوریتم BP متوقف می شود.

۳- Overtraining و قدرت تعمیم

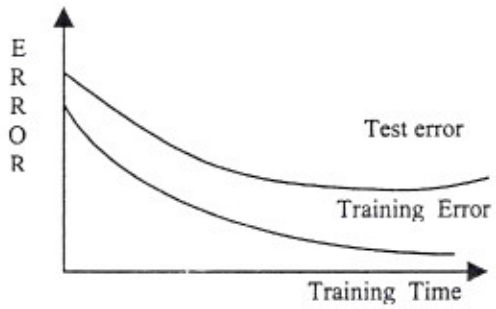
موقعیکه یک شبکهٔ عصبی آموزش می بیند، وزنها طوری تغییر داده میشوند که خطا کاهش یابد. اگر شبکه توسط الگوهایی که فقط اختلاف جزئی با الگوهای آموزش دارند تست شود خطا در الگوهای تست همچنانکه شبکه آموزش میبیند و قدرت تعمیم آن بیشتر میشود کاهش پیدا میکند. این مسئله در شکل (۱) نشان داده شده است. اگر الگوهای آموزش ناکافی باشد در اینصورت محتوی داده های نادرست و گمراه کننده خواهد بود[۴۱].

در بعضی نقاط، بویژه در مراحل بعدی یادگیری، داده های ناکافی تأثیر خود را نشان داده و با اینکه خطای آموزش کاهش می یابد خطای آزمایش شروع به افزایش میکند [۲۹] . یک روش برای پرهیز از Overfitting این است که توانائی تعمیم را بهنگام آموزش تخمین زده و زمانیکه شروع به کاهش میکند آموزش را قطع نمائیم. ساده ترین روش این است که الگوها را به دو گروه الگوهای آموزش و الگوها تصدیق[1] تقسیم کنیم. الگوهای آموزش برای تغییر دادن وزنها استفاده میشود و الگوهای تصدیق برای تخمین توانائی تعمیم استفاده شده و آموزش زمانی متوقف می شود که خطا روی الگوهای تصدیق شروع به افزایش کند. این روش موقعیکه الگوهای کمی برای آموزش در اختیارمیباشد بدلیل اینکه الگوهای تصدیق نمی تواند برای آموزش مورد استفاده قرار گیرد عملی نمی باشد[۳۲] -[۳۰] . یک راه دیگر برای دوری جستن از Overtraining این است که توانایی شبکه در بهره گیری از همبستگیهای نادرست در الگوها را محدود کنیم. Overtraining موقعی اتفاق می افتد که درجهٔ آزادی شبکه (تعداد وزنها) بیشتر از الگوهای آموزش باشد. با اینکه شبکه با اعمال الگوهای آموزش جواب دقیقی رابدست میدهد

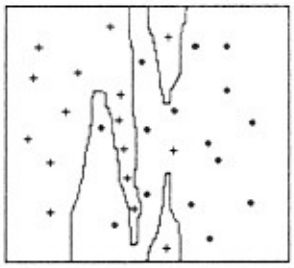
1)Validation

ولی ممکن است در نقاط دیگر خیلی بد عمل کند. شبکه های کوچک در کنار قدرت تعمیم بالا مزیتهای دیگری را نیز بهمراه دارند. معمولاً اینها را میتوان خیلی سریع و با هزینهٔ کمتری پیاده سازی کرد. تئوریها و فرمولهایی برای تخمین اندازهٔ مورد نیاز سیستم، مورد استفاده قرار گرفته است[۳۵]-[۳۳].

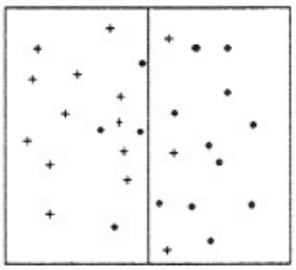
این روشها پیچیدگی سیستم یادگیری و تعداد الگوهای مورد نیاز برای یادگیری یک تابع را بهم ربط میدهند. اگر تعداد الگوها نسبت به پیچیدگی سیستم کوچک باشد. در اینصورت انتظار داریم خطای تعمیم زیاد باشد. شکلهای ۲و ۳ تأثیر شاخه زنی را نشان میدهد. شکل ۲ مرزهای تشکیل شده توسط یک شبکهٔ ۴ لایه با دو ورودی، ۲ لایهٔ مخفی با ۵۰ نرون در لایهٔ مخفی اول و ۱۰ نرون در لایهٔ مخفی دوم و یک خروجی را نشان میدهد. شبکه دارای ۶۷۱ وزن بوده اما فقط ۳۱ الگو برای آموزش وجود دارد. بنابر این تعداد پارامترهای شبکه در مقایسه با تعداد الگوها خیلی زیاد میباشد. گرچه داده ها تقریباً بطور خطی جدا پذیرند (با کمی همپوشانی در نزدیکی مرز) با اینحال مرز طبقه بندی شبکه خیلی غیر خطی بوده و احتمالاً قدرت تعمیم خوبی روی داده های اضافه دیگر از همان تابع نخواهد داشت. شکل ۳ همان شبکه را پس از شاخه زنی نشان میدهد. اندازهٔ شبکه به ۲/۲/۲/۱ با ۱۵ وزن تقلیل پیدا کرده و مرز تصمیم گیری [illegible] یکنواخت میباشد. روشهای پیچیده تر می تواند شبکه را به یک شبکهٔ یک بعدی فقط با ۲ وزن تقلیل دهد. مثال همچنان نشان میدهد که [illegible] شاخه زنی میتواند بعنوان ابزاری برای انتخاب ویژگی[1] مورد استفاده قرار گیرد. اگر ورودیهای خاصی برای مسئله مهم نباشد الگوریتم، اتصالات مربوط به آنرا قطع خواهد کرد.



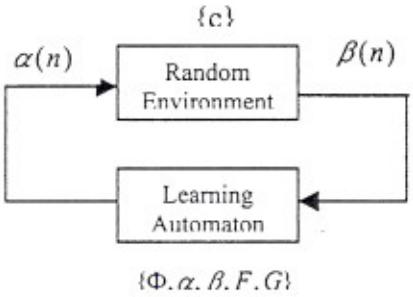
شکل ۱: تغییرات خطای مربوط به الگوهای آزمایشی و آموزشی



شکل ۲: مرزهای تصمیم گیری مربوط به شبکهٔ هرس نشده



شکل ۳: مرز تصمیم گیری مربوط به شبکهٔ هرس شده



شکل ۴ : اتصال فیدبک اتوماتان و محیط

۴- اتوماتان یادگیر

اتوماتان یادگیر[2] (LA) را می توان به دو گروه اصلی اتوماتان یادگیر با ساختار ثابت (FSLA)[3] و اتوماتان یادگیر با ساختار متغیر (VSLA)[4] تقسیم کرد [۳۹] -[۳۶] . اگر احتمال انتقال از یک حالت به حالت دیگر و احتمالهای اقدام و حالت ثابت باشند، اتوماتان با ساختار ثابت، در غیر این صورت اتوماتان با ساختار متغیر نامیده می شود. بعضی از انواع FSLA عبارتند از اتوماتانهای Tsetline ، Krinsky و Krylov . اتوماتان یادگیر با ساختار ثابت یک پنج تایی بصورت $<\underline{\alpha},\underline{\Phi},\underline{\beta},F,G>$ می باشد که در آن داریم:

۱) $\alpha=(\alpha_1,\dots,\alpha_r)$ مجموعهٔ اقدامهاست که در هر مرحله یکی از این اقدامها انتخاب می شود.

۲) $\Phi=(\Phi_1,\dots,\Phi_s)$ مجموعهٔ حالتهاست.

۳) $\beta=\{0,1\}$ مجموعهٔ ورودیهاست که در آن ۱ جریمه و ۰ پاداش را نشان می دهد.

۴) $F:\Phi\times\beta\longrightarrow\Phi$ نگاشت انتقال نامیده می شود. این نگاشت، انتقال حالتهای اتوماتان پس از دریافت ورودی را تعریف می کند. F می تواند استوکستیک[5] باشد.

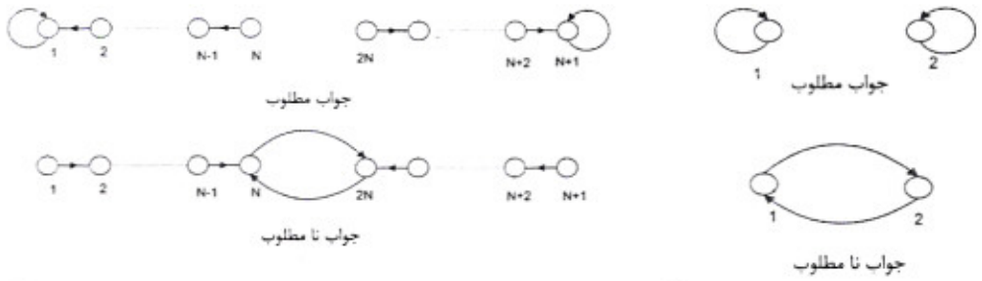
1)Feature
2) Learning Automata
3) Fixed Structure Learning Automata
4) Variable Structure Learning Automata
5) Stochastic

۵) $G:\Phi \longrightarrow \alpha$ نگاشت خروجی بوده و اقدام اتوماتان درحالت Φ_i را نشان میدهد.

اقدام انتخاب شده بعنوان ورودی به محیط اعمال شده و محیط در واکنش به آن پاسخ تصادفی $\beta(n)$ را در لحظهٔ n تولید می کند. $\beta(n)$ عنصری از $\beta=\{0,1\}$ بوده و پاسخ فیدبک محیط، به اتوماتان می باشد. محیط اتوماتان را با احتمال c_i جریمه کرده $(\beta(n)=1)$ که این احتمال وابسته به اقدام می باشد. بر اساس پاسخ $\beta(n)$ ، حالت اتوماتان $\Phi(n)$ تغییر یافته و اقدام جدیدی در لحظهٔ n +1 انتخاب می شود. نحوهٔ اتصال اتوماتان و محیط در شکل (۴) نشان داده شده است.

اتوماتان با دو حالت ($L_{2,2}$)

این اتوماتان دارای دو حالت Φ_1 و Φ_2 و دو اقدام α_1 و α_2 می باشد. اتوماتان از مجموعهٔ {0, 1} ورودی گرفته، با در یافت ورودی ۱ (پاسخ نا مطلوب) تغییر حالت داده و با دریافت ورودی ۰ (پاسخ مطلوب) در همان حالت باقی می ماند. اتوماتانی که از این استراتژی استفاده کند با $L_{2,2}$ نشان داده می شود. زیر نویس اول تعداد حالتها و زیر نویس دوم تعداد اقدامها را نشان می دهد. نحوهٔ انتقال حالت در شکل (۵) نشان داده شده است.



شکل ۵: گراف انتقال حالت برای $L_{2,2}$

شکل ۶: گراف انتقال حالت برای $L_{2N,2}$

اتوماتان دو اقدامی با حافظه ($L_{2N,2}$)

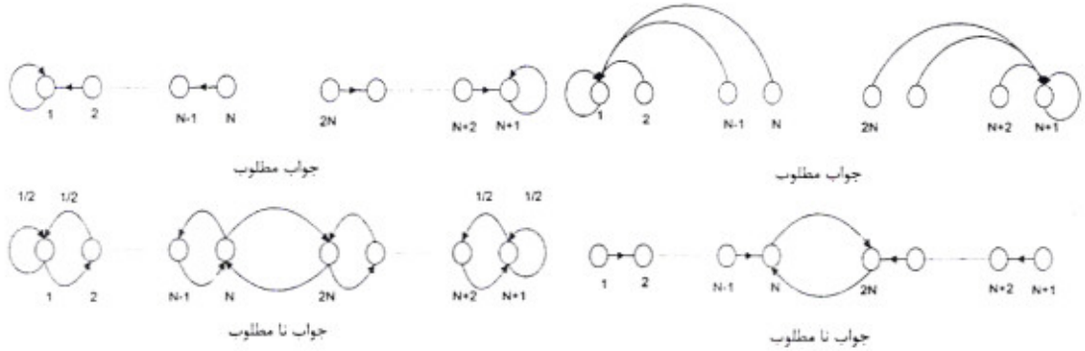
این اتوماتان دارای N ۲ حالت و دو اقدام بوده و رفتار گذشتهٔ سیستم را در قانون تصمیم گیری برای انتخاب رشتهٔ اقدام دخالت میدهد. چنانچه دیدیم اتوماتان $L_{2,2}$ با دریافت پاسخ نامطلوب از محیط، از یک اقدام به اقدام دیگر سوئیچ می کند. در صورتیکه اتوماتان $L_{2N,2}$ وقتی که در داخلی ترین حالت قرار گرفته باشد برای انتقال به اقدام دیگر نیاز به N پاسخ نا مطلوب متوالی دارد. N عمق حافظهٔ اقدام نامیده شده و می گوییم اتوماتان دارای تعداد N ۲ حافظه می باشد. برای هر پاسخ مطلوب، حالت اتوماتان به سمت حالتهای داخلی تر حرکت می کند و به ازای پاسخ نامطلوب به سمت حالتهای بیرونی تر میل میکند. گراف انتقال حالت اتوماتان $L_{2N,2}$ در شکل (۶) نشان داده شده است.

اتوماتان Krinsky

این اتوماتان موقعیکه پاسخ محیط نا مطلوب باشد. دقیقاً مثل $L_{2N,2}$ رفتار می کند. امّا برای پاسخ مطلوب، هر حالت Φ_i (برای I=1,...,N) به حالت Φ_1 می رود و هر حالت Φ_i (برای 2N i =N+1) به حالت Φ_{N+1} می رود. لذا پس از هر پاسخ مطلوب تعداد N پاسخ نامطلوب لازم است تا اتوماتان از یک اقدام به اقدام دیگر سوئیچ کند. گراف انتقال حالت اتوماتان Krinsky در شکل (۷) نشان داده شده است.

اتوماتان Krylov

این اتوماتان موقعیکه خروجی محیط مطلوب باشد انتقال حالت مشابه $L_{2N,2}$ دارد. ولی موقعیکه پاسخ محیط نامطلوب باشد حالت Φ_i ($i \neq 1, N, N+1, 2N$) با احتمال ۰/۵ به حالت Φ_{i+1} و با احتمال ۰/۵ به حالت Φ_{i-1} می رود. موقعیکه۱ = i یا i= N+1 می باشد، Φ_i با احتمال ۰/۵ در همان حالت باقی می ماند. و با همان احتمال به حالت Φ_{i+1} می رود. موقعیکه i= N می باشد اتوماتان با احتمال ۰/۵ به حالت ۲N می رود و با همان احتمال به حالت N-1 می رود و بالاخره موقعیکه i=2N می باشد اتوماتان با احتمال ۰/۵ به حالت N و با همان احتمال به حالت 2N-1 می رود. گراف انتقال حالت این اتوماتان در شکل (۸) نشان داده شده است.



شکل ۷: گراف انتقال حالت برای اتوماتان Krinisky

شکل (۸) : گراف انتقال حالت برای اتوماتان Krylov

اتوماتان با ساختار متغیر:

اتوماتان با ساختار متغیر، احتمالهای انتقال یا احتمالهای اقدام را بر اساس ورودی تغییر می دهد. این نوع اتوماتان توسط شش تایی $< \underline{\beta}, \underline{\Phi}, \underline{\alpha}, \underline{P}, G, T >$ نمایش داده می شود. که در آن $\underline{\beta}$ مجموعهٔ ورودیها، $\underline{\Phi}$ مجموعهٔ حالتهای داخلی، $\underline{\alpha}$ مجموعهٔ خروجیها و $\underline{p}$ بردار احتمال اقدامهاست. که بر اساس آن در هر مرحله اقدامی انتخاب می شود.G نگاشت خروجی و T الگوریتم یادگیری را نشان می دهد. الگوریتم یادگیری یک رابطهٔ برگشتی بوده و برای تغییر دادن بردار احتمال مورد استفاده قرار می گیرد. الگوریتمهای یادگیری مختلفی در مرجع [۳۶] گزارش شده است. الگوریتم یادگیری پاداش- جریمهٔ خطی[1] (L_{R-P}) جزء اولین روشها می باشد. فرض کنیم در لحظهٔ k، اقدام α_i از مجموعهٔ اقدامها با توجه به توزیع p (k) انتخاب شود. در الگوریتم L_{R-P} معادله بر گشتی برای تغییر p به صورت زیر تعریف می شود.

$$p_i(n+1) = p_i(n) + a[1 - p_i(n)] \qquad \beta(n) = 0 \qquad \text{پاسخ مطلوب} \qquad (۵)$$
$$p_j(n+1) = (1-a)p_j(n) \qquad j \neq i$$

$$p_i(n+1) = (1-b)p_i(n) \qquad \beta(n) = 1 \qquad \text{پاسخ نامطلوب} \qquad (۶)$$
$$p_j(n+1) = b/(r-1) + (1-b)p_j(n) \qquad j \neq i$$

پارامترهای a و b طول گـــام را نشـان داده، مقـدار افزایـش (کاهش) احتمال اقدامها را تعیین می کنند. الگوریتم یادگیری دیگری که اکـثراً مورد استفاده قرار می گیرد الگوریتم پاداش - بیحرکت خطی[2] (L_{R-I}) می باشد. در L_{R-I} به ازای پاسخ مطلوب $\beta(n) = 0$ احتمال متناظر بـا اقدام α_i افزایش یافته و بقیهٔ احتمالها کاهش می یابند. ولی به ازای پاسخ نا مطلوب $\beta(n) = 1$ هیچگونه تغییری در احتمالها صورت نمی گیرد. معادلهٔ برگشتی برای تغییرP به صورت زیر می با شد.

$$p_i(n+1) = p_i(n) + a[1 - p_i(n)] \qquad \beta(n) = 0 \qquad \text{پاسخ مطلوب} \qquad (۷)$$
$$p_j(n+1) = (1-a)p_j(n) \qquad j \neq i$$

$$p_j(n+1) = p_j(n) \qquad 1 \leq j \leq r \qquad \beta(n) = 1 \qquad \text{پاسخ نا مطلوب} \qquad (۸)$$

در روابط فوق r تعداد اقدامها را نشان میدهد.

اتوماتان مهاجرت اشیا:

اتوماتـان مـهاجرت اشـیاء [۴۲] توسـط پنـج تـایی $\langle \underline{\alpha}, \underline{\Phi}, \underline{\beta}, \underline{F}, \underline{G} \rangle$ نشـان داده میشـود. کـه $\underline{\alpha} = \{\alpha_1, \alpha_2, ..., \alpha_r\}$ مجموعـهٔ اقدامـــهای مجاز، $\underline{\Phi} = \{\Phi_1, \Phi_2, ..., \Phi_r\}$ مجموعهٔ وضعیتها، $\underline{\beta}\{0,1\}$ مجموعهٔ ورودیها، $F : \Phi \times \beta \to \Phi$ تابع نگاشت وضعیتها و $G : \Phi \to \alpha$ تـابع نگاشت خروجی اتوماتان میباشد. این نوع اتوماتان برای دسته بندی اشیاء ، انتساب حروف به کلیدها ، افراز گراف[3]، تناظر گـراف و تعییـن تعـداد واحدهای مخفی شبکه های عصبی[۴۰] مورد استفاده قرار گرفته است. در ایـن اتوماتـان هـر اقـدام یـک دسـته از الگوهـا را نشـان میدهـد. در اتوماتانهای با ساختار ثابت پاسخ محیط به اتوماتان سبب تغییر وضعیت اتوماتان میگـردد. در صورتیکـه در اتوماتـان مـهاجرت اشـیاء، اشـیاء بـه وضعیت های اتوماتان انتساب داده میشوند و پاسخ محیط به اتوماتان سبب گردش اشیاء در بین وضعیت هـای اتوماتـان میگـردد. از طریـق ایـن گردش طبقه بندی اشیاء صورت میگیرد. اگرشئ W_i در اقدام شمارهٔj اتوماتان مهاجرت اشیاء قرار داشته باشد. این شئ متعلق به دستهٔ شمارهٔ j است. برای اقدام α_k مجموعهٔ وضعیت $\{\Phi_{(k-1)N+1}, ..., \Phi_{kN}\}$ در نظر گرفته میشود. کهN عمق حافظـه را نشـان میدهـد. بـدون از دسـت دادن عمومیت بحث میتوان $\Phi_{(k-1)N+1}$ را داخلی ترین وضعیت و Φ_{kN} را خارجی ترین وضعیت این اقدام در نظر گرفت. اگردو W_i و W_j بـترتیب در وضعیتهای $\Phi_{(k-1)N+1}$ و $\Phi_{(k-1)N+m}$ (برای m>1) قرار داشته باشند در اینصورت احتمال تعلق شئ W_i به این دسته از احتمـال تعلـق شـئ W_j بیشتر است. بنابر این برای اقدام α_k، وضعیت $\Phi_{(k-1)N+1}$، وضعیت با بیشترین احتمال و وضعیـت Φ_{kN}، وضعیـت بـا کمـترین احتمـال نـامیده میشود.

۵- الگوریتم پیشنهادی

در این قسمت ابتدا بطور مختصر الگوریتم کاهش تعداد وزن ارائهٔ شده در مرجع [۴۲] را شرح داده در ادامه، الگوریتم جدیدی را ارائه میکنیم.

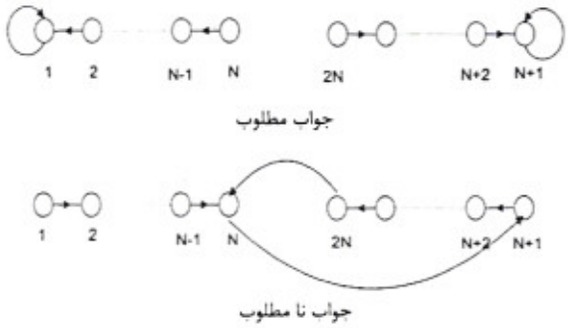
الگوریتم کـــاهش تعـداد وزن: در ایـن الگوریتـم از یـک اتوماتـان اسـتفاده شـده اسـت بـرای سـادگـی در ارائـهٔ مطـالب اتوماتـان بصـورت $HWLA(2,N,W)$ نـشان داده شـده اسـت که در آن عدد ۲ تعداد اقدامها، N عمق حافظه و W وزنهای ورودی لایهٔ مخفی را نشان میدهـد. ایـن اتـوماتان برای تعیین تعداد وزنهای لایـهٔ مخفـی اسـت کـه دارای دو اقـدام و عمـق حافظـهٔ N اسـت. و روی وضعیـت هـای آن مجموعـهٔ وزن $W = \{w_1, w_2, w_3, ...\}$ قرار دارد. گراف تغییر حالت برای اتوماتان تعیین تعداد وزنهای لایهٔ مخفی شبکه در شکل ۹ نشان داده شده اسـت. این الگوریتم نحوهٔ اتصال وزنهای بین لایه ورودی و لایه مخفی را مشخص میکند. در ابتدا همهٔ وزنـهای موجـود در شـبکه در وضعیـت Φ_1 قـرار میگیرند. به همهٔ وزنهای روشن مدتی اجازه داده میشود تا در آموزش شبکه شرکت نمایند. وزنهایی که عملکرد آنها در این مدت مناسـب نیسـت

1) Linear Reward-Penalty
2) Linear Reward-Inaction
3) Graph Partitioning

جریمه میشوند، وزنهایی که عملکرد آنها خیلی خوب است پاداش میگیرند و وزنهایی که در مورد آنها نمیتوان تصمیم گیری قطعی نمود بدون تغییر وضعیت باقی می مانند. برای تشخیص خوب یا بد بودن عملکرد یک وزن از قدرت انتشار این وزن (قدر مطلق مقدار وزن) استفاده میشود که توسط دو قانون مکاشفهای زیر بیان میشود.

عملکرد خوب یک وزن: وزنی دارای عملکرد خوب میباشد که سیگنال وارد شده به این وزن بمیزان زیادی تضعیف نگردد و یا به عبارتی دیگر قدر مطلق آن کوچک نباشد.

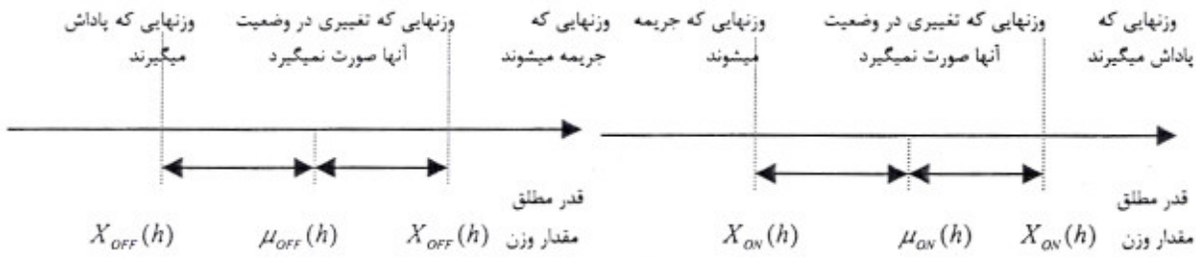
عملکرد بد یک وزن: وزنی دارای عملکرد بد میباشد اگر سیگنال وارد شده به این وزن بمیزان زیادی تضعیف گردد و یا به عبارتی دیگر قدر مطلق آن کوچک باشد.



شکل ۹: اتوماتان $HWLA(2,N,1)$

تشخیص نحوهٔ عملکرد یک وزن روشن: الگوریتم پس انتشار خطا را به دفعات معینی تکرار میکنیم در انتها اگر قدر مطلق وزنی از یک مقدار آستانه ای بیشتر باشد آن وزن مناسب بوده و اگر از یک مقدار آستانه ای دیگر کوچک باشد نا مناسب میباشد. برای تعیین مقادیر آستانه ای از واریانس قدر مطلق مقدار وزنها استفاده شده است. برای وزنهای مربوط به هر کدام از نرونها، واریانس بصورت زیر محاسبه میگردد.

$$\sigma_h = \sqrt{\frac{\sum_{W_{hk} \in ON_w(h)} \left(|W_{hk}| - \mu_W(h)\right)^2}{|ON_w(h)|}} \qquad h \in ON \qquad (۹)$$



شکل ۱۰: تشخیص مقدار آستانه وزنهای روشن *شکل ۱۱: تشخیص مقدار آستانه وزنهای خاموش*

که در آن ON مجموعهٔ واحدهای روشن، $ON_w(h)$ مجموعهٔ وزنهای ورودی روشن متصل به واحد مخفیh و $\mu_W(h)$ میانگین قدر مطلق مقدار وزنهای روشن متصل به واحد مخفی h میباشد. پس از محاسبهٔ واریانس، وزنهای روشنی که قدر مطلق مقدار آنها از یک مقدار آستانه ای کمتر باشد جریمه میشوند و وزنهای روشنی که قدر مطلق آنها از یک مقدار آستانه ای دیگر بیشتر باشد پاداش میگیرند و وزنهای روشنی که قدر مطلق مقدار آنها بین دو مقدار آستانه ای قرار میگیرند تغییری در وضعیت آنها ایجاد نمیشود (شکل ۱۰). $X_{ON}(h)$ پهنای روشنی نامیده شده و بصورت زیر محاسبه میگردد.

$$X_{ON}(h) = \lambda_{ON} \frac{OFF_w(h)}{|\#INPUT + 1|} \sigma_h \qquad (۱۰)$$

ثابت ۰ < λ_{ON} ضریب پهنای روشنی، #INPUT تعداد واحدهای لایهٔ ورودی و $OFF_w(h)$ مجموعهٔ وزنهای خاموش متصل به واحد مخفی روشن h است.

تشخیص نحوهٔ عملکرد یک وزن خاموش: یک وزن خاموش در آموزش شرکت نمیکند. بنابر این مقدار آنرا از آخرین زمان روشن بودن آن محاسبه میکنیم. اگر یک وزن برای مدت طولانی خاموش باشد از اهمیت و در نتیجه از مقدار آن کاسته میشود. بنابر این مقدار یک وزن خاموش بصورت زیر محاسبه میشود.

$$W_{hk}(n+1) = W_{hk}(n)\lambda_d \qquad (۱۱)$$

که ثابت ۱ > λ_d > ۰ ضریب کاهش مقدار وزن نامیده میشود. به این ترتیب مقدار یک وزن خاموش بتدریج کم میشود. واریانس وزنهای خاموش بصورت زیر محاسبه میشود.

$$\sigma_h = \sqrt{\frac{\sum_{W_{uh} \in OFF_u(h)} \left(|W_{uh}| - \mu_{OFF}(h)\right)^2}{|OFF_u(h)|}} \qquad h \in ON \tag{۱۲}$$

که در آن ON مجموعهٔ واحدهای روشن و $\mu_{OFF}(h)$ میانگین قدر مطلق مقدار وزنهای خاموش متصل به واحد مخفی روشن h میباشد. با استفاده از واریانس و طبق فرمول زیر $X_{OFF}(h)$ که پهنای خاموشی نامیده میشود بصورت زیر محاسبه میگردد.

$$X_{OFF}(h) = \lambda_{OFF} \frac{ON_u(h)}{|\#INPUT + 1|} \sigma_h \tag{۱۳}$$

که ثابت ۱ > λ_{OFF} > ۰ ضریب پهنای خاموشی و $\#INPUT$ تعداد واحد های لایهٔ ورودی است. وزنهایی که قدر مطلق مقدار آنها از $\mu_{OFF}(h) - X_{OFF}(h)$ کمتر باشد پاداش میگیرند. وزنهایی که قدر مطلق آنها از $\mu_{OFF}(h) + X_{OFF}(h)$ بیشتر باشند جریمه میشوند. و وزنهایی که قدر مطلق آنها در فاصلهٔ[$\mu_{OFF}(h) - X_{OFF}(h)$, $\mu_{OFF}(h) + X_{OFF}(h)$] قرار دارند و ضعیت آنها تغییر نمیکند(شکل ۱۱).

الگوریتم تعیین تعداد وزن تطبیقی: مقدار $X_{ON}(h)$ که بازهٔ تصمیم گیری را نشان میدهد تأثیر بسزائی در عملکرد الگوریتم کاهش تعداد وزن دارد. وزنهای روشنی که قدر مطلق آنها کمتر از $\mu_{ON}(h) - X_{ON}(h)$ می باشد جریمه شده و وزنهای روشنی که قدر مطلق آنها بیشتر از $\mu_{ON}(h) + X_{ON}(h)$ می باشد پاداش میگیرند. هر قدر مقدار بازهٔ $X_{on}(h)$ کم باشد وزنهای بیشتری جریمه شده در نتیجه وزنها با سرعت بیشتری ریزش خواهند کرد. اگر این سرعت از یک حدی بیشتر باشد شبکه دیگر قادر به یادگیری الگوهای آموزشی نخواهد بود. اگر مقدار $X_{ON}(h)$ خیلی زیاد انتخاب شود شرط جریمه کردن وزنهای روشن خیلی سنگین شده، در نتیجه وزنی خاموش نشده یا تعداد بسیار کمی وزن خاموش خواهد شد. در نتیجه شاهد هیچگونه کاهشی در پیچیدگی شبکه نخواهیم بود. پس ملاحظه میکنیم مقدار $X_{ON}(h)$ خیلی مهم بوده و با انتخاب یک مقدار بهینه برای $X_{ON}(h)$ میتوانیم در کمترین زمان به شبکه هایی با حداقل تعداد وزن که بتواند با خطای قابل قبولی نیز الگوهای آموزش را یاد بگیرد برسیم. پارامتر $X_{ON}(h)$ براساس فرمول زیرتابع چندین متغیر می باشد..

$$X_{ON}(h) = \lambda_{ON} \frac{OFF_u(h)}{|\#INPUT + 1|} \sigma_{ON}(h) \tag{۱۴}$$

#INPUT بعد فضای ورودی بوده و برای یک مسئلهٔ خاص مقدار ثابتی می باشد. $OFF_u(h)$ تعداد وزنهای خا موش را نشان داده و بعنوان یک عامل بازدارنده در مقابل ریزش بیش از حد وزنها عمل میکند. با حذف تدریجی وزنها مقدار عبارت $OFF_u(h)$ افزایش یافته و باعث افزایش $X_{ON}(h)$ میگردد. با افزایش $X_{ON}(h)$ از سرعت ریزش وزنها کاسته میشود. λ_{ON} مقدار ثابتی بوده و ضریب پهنای روشنی نامیده میشود. $\sigma_{ON}(h)$ واریانس وزنهای روشن مربوط به نرون مخفی h میباشد. بحثهائی که در مورد بازهٔ $X_{ON}(h)$ مطرح کردیم در مورد بازهٔ $X_{OFF}(h)$ نیز قابل طرح میباشد. $X_{OFF}(h)$ که بازهٔ تصمیم گیری برای وزنهای خاموش را نشان میدهد تأثیر بسزائی در عملکرد الگوریتم کاهش تعداد وزن دارد. وزنهای خاموشی که قدر مطلق آنها کمتر از $\mu_{OFF}(h) - X_{OFF}(h)$ می باشد پاداش میگیرند و وزنهای خاموشی که قدر مطلق آنها بیشتر از $\mu_{OFF}(h) + X_{OFF}(h)$ می باشد جریمه میشوند. هر قدر مقدار بازهٔ $X_{uFF}(h)$ کم باشد وزنهای خاموش بیشتری جریمه شده در نتیجه وزنهای خاموش با سرعت بیشتری شروع به روشن شدن میکنند و اگر مقدار $X_{ON}(h)$ خیلی زیاد انتخاب شود شرط جریمه کردن وزنهای خاموش خیلی سنگین شده، در نتیجه وزنی روشن نشده یا تعداد بسیار کمی وزن روشن میشوند. در نتیجه آنجائیکه نیاز داریم وزنهایی به شبکه برگردانده شوند وزنی روشن نمیشود. پس ملاحظه میکنیم مقدار $X_{uFF}(h)$ خیلی مهم بوده و با انتخاب یک مقدار بهینه برای $X_{OFF}(h)$ میتوانیم در کمترین زمان به شبکه هایی ا حداقل تعداد وزن که بتواند با خطای قابل قبولی نیز الگوهای آموزش را یاد بگیرد برسیم. پارامتر $X_{OFF}(h)$ براساس فرمول زیرتابع چندین متغیر می باشد..

$$X_{OFF}(h) = \lambda_{OFF} \frac{ON_u(h)}{|\#INPUT + 1|} \sigma_{OFF}(h) \tag{۱۵}$$

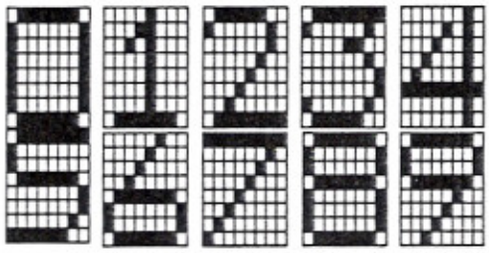
#INPUT بعد فضای ورودی بوده و برای یک مسئلهٔ خاص مقدار ثابتی می باشد. $ON_u(h)$ تعداد وزنهای روشن را نشان داده و بعنوان یک عامل بازدارنده در مقابل برگشت بیش از حد وزنها عمل میکند. λ_{OFF} مقدار ثابتی بوده و ضریب پهنای خاموشی نامیده میشود. $\sigma_{OFF}(h)$ واریانس وزنهای خاموشن مربوط به نرون مخفی h میباشد.

سطح خطا شبکه های MLP با الگوریتم یادگیری BP، برای حالتهائیکه درجهٔ سطوح از درجهٔ دوم به بالا میباشد بسیار پیچیده بوده و دارای مناطق مسطح و مناطق با شیب تند زیادی میباشد. علاوه براین چون با اعمال الگوریتم کاهش تعداد وزن، توپولوژی و تعداد وزنها دائماً در حال تغییر میباشد لذا سطح خطا دائماً در حال تغییر بوده، نتیجتاً شکل آن بسیار پیچیده و تصادفی خواهد بود در نتیجه در نقاط مختلف سطح خطا و در طول یادگیری حساسیت شبکه نسبت به حذف وزن متفاوت خواهد بود در بعضی از نقاط حساسیت شبکه کم بوده و میتوان با سرعت بیشتری عمل ریزش وزن را انجام داد و در برخی نقاط حساسیت زیاد بوده و سرعت بالای ریزش وزن، خطرناک بوده و میتواند باعث ناپایدار ی گردد. لذا برای دستیابی به شبکه هایی با کمترین پیچیدگی و درکمترین زمان، نیاز به الگوریتمهایی داریم که بتواند بطور پویا سرعت عمل شاخه

زنی و بر گشت وزن را بر اساس معیاری تغییردهد. چنانچه قبلاً توضیح دادیم سرعت عمل شاخه زنی وابسته به پارامتر $X_{ON}(h)$ و سرعت عمل برگشت وزن تابعی از پارامتر $X_{OFF}(h)$ می باشد و با تغییر آنها میتوان سرعت شخه زنی و سرعت روشن شدن وزنهای خاموش را کنترل کرد.

```
Input:
    Training Patterns (p,t), p is input and t is desired output
    No. of Hidden Units H
Output:
    Network Weight Vector:W
    Network Topology
Iinitialize Weights and baises, Automata Parameters and
Learning Parameters
for i=1:100
    for i=1:K
        Call BP
    end
    MinimumOfError_new=minimum(Error Vector);
    Select an Action from on_width_coefficient set
    Select an Action from off_width_coefficient set
    for all Hidden Neurons
        for all weights
            if W_state >= N+1 % if weight is OFF
                W = W*WEIGHT_DECAY_COEFFICIEN
            end
        end
        Compute Xon, Xoff
        for all weights
            if W_state >= N+1 % if weight is OFF
                if abs(W) < Average_off- Xoff
                    if W_state ~= (N+1)
                        W_state = W_state - 1
                    elseif W_state == (N+1)
                        W_state = W_state
                    end
                elseif abs(W) > Average_off + Xoff
                    if W_state ~= (2*N)
                        W_state = W_state + 1 ;
                    elseif W_state == (2*N)
                        W_state = N;
                    end
                end
            else % if weight is ON.
                if abs(W) < Average_on - Xon
                    if W_state== N
                        W_state = N+1;
                    else
                        W_state= W_state +1;
                    end
                elseif abs(W) > Average_on + Xon
                    if W_state > 1
                        W_state = W_state - 1 ;
                    else
                        W_state = W_state;
                    end
                end
            end
        end
    end
    if  MinOfError_new / MinOfError  > MAX_ERR_RATIO
        EnvironmentResponce=1;
    else
        EnvironmentResponce=0;
    end
    Update_Tsetlin_State_ λON (EnvironmentResponce)
    Update_Tsetlin_State_ λOFF (EnvironmentResponce)
end
```

شکل ۱۲: الگوریتم تعیین وزن تطبیقی



شکل ۱۳: نحوهٔ نمایش اعداد از ۰ الی ۹

جدول (۱): نحوهٔ کد کردن اعداد از ۰ الی ۹

عدد	بیت ۳	بیت ۲	بیت ۱	بیت ۰
۰	-۱	-۱	-۱	-۱
۱	-۱	-۱	-۱	۱
۲	-۱	-۱	۱	-۱
۳	-۱	-۱	۱	۱
۴	-۱	۱	-۱	-۱
۵	-۱	۱	-۱	۱
۶	-۱	۱	۱	-۱
۷	-۱	۱	۱	۱
۸	۱	-۱	-۱	-۱
۹	۱	-۱	-۱	۱

۰۱۲۳۴۵۶۷۸۹

شکل ۱۴: اعداد چاپی فارسی

برای تنظیم و تطبیق $X_{ON}(h)$ و $X_{OFF}(h)$ میتوان بترتیب از ضرائب λ_{ON} و λ_{OFF} استفاده کرد. در این مقاله با استفاده از اتوماتانهای یادگیر ضرائب λ_{ON} و λ_{OFF} را تطبیق داده ایم. الگوریتم پیشنهادی بصورت زیر عمل میکند. در ابتدا تمامی وزنهای موجود در شبکه در وضعیت Φ_1 قرار میگیرند. به همهٔ وزنهای روشن مدتی اجازه داده میشود تا در آموزش شبکه شرکت نمایند. وزنهائی که عملکرد آنها در این مدت مناسب نیست جریمه میشوند، وزنهائی که عملکرد آنها خیلی خوب است پاداش میگیرند و وزنهائی که در مورد آنها نمیتوان تصمیم گیری قطعی نمود بدون

تغییر وضعیت باقی میمانند. برای پاسخ دادن به اتوماتان مربوط به تنظیم پارامترهای ضرائب پهنای روشنی وخاموشی (λ_{OFF} و λ_{ON}) از معیار خطا کمک میگیریم. مینیمم مقدار خطا در تکرار فعلی را با مینیمم مقدار خطا در تکرار قبلی مقایسه میکنیم اگر با یک ضریبی خطا افزایش پیدا کرده بود اتوماتان را جریمه میکنیم در غیر اینصورت به آن پاداش میدهیم. برای تطبیق هر کدام از پارامترهای λ_{OFF} و λ_{ON} از اتوماتان Tsetline استفاده کرده ایم. الگوریتم جدید را که تحت عنوان الگوریتم تعیین تعداد وزن تطبیقی می نامیم در شکل ۱۲ نشان داده شده است.

۶- نتایج شبیه سازی

در این قسمت ۳ مسئلهٔ نمونه که الگوریتم پیشنهادی روی آنها پیاده سازی شده است معرفی میگردد. در ادامه نتایج آزمایشات ارائه خواهد شد.

الف) مسئلهٔ شناسایی اعداد لاتین: در این مسئله می خواهیم شبکه ای را آموزش بدهیم که بتواند اعداد از ۰ تا۹ را تشخیص دهد. برای این منظور اعداد را مطابق شکل (۱۳) نمایش میدهیم. برای هر عدد یک ماتریس ۸×۸ در نظر می گیریم. خانه های سیاه را با ۱ و خانه های سفید را با ۱- نشان میدهیم. اعداد ۰ تا ۹ را می توان مطابق جدول (۱) با چهار بیت کُد کرد. لذا خروجی شبکه دارای ۴ نرون خواهد بود. از یک شبکهٔ سه لایه با ۶۴ نرون در لایهٔ ورودی و ۴ نرون در لایهٔ خروجی استفاده کرده ایم. برای آزمایشهای مختلف تعداد نرونهای لایهٔ خروجی را بین ۱۵ و ۲۳ تغییر میدهیم.

ب) مسئلهٔ تقریب تابع غیر خطی زمان گسستهٔ درجهٔ دوّم: تابع غیر خطی زمان گسسته از مرتبهٔ دوّم با رابطهٔ زیر مفروض می باشد.

$$y_{k+1} = \frac{1.5 y_k y_{k-1}}{1 + y_k^2 + y_{k-1}^2} + 0.35(y_k + y_{k-1}) + 1.2u_k \qquad (۱۶)$$

می خواهیم با استفاده از یک شبکهٔ عصبی سه لایه تابع فوق را با تقریب قابل قبولی شبیه سازی کنیم. برای این منظور ورودیهایی بین ۱- و ۱+ بطور تصادفی ایجاد کرده و به تابع اعمال می کنیم شرایط اولیّه را نیز بطور تصادفی بین ۱- و ۱+ انتخاب می کنیم بدین ترتیب یک سری الگوی آموزش ایجاد می شود که از این الگو ها برای آموزش شبکه استفاده می کنیم. در رابطه فوق u_k و y_k به ترتیب ورودی و خروجی در لحظهٔ k بوده و y_{k-1} و y_{k+1} به ترتیب خروجی در لحظات k-1 و k+1 می باشد. برای تقریب تابع از یک شبکهٔ عصبی سه لایه با ۳ نرون در لایهٔ ورودی و ۱ نرون در لایهٔ خروجی استفاده کرده ایم. تعداد نرونهای لایه میانی را برای آزمایشهای مختلف بین ۸ و ۲۵ تغییر میدهیم.

ج) مسئلهٔ تشخیص اعداد چاپی فارسی:

در این مسئله هدف تشخیص اعداد چاپی فارسی میباشد (شکل۱۴). این اعداد روی یک صفحه کاغذ چاپ شده اند. اعداد فارسی از طریق یک دستگاه پویشگر با قدرت تفکیک ۳۰۰ نقطه در اینچ نمونه برداری گردیده است.

پس از جدا سازی حروف در تصویر، ثابت های گشتاور M_1 تا M_7 که توسط روابط زیر نشان داده شده اند استخراج شده و ۷ ویژگی ورودیهای شبکه عصبی را تشکیل میدهند.

$$M_1 = \mu'_{20} + \mu'_{02}\,,\quad M_2 = (\mu'_{20} - \mu'_{02})^2 + 4(\mu'_{11})^2\,,\quad M_3 = (\mu'_{30} - 3\mu'_{12})^2 + (\mu'_{21} - 3\mu'_{03})^2$$

$$M_4 = (\mu'_{30} + 3\mu'_{12})^2 + (\mu'_{21} + 3\mu'_{03})^2$$

$$M_5 = (\mu'_{30} - 3\mu'_{12})(\mu'_{30} + \mu'_{12})\left[(\mu'_{30} + \mu'_{12})^2 - 3(\mu'_{21} + \mu'_{03})^2\right] + (3\mu'_{21} - 3\mu'_{03})(\mu'_{21} + \mu'_{03})\left[3(\mu'_{30} + \mu'_{12})^2 - (\mu'_{21} + \mu'_{03})^2\right]$$

$$M_6 = (\mu'_{20} - 3\mu'_{02})\left[(\mu'_{30} + \mu'_{12})^2 - (\mu'_{21} + \mu'_{03})^2 + 4\mu'_{11}(\mu'_{30} + \mu'_{12})(\mu'_{21} + \mu'_{03})\right]$$

$$M_7 = (3\mu'_{21} - \mu'_{03})(\mu'_{30} + 3\mu'_{12})\left[(\mu'_{30} + \mu'_{12})^2 - 3(\mu'_{21} + \mu'_{03})^2\right] - (\mu'_{30} - 3\mu'_{12})(\mu'_{21} + \mu'_{03})\left[3(\mu'_{30} + \mu'_{12})^2 - (\mu'_{21} + \mu'_{03})^2\right]$$

که μ'_{pq} گشتاورهای مقیاس شده از مرتبه p+q برای تصویر یک حرف میباشد. در ادامه نتایج شبیه سازیهای انجام گرفته شده برای مسئله های فوق آورده شده است.

الف)الگوریتم تعیین تعداد وزن:در این الگوریتم همچنانکه قبلاً توضیح دادیم در ابتدا همهٔ وزنهای موجود در شبکه در وضعیت Φ_1 قرار میگیرند. به همهٔ وزنهای روشن مدتی اجازه داده میشود تا در آموزش شبکه شرکت نمایند. وزنهایی که عملکرد آنها در این مدت مناسب نیست جریمه میشوند، وزنهایی که عملکرد آنها خیلی خوب است پاداش میگیرند و وزنهایی که در مورد آنها نمیتوان تصمیم گیری قطعی نمود بدون تغییر وضعیت باقی می مانند. در این الگوریتم در طی آموزش ضرائب پهنای روشنی و پهنای خاموشی مقادیر ثابتی میباشند.

مسئلهٔ شناسایی اعداد لاتین: الگوریتم تعیین تعداد وزن برای ۶ شبکهٔ مختلف با شرایط اولیهٔ تصادفی و با تعداد نرونهای لایهٔ مخفی متفاوت پیاده سازی شده و نتایجدر جدول ۲ نشان داده شده است. برای تمامی شبکه ها از اتوماتان (HWLA(2, 7, N_w استفاده شده است. N_w تعداد وزنهای لایهٔ مخفی شبکه در ابتدای الگوریتم می باشد.ضرائب پهنای روشنی و خاموشی ۱۰، ضریب کاهش وزن ۰/۹ ، تعداد تکرار الگوریتم BP در هر گام ۵۰ ، نرخ یادگیری ۰/۰۱ و ضریب ممنتم صفر در نظر گرفته شده است. از تابع غیر خطی سیگموئید که بین ۱- و ۱+ تغییر میکند بعنوان توابع لایهٔ مخفی و لایهٔ خروجی استفاده شده است. الگوهای آموزش ۵۰۰۰ بار به شبکه اعمال شده است.

مسئلهٔ تقریب تابع غیر خطی زمان گسستهٔ درجهٔ دوّم: الگوریتم تعیین تعداد وزن برای ۶ شبکهٔ مختلف با شرایط ا ولیهٔ تصادفی و با تعداد نرونهای لایهٔ مخفی متفاوت پیاده سازی شده و نتایج در جدول ۳ نشان داده شده است. برای تمامی شبکه ها از اتوماتان (HWLA(2, 7, N_w استفاده شده است. N_w تعداد وزنهای لایهٔ مخفی شبکه در ابتدای الگوریتم می باشد. ضرائب پهنای روشنی و خاموشی ۱۰، ضریب کاهش وزن ۰/۹ ، تعداد تکرار الگوریتم BP در هر گام ۵۰ ، نرخ یادگیری ۰/۰۱ و ضریب ممنتم ۰/۹۸ در نظر گرفته شده است. از تابع غیر خطی سیگموئید که بین ۱- و ۱+ تغییر میکند بعنوان تا بع لایهٔ مخفی و از تابع خطی بعنوان تابع لایهٔ خروجی استفاده شده است. الگوهای آموزش ۵۰۰ بار به شبکه اعمال شده است.

مسئلهٔ تشخیص اعداد چاپی فارسی: الگوریتم تعیین تعداد وزن برای ۶ شبکهٔ مختلف با شرایط ا ولیهٔ تصادفی و با تعداد نرونهای لایهٔ مخفی متفاوت پیاده سازی شده و نتایج در جدول ۴ نشان داده شده است. برای تمامی شبکه ها از اتوماتان (HWLA(2, 20, N_w استفاده شده است. N_w تعداد وزنهای لایهٔ مخفی شبکه در ابتدای الگوریتم می باشد. ضرائب پهنای روشنی و خاموشی ۵، ضریب کاهش وزن ۰/۹ ، تعداد تکرار الگوریتم BP در هر گام ۵۰ ، نرخ یادگیری ۰/۰۱ و ضریب ممنتم ۰/۹۸ در نظر گرفته شده است. از تابع غیر خطی سیگموئید که بین ۱- و ۱+ تغییر میکند بعنوان توابع لایهٔ مخفی و لایهٔ خروجی استفاده شده است.

ب)الگوریتم تعیین تعداد وزن تطبیقی: الگوریتم تعیین تعداد وزن تطبیقی بصورت زیرعمل میکند. درابتدا تمامی وزنهای موجود در شبکه در وضعیت Φ_1 قرار میگیرند. به همهٔ وزنهای روشن مدتی اجازه داده میشود تا در آموزش شبکه شرکت نمایند. وزنهائی که عملکرد آنها در این مدت مناسب نیست جریمه میشوند، وزنهائی که عملکرد آنها خیلی خوب است پاداش میگیرند و وزنهائی که در مورد آنها نمیتوان تصمیم گیری قطعی نمود بدون تغییر وضعیت باقی میمانند. در این روش بمنظور دستیابی به شبکه هائی با حداقل تعداد لینک بین ورودی و نرونهای لایهٔ مخفی، ضرائب پهنای روشنی و خاموشی در طول آموزش تغییر داده میشوند. برای تطبیق این ضرائب از اتوماتان Tsetline استفاده شده است. برای پاسخ دادن به اتوماتان مربوط به تنظیم پارامترهای ضرائب پهنای روشنی وخاموشی (λ_{OFF} و λ_{ON}) از معیار خطا کمک میگیریم. مینیمم مقدار خطا در تکرار فعلی را با مینیمم مقدار خطا در تکرار قبلی مقایسه میکنیم اگر با یک ضریبی خطا افزایش پیدا کرده بود اتوماتان را جریمه میکنیم در غیر اینصورت به آن پاداش میدهیم.

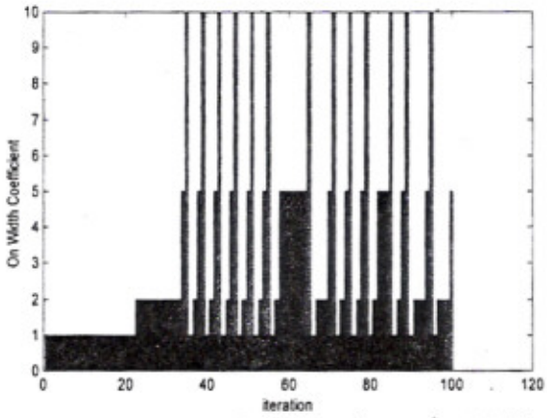
مسئلهٔ شناسایی اعداد لاتین: الگوریتم تطبیقی برای ۶ شبکهٔ مختلف با شرایط ا ولیهٔ تصادفی و با تعداد نرونهای لایهٔ مخفی متفاوت پیاده سازی شده و نتایج در جدول ۲ نشان داده شده است. شکلهای ۱۵و ۱۶ نمودار تغییرات ضریب پهنای روشنی و ضریب پهنای خاموشی را نشان میدهد. برای تمامی شبکه ها از اتوماتان (HWLA(2, 7, N_w برای وزنها و از اتوماتان Tsetline(4, 5) برای تطبیق ضرائب پهنای روشنی و خاموشی استفاده شده است. N_w تعداد وزنهای لایهٔ مخفی شبکه در ابتدای الگوریتم می باشد. مجموعهٔضرائب پهنای روشنی {۱۰ ,۵ ,۲ ,۱}، مجموعهٔ ضرائب پهنای خاموشی{۲۵ ,۲۰ ,۱۵ ,۱۰}، ضریب کاهش وزن ۰/۹ ، تعداد تکرار الگوریتم BP در هر گام ۵۰ ، نرخ یادگیری ۰/۰۱ و ضریب ممنتم صفر در نظر گرفته شده است. از تابع غیر خطی سیگموئید که بین ۱- و ۱+ تغییر میکند بعنوان توابع لایهٔ مخفی و لایهٔ خروجی استفاده شده است. الگوهای آموزش ۵۰۰۰ بار به شبکه اعمال شده است.

مسئلهٔ تقریب تابع غیرخطی زمان گسستهٔ درجهٔ دوّم: الگوریتم تطبیقی برای ۶ شبکهٔ مختلف با شرایط ا ولیهٔ تصادفی و با تعداد نرونهای لایهٔ مخفی متفاوت پیاده سازی شده و نتایج در جدول ۳ نشان داده شده است. در الگوریتم تطبیقی بیشترین خطا مربوط به شبکهٔ شمارهٔ ۳ میباشد که مقدار آن ۲/۲۰۸۲ میباشد در این حالت به ازای ورودی سینوسی شکل موج خروجی تابع و خروجی شبکه در شکلهای ۱۷و ۱۸ به ازای دو فرکانس مختلف نشان داده شده است. چنانچه از روی شکلها مشخص میباشد خروجی شبکهٔ عصبی بخوبی خروجی تابع را دنبال میکند. برای تمامی شبکه ه ها از اتوماتا(HWLA(2, 7, N_w برای وزنها و از اتوماتان Tsetline(4, 5) برای تطبیق ضرائب پهنای روشنی و خاموشی ا ستفاده شده است. N_w تعداد وزنهای لایهٔ مخفی شبکه در ابتدای الگوریتم می باشد. مجموعهٔ ضرائب پهنای روشنی ۱{۰ ,۸ ,۶ ,۴}، مجموعهٔ ضرائب پهنای خاموشی {۱۶ ,۱۴ ,۱۲ ,۱۰}، ضریب کاهش وزن ۰/۹ ، تعداد تکرار الگوریتم BP در هر گام ۵۰ ، نرخ یادگیری ۰/۰۱ و ضریب ممنتم ۰/۹۸ در نظر گرفته شده است. از تابع غیر خطی سیگموئید که بین ۱- و ۱+ تغییر میکند بعنوان تا بع لایهٔ مخفی و از تابع خطی بعنوان تابع لایهٔ خروجی استفاده شده است. الگوهای آموزش ۵۰۰۰ بار به شبکه اعمال شده است.

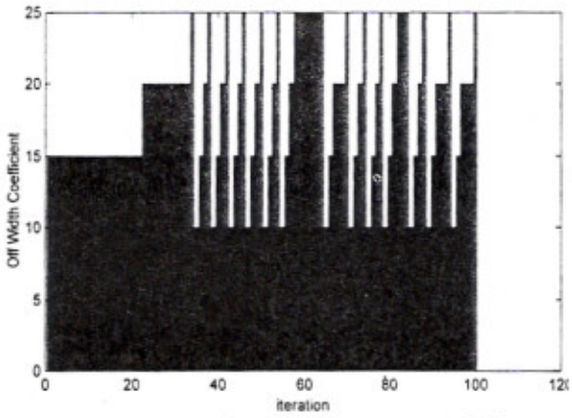
مسئلهٔ تشخیص اعداد چاپی فارسی: الگوریتم تطبیقی برای ۴ شبکهٔ مختلف با شرایط ا ولیهٔ تصادفی و با تعداد نرونهای لایهٔ مخفی متفاوت پیاده سازی شده و نتایج در جدول ۴ نشان داده شده است. برای تمامی شبکه ها از اتوماتان (HWLA(2, 20, N_w برای وزنها و از اتوماتان Tsetline(4, 5) برای تطبیق ضرائب پهنای روشنی و خاموشی ا ستفاده شده است. N_w تعداد وزنهای لایهٔ مخفی شبکه در ابتدای الگوریتم می باشد. مجموعهٔ ضرائب پهنای روشنی {۵ ,۴/۵ ,۴ ,۳/۵}، مجموعهٔ ضرائب پهنای خاموشی {۱۸ ,۱۷ ,۱۶ ,۱۵}، ضریب کاهش وزن ۰/۹ ، تعداد تکرار الگوریتم BP در هر گام ۵۰ ، نرخ یادگیری ۰/۰۱ و ضریب ممنتم ۰/۹۸ در نظر گرفته شده است. از تابع غیر خطی سیگموئید که بین ۱- و ۱+ تغییر میکند بعنوان تا بع لایهٔ مخفی و لایهٔ خروجی استفاده شده است

جدول ۲: مقایسهٔ الگوریتم غیر تطبیقی با الگوریتم تطبیقی برای مسئلهٔ شناسایی اعداد لاتین

شبکه	واحد مخفی	وزنهای لایهٔ مخفی (الگوریتم غیر تطبیقی)	در صد تشخیص (آموزش)	در صد تشخیص (آزمایشی)	وزنهای لایهٔ مخفی (الگوریتم تطبیقی)	در صد تشخیص (آموزش)	در صد تشخیص (آزمایشی)
۱	۱۵	۴۹۹	٪۱۰۰	٪۱۰۰	۲۰۶	٪۱۰۰	۱۰۰
۲	۱۵	۵۳۴	٪۱۰۰	٪۹۰	۱۸۷	٪۱۰۰	٪۹۰
۳	۱۷	۵۵۸	٪۱۰۰	٪۱۰۰	۱۷۷	٪۱۰۰	٪۹۰
۴	۱۹	۶۳۷	٪۱۰۰	٪۱۰۰	۱۹۲	٪۱۰۰	۱۰۰
۵	۲۱	۷۱۴	٪۱۰۰	٪۱۰۰	۱۷۳	٪۱۰۰	۱۰۰
۶	۲۳	۷۷۲	٪۱۰۰	٪۹۰	۲۶۱	٪۱۰۰	٪۹۰
متوسط	۱۸/۳	۶۱۹	٪۱۰۰	٪۹۶/۷	۱۹۹/۳	٪۱۰۰	٪۹۵



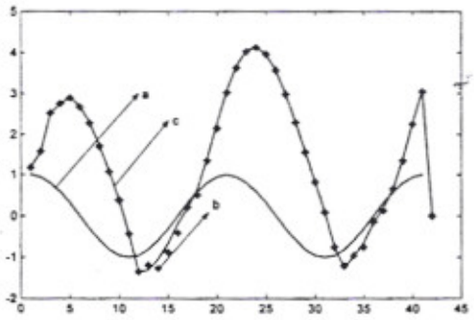
شکل۱۵: منحنی تغییرات ضریب پهنای روشنی



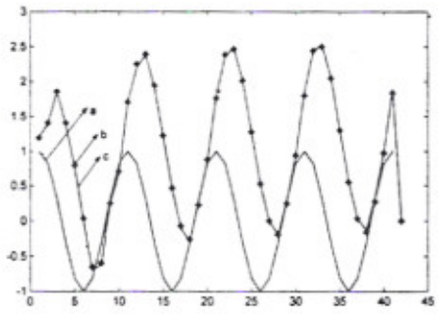
شکل۱۶: منحنی تغییرات ضریب پهنای روشنی

جدول ۳: مقایسهٔ الگوریتم غیر تطبیقی با الگوریتم تطبیقی برای مسئلهٔ تقریب تابع غیر خطی زمان گسستهٔ درجهٔ دوّم

شبکه	واحد مخفی	وزنهای لایهٔ مخفی (الگوریتم غیر تطبیقی)	خطای شبکهٔ عصبی به ازای ورودی سینوسی	وزنهای لایهٔ مخفی (الگوریتم تطبیقی)	خطای شبکهٔ عصبی به ازای ورودی سینوسی
۱	۸	۲۳	۰/۳۸۲۲	۱۹	۱/۱۱۶۲
۲	۱۰	۲۵	۰/۹۷۱۷	۲۲	۰/۹۴۰۹
۳	۱۳	۳۲	۲/۷۰۰۵	۲۸	۲/۲۰۸۲
۴	۱۷	۴۱	۰/۶۵۲۶	۳۸	۰/۷۳۲۵
۵	۲۰	۴۷	۰/۶۶۳۱	۴۳	۰/۸۰۱۲
۶	۲۵	۶۳	۱/۳۰۴۷	۵۷	۱/۴۵۸۰
متوسط	۱۵/۵	۳۸/۵	۱/۱۱۲۵	۳۴/۵	۱/۲۰۹۵



شکل ۱۷: a) ورودی b) خروجی تابع c) خروجی شبکه



شکل ۱۸: a) ورودی b) خروجی تابع c) خروجی شبکه

جدول ۴: مقایسه الگوریتم غیر تطبیقی با الگوریتم تطبیقی برای مسئلهٔ تشخیص اعداد چاپی فارسی

شبکه	واحد مخفی	وزنهای لایهٔ مخفی (الگوریتم غیر تطبیقی)	در صد تشخیص (آموزش)	در صد تشخیص (آزمایشی)	وزنهای لایهٔ مخفی (الگوریتم تطبیقی)	در صد تشخیص (آموزش)	در صد تشخیص (آزمایشی)
۱	۶	۲۶	٪۱۰۰	٪۱۰۰	۱۱	٪۱۰۰	٪۱۰۰
۲	۷	۲۳	٪۱۰۰	٪۱۰۰	۱۹	٪۱۰۰	٪۹۰
۳	۱۲	۴۷	٪۱۰۰	٪۹۰	۲۹	٪۱۰۰	٪۱۰۰
۴	۱۲	۳۲	٪۱۰۰	٪۱۰۰	۲۹	٪۱۰۰	٪۱۰۰
متوسط	۹/۲۵	۳۲	٪۱۰۰	٪۹۷/۵	۲۲	٪۱۰۰	٪۹۷/۵

۷- نتیجه گیری

در این مقاله یک الگوریتم جدید بر اساس اتوماتانهای یادگیر برای تطبیق بازهٔ تصمیم گیری مورد استفاده در الگوریتم تعیین تعداد وزن، ارائه گردیده است. الگوریتم پیشنهادی از اتوماتان یادگیر و الگوریتم پس انتشار خطا برای تعیین ساختار شبکه استفاده میکند. درالگوریتم پیشنهادی برای پاسخ دادن به اتوماتان مربوط به تنظیم بازه های تصمیم گیری از معیار خطا کمک گرفته ایم.الگوریتم جدید شبکه هایی با اتصالات محلی که دارای پیچیدگی پائین و قدرت تعمیم بالایی هستند تولید میکنند. این الگوریتم بر روی مسائل مختلفی پیاده سازی شده است. نتایج مشابه سازیها میدهد شبکه های تولید شده توسط الگوریتم جدید از پیچیدگی پائین تری نسبت به الگوریتم تعیین وزن غیر تطبیقی برخوردار میباشند.

- تشکر و قدردانی

در اینجا بر خود لازم میدانم از آقای مهندس حمید بیگی دانشجوی دکتری دانشکده کامپیوتر دانشگاه صنعتی امیرکبیر بخاطر داده های اعداد فارسی که در اختیار اینجانب قرار داد کمال تشکر و قدردانی را بعمل بیاورم.

مراجع

[1] Tin-Yau Kwok and Dit-Yan Yeung, "Constructive Algorithms for Structure Learning in Feedforward Neural Networks for Regression Problems,"IEEE Trans. Neural Networks, vol. 8, NO. 3, pp. 630-645, MAY 1997.

[2] Meybodi, M. R. and Beigy. H. (1999), "Optimization of Neural Networks Using Learning Automata," Proc. Of 4th Annual Int. Computer Society of Iran Computer Conf. CSICC-98, Tehran, Iran, pp. 417-428, Iran (In Persian).

[3] M. C. Mozer and P. Smolensky, "Skeletonization: A technique for trimming the fat from a network via relevance assessment," in Advances in Neural Information Processing (1), D.S. Touretzhy, Ed. (Denver 1988), 1989, pp. 107-115.

[4] B. E. Segee and M. J. Carter, "Fault tolerance of pruned multilayer networks," in Proc. Int. Joint Conf. Neural Networks, vol. II (Seattle), pp. 447-452, 1991.

[5] E. D. Karnin, "A simple procedure for pruning back-propagation trained neural networks ," IEEE Trans. Neural Networks, vol. 1, no. 2, pp. 239-242, 1990.

[6] Y. Le Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in Advances in Neural Information Processing (2), D.S. Touretzky, Ed. (Denver 1989), 1990, pp. 598-605.

[7] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman, " Generalization by weight elimination with application to forecasting," in Advances in Neural Information Processing (3), R. Lippmann, J. Moody, and D. Touretzky, Eds., 1991, pp. 875-882.

[8] A. S. Weigend, D. E. Rumelhart, and B. A. Huberman, "Backpropagation, weight-elimination and time series prediction," in Proc. 990 Connectionist Models Summer School, D. Touretzky, J. Elman, T. Sejnowski, and G. Hinton, Eds., 1990, pp. 105-116.

[9] Y. Chauvin, "A backpropagation algorithm with optimal use of hidden units," in Advances in Neural Information (1), D. S. Touretzky, Ed. (Denver 1988), 1989, pp. 519-526.

[10] S. J. Hanson and L. Y. Pratt, "Comparing biases for minimal network construction with backpropagation," in Advances in Neural Information Processing (1), D.S. Touretzky, Ed. (Denver 1988), 1989, pp. 177-185.

[11] Y. Chauvin, " Dynamic behavior of constrained backpropagation networks," in Advances in Neural Information Processing(2), D. S. Touretzky, Ed. (Denver 1989), 1990, pp. 642-649.

[12] C. Ji, R. R. Snapp, and D. Psaltis, "Generalizing smothness constrains from discrete samples," Neural Computation, vol. 2, no. 2, pp. 188-197, 1990.

[13] D. C. Plaut, S. J. Nowlan, and G. E. Hinton, "Experiments on learning by backpropagation," Tech. Rep. CMU-CS-86-126, Carnegie-Mellon Univ., 1986.

[14] S . J. Nowlan and G. E. Hinton, " Simplifying neural networks by soft weight sharing," Neural Computation, vol. 4, pp. 473-493, 1992.

[15] Marco Muselli, "On Sequential Construction of Binary Neural Networks," IEEE Trans. Neural Networks,

vol. 6, no. 3, pp. 678-690, MAY. 1995.
[16] Steven Young and Tom Downs, "CARVE-A Constructive Algorithms for Real-Valued Examples," IEEE Trans. Neural Networks Networks, vol. 9, no. 6, pp. 1180-1190, NOV. 1998.
[17] John O. Moody and Panos J. Antsaklis, " The Dependence Identification Neural Network Construction Algorithm," IEEE Trans. Neural Networks, vol. 7, no. 1, pp. 3-15, Jan. 1996.
[18] F. M. Frattale Mascioli and G. Martinelli, "A Constructive Algorithm for Binary Neural Networks: The Oil-Sport Algorithm,"IEEE Trans. Neural Networks, vol. 6, no. 3, pp. 794-797, MAY. 1995.
[19] Rudy Setiono and Lucas Chi Kwong Hui, " Use of a Quasi-Newton Method in a Feedforward Neural Network Construction Algorithm," IEEE Trans. Neural Networks, vol. 6, no. 1, pp. 273-277, JAN. 1995.
[20] Nabhan T. M. and Zomaya A. Y., " Toward Neural Networks Structure for Function Approximation," Neural Networks, vol. 7, no.1, pp. 89-99, 1993.
[21] Hirose Y., Yamashita K. and Hijya S., "Backpropagation Algorithm which Varies The Number of Hidden Units," Neural Networks, vol. 4, no. 1, pp. 61-66, 1991.
[22] J. D. Schaffer, D. Whitely and L. J. Eshelman, "Combination of genetic algorithms and neural networks: A Survey of the state of the art," IEEE Proc. COGANN-92, pp. 1-37, 1992.
[23] V. Maniezzo, " Genetic Evolution of the Topology and Weight Distribution of Neural Networks," IEEE Trans. on Neural Networks, vol. 5, no. 1, pp. 39-53, 1994.
[24] P. J. Angeline, G. M. Saunders and J. B. Pollack, "Evolutionary Algorithms that Construct Recurrent Neural Networks," IEEE Trans. on Neural Networks, vol. 5, no. 1, pp. 54-65, 1994.
[25] X. Yao and Y. Liu, "A New Evolutionary System for Evolving Artificial Neural Networks," IEEE Trans. On Neural Networks, vol. 8, no. 3, pp. 694-713, 1997.
[26] D. Whitley and C. Bogart, " The Evolution of Connectivity: Pruning Neural Networks Using Genetic Lagorithms," Proc. Of Int. Joint Conf.on Neural Networks, vol. 1, pp. 134, 1990.
[27] Beigy. H. and Meybodi, M. R., " Optimization of Topology of Neural Networks Using Learning Automata," Proc. Of 4th Annual Int. Computer Society of Iran Computer Conf. CSICC-98, Tehran, Iran, pp. 417-428 (In Persian) 1999.
[28] M. B. Menhaj, " Computational Intelligence (vol. 1) Fundamentals of Neural Networks ", Professor Hessabi Publication, 1998.
[29] Y. Chauvin, "Generalization performance of overtrained backpropagation networks," in Neural Networks, Proc. EUROSIP Workshop, L. B. Almedia and C. J. Wellekens, Eds., Feb. 1990, pp. 46-55.
[30] E. Levin, N. Tishby, and S. A. Solla, " A statistical appaoach to learning and generalization in layered neural networks," Proc IEEE, vol. 78, no. 10, pp.1568-1574, Oct. 1990.
[31] D. B. Schwartz, V. K. Samalan, S. A. Solla, and J. S. Denker, " Exhaustive learning, " Neural Computation, vol. 2, no. pp. 374 385, 1990.
[32] N. Tishby, E. Levin, and S. A. Solla, " Consistent in ference of probabilities in layered networks: Predictions and generalization," in Proc. Int. Joint Conf. neural Networks, 1989, p. 403.
[33] A. Blumer, A. Ehrenfeucht, D. Haussler, and M. Warmuth,"Learnability and the Vapnik-Chervonenkis Dimention," J. Ass. Comput. Mach., vol. 36, no. 4, pp. 929-965, 1989.
[34] A. Ehrenfeucht, D. Haussler, M. Kearns, and L. Valiant, " A genral lower bound on the number of examples needed for learning." In Proc. 1988 Workshop Computational Learning theory, 1988.
[35] L. G. Valiant, " A theory of the learnable, " Commun.1 Ass. Comut. Mach., vol. 27, no. 11, pp. 1134-1142,1984.
[36] K. S. Narendra and M. A. L. Thathachar, Learning Automata: An Introduction, Prentice Hall, Englewood cliffs, 1989.
[37] M. R. Meybodi and S. Lakshmivarahan," Optimality of a General Glass of Learning Algorithm ", Information science, Vol. 28, pp. 1-20, 1982.
[38] M. R. Meybodi and S. Lakshmivarahan," on a class of Learning Algorithm which have a symmetric Behavior under Success and Failure", Springer Verleg Lecture Notes in Statistics, pp. 145 -155, 1984.
[39] M. R. Meybodi, " Results on a Strongly Absoulutely Expedient Learning Automata ", Proc. Of OU Inference Conf. 86, ed. D. R. Mootes and R. Butrick, Athens, Ohio: Ohio University Prees, pp. 197-204.
[40] Beigy. H. and Meybodi, M. R. (1999), " Optimization of Neural Networks Using Learning Automata," Proc. Of 4th Annual Int. Computer Society of Iran Computer Conf. CSICC-98, Tehran, Iran, pp. 417-428, Iran (In Persian).
[41] R. Reed, " Pruning algorithms-Asurvey, " IEEE Trans. Neural Networks, vol. 4, no. 5, pp. 740-747, Sept. 1993.
[42] Meybodi, M. R. and Beigy. H. (2000), " Neural Networks Engineering Using Learning Automata: Determining Optimal size for three layer Neural Networks" Proc. Of 5th Annual Int. Computer Society of Iran Computer Conf. CSICC-99, Tehran, Iran, pp. 431-450, Iran (In Persian).

P. W. I. T

The Ninth
Iranian Conference on Electrical Engineering

Proceedings

CONTROL

May 8-10, 2001
Power & Water Institute of Technology ,
Tehran ,Iran

I.C.E.E