الگوریتم بقا تعمیم یافته: الگوریتمی جهت تعیین ساختار شبکه‌های عصبی چندلایه

مجید انجیدنی **محمد رضا میبدی**

anjidani@ce.aut.ac.ir **meybodi@ce.aut.ac.ir**

آزمایشگاه محاسبات نرم
دانشکده مهندسی کامپیوتر و فناوری اطلاعات
دانشگاه صنعتی امیرکبیر
تهران ایران

چکیده

جهت تعیین ساختار شبکه‌های عصبی و یافتن یک ساختار مناسب (نزدیک بهینه) برای شبکه الگوریتمهای متعددی ارائه شده است. نمونه‌هایی از این الگوریتمها، الگوریتمهای هرس، سازنده، ترکیبی، تکاملی و الگوریتمهای مبتنی بر اتوماتاهای یادگیر هستند که با هدف ایجاد شبکه‌های کوچک ارائه شده اند. تنها الگوریتمهایی که از اتوماتاهای یادگیر جهت تعیین ساختار شبکه استفاده می‌کند، الگوریتم بقاء[۱] و نسخه اصلاح شده آن هستند که هر دو برای شبکه‌های عصبی سه لایه ارائه شده‌اند. در این مقاله یک نسخه تعمیم‌یافته از الگوریتم بقاء نورون[۲] اصلاح شده برای شبکه‌های عصبی با تعداد لایه دلخواه ارائه می‌گردد. الگوریتم بقاء تعمیم‌یافته با دو مورد از الگوریتمهای هرس با نامهای S&D[۳] و تکراری[۴] مقایسه می‌گردد. الگوریتمها بر روی مسائل ارقام انگلیسی، Encoding، ارقام دست نویس فارسی، XOR سه بیتی و حروف چینی آزمایش گردید. نتایج آزمایشها برتری الگوریتم بقاء تعمیم‌یافته را در شبکه‌های با دو و سه لایه میانی در مقایسه با دو الگوریتم هرس ذکر شده نشان می‌دهد. همچنین اتوماتای کرایلو در میان اتوماتاهای ساختار ثابت به عنوان بهترین گزینه برای تعیین ساختار شبکه‌های شامل دو و سه لایه میانی معرفی می‌شود.

واژه‌های کلیدی: شبکه‌های عصبی چند لایه، انتشار خطا به عقب، الگوریتم تنظیم ساختار شبکه، اتوماتای یادگیر

۱) مقدمه

تعداد لایه های مخفی، تعداد نورونها در هر لایه مخفی و وزنهای آن در شبکه های عصبی چند لایه تاثیر بالایی برروی کارایی آنها دارد. یک شبکه با اندازه کوچک، خروجی دقیق تولید نمی‌کند و شبکه‌ای با اندازه بزرگ، بسیار کند و پر هزینه خواهد بود و برای تعمیم مجموعه آموزشی، نیاز به مجموعه آموزشی بزرگی خواهد داشت. طراحی یک شبکه یا ساختار بهینه یک مسئله NP-Hard است[۱]. بهمین جهت بیشتر الگوریتمهای ارائه شده برای تعیین ساختار شبکه‌های عصبی، الگوریتمهای تقریبی هستند. این الگوریتمها قبل، در حین یا بعد از یادگیری، ساختار مناسبی برای شبکه تعیین می‌نمایند. بعضی از این الگوریتمها از اطلاعات محلی و بعضی دیگر از اطلاعات عمومی برای یافتن ساختار مناسب شبکه استفاده می‌کنند. این الگوریتمها را می‌توان به پنج گروه عمده زیر تقسیم کرد:

الگوریتمهای هرس[۵]: روند کار در این الگوریتمها به این نحو است که ابتدا یک شبکه بزرگ را در نظر گرفته و بتدریج در حین آموزش یا بعد از آن نورونها و وزنهای اضافی را از شبکه هرس می‌کند. در الگوریتم‌های هرس، تعداد نورونهای مخفی بایستی در ابتدای آموزش مشخص شود که معمولا بزرگترین تعدادی که ممکن است مورد نیاز باشد یا حداکثر تعدادی که امکان دارد در نظر گرفته می‌شود[۲][۳][۴][۵].

---

[۱] Survival
[۲] Neuron Survival Algorithm (NSA)
[۳] Sietsma and Dow
[۴] Iterative
[۵] Pruning Algorithms

الگوریتمهای سازنده[۵]: این الگوریتمها با یک شبکه کوچک شروع به آموزش کرده و بتدریج در حین آموزش شبکه ، نورون یا لایه مخفی به شبکه می‌افزایند. این الگوریتمها معمولا شبکه‌های کوچک تولید می‌کنند که دارای پیچیدگی آموزش بالا هستند[۶][۷].

الگوریتمهای ترکیبی[۷]: الگوریتمهای ترکیبی از الگوریتمهای سازنده و الگوریتمهای هرس برای تعیین ساختار شبکه استفاده می‌کنند. در این الگوریتمها برای رسیدن به شبکه مطلوب ممکن است وزن ، نورون یا لایه مخفی کم و یا زیاد شود[۸][۹].

الگوریتمهای تکاملی[۸]: در این الگوریتمها تعیین ساختار بهینه برای شبکه از طریق جستجو در فضای ساختارها انجام می‌گیرد. هر نقطه از این فضا نماینده یک ساختار شبکه است. الگوریتم جستجو با استفاده از یک معیار کارایی مانند حداقل خطا و یا پیچیدگی آموزش به دنبال مناسبترین ساختار می‌باشد[۱۰][۱۱].

الگوریتم‌های بر اساس اتوماتاهای یادگیر: تنها الگوریتم گزارش شده بر اساس اتوماتاهای یادگیر الگوریتم بقاء نام دارد که توسط بیگی و میبدی ارائه گردیده است[۱۲][۱۳][۱۴]. دو نسخه متفاوت از الگوریتم بقاء موجود است که یکی برای تعیین حداقل تعداد نورونها (الگوریتم بقاء نورون) و دیگری برای تعیین حداقل تعداد وزنها (الگوریتم بقاء وزن) به کار برده می‌شود. الگوریتم بقاء از یک اتوماتای یادگیر مهاجرت اشیاء به عنوان ابزار جستجوی عمومی و الگوریتم یادگیری انتشار خطا به عقب استفاده می‌کند و در حین آموزش، یک ساختار مناسب برای شبکه عصبی سه لایه(ساختاری که دارای اندازه کوچک، پیچیدگی آموزش کم و قدرت تعمیم بالا باشد) تعیین می‌نماید. در الگوریتم بقاء آموزش از یک شبکه عصبی سه لایه بزرگ شروع شده و اتوماتای یادگیر با افزودن و کاستن نورونهای مخفی، تعداد نورونهای لایه مخفی و یا وزنهای این شبکه را تعیین می‌کند. به دلیل استفاده از روشهای جستجوی عمومی(اتوماتاهای یادگیر)، امکان گرفتاری در مینیمم‌های محلی کاهش می‌یابد. قبلا اتوماتاهای یادگیر برای برای تطبیق پارامترهای شبکه‌های عصبی مورد استفاده قرار گرفته است[۱۶][۱۷][۱۸][۱۹][۲۰] .

الگوریتم بقاء اصلاح شده نیز برای شبکه‌های عصبی سه لایه توسط انجیدنی و میبدی ارائه شده است[21]. در این مقاله یک نسخه تعمیم‌یافته از الگوریتم بقاء اصلاح شده برای شبکه‌های عصبی با تعداد لایه میانی دلخواه (ساختار لایه‌ای دلخواه) ارائه می‌گردد. نسخه تعمیم‌یافته الگوریتم بقاء علاوه بر امکان استفاده از اتوماتاهای دلخواه (در این مقاله از اتوماتای یادگیر مهاجرت اشیاء و اتوماتای یادگیر کرایلو استفاده می‌شود) و کاهش فعالیت نورون به روش اصلاح شده، قادر به تعیین ساختار شبکه‌های با ساختار لایه‌ای دلخواه می‌باشد. همچنین در نسخه تعمیم یافته، استراتژیهایی نیز جهت فعال سازی اتوماتاها در لایه‌ها مطرح می‌شود که بسته به شرایط، ممکن است یکی از آنها منلسب باشد. الگوریتم بقاء تعمیم‌یافته برای شبکه‌های عصبی شامل دو و سه لایه میانی با دو مورد از الگوریتمهای هرس(S&D و تکراری) مقایسه می‌گردد. الگوریتمها بر روی مسائل ارقام انگلیسی، Encoding، ارقام دست نویس فارسی، XOR سه بیتی و حروف چینی آزمایش گردیده‌اند. نتایج آزمایشها برتری الگوریتم بقاء تعمیم‌یافته در شبکه‌های با دو و سه لایه میانی را در مقایسه با دو الگوریتم هرس ذکر شده نشان می‌دهد. همچنین اتوماتای کرایلو از میان اتوماتاهای ساختار ثابت به عنوان بهترین گزینه برای تعیین ساختار شبکه‌های با دو و سه لایه میانی معرفی می‌شود.

ادامه مقاله بصورت زیر سازماندهی شده است. در بخش ۲ اتوماتای یادگیر و انواع آن معرفی می‌گردد. سپس در بخش ۳ الگوریتم بقاء با ذکر اصلاحات انجام گرفته در آن شرح داده می‌شود. نتایج آزمایشها و نتیجه گیری نیز در بخشهای بعدی ارائه خواهند شد.

۲- اتوماتاهای یادگیر و الگوریتم انتشار خطا به عقب

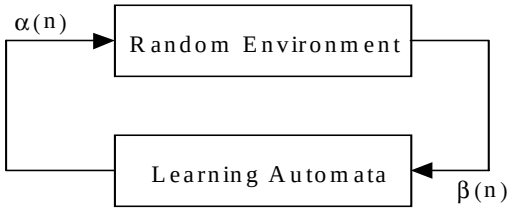
۲-۱- اتوماتاهای یادگیر[۹]

اتوماتای یادگیر یک ماشین با حالات محدود[۱۰] است که میتواند تعدادی محدود عمل را انجام دهد. هر عمل انتخاب شده توسط محیطی احتمالی ارزیابی می‌گردد و پاسخی به اتوماتای یادگیر داده می‌شود. اتوماتای یادگیر از این پاسخ استفاده مینماید وعمل خود را برای مرحله بعد انتخاب می کند.. در طی این فرایند، اتوماتای یادگیر یاد می‌گیرد که چگونه بهترین عمل را انتخاب نماید. شکل ۱ ارتباط بین اتوماتای یادگیر و محیط را نشان می دهد. محیط[۱۱] را می توان توسط سه تایی $E \equiv \{\alpha, \beta, c\}$ نشان داد که در آن $\alpha = \{\alpha_1, ..., \alpha_r\}$

---

[۵] Constructive Algorithms
[۶] Hybrid Algorithms
[۷] Evolutionary Algorithms
[۹] Learning Automata
[10] Finite State Machine
[11] Environment

مجموعه ورودیها، $\beta=\{\beta_1,...,\beta_m\}$ مجموعه خروجیها و $c=\{c_1,...,c_r\}$ مجموعه احتمالهای جریمه می باشد. هرگاه β مجموعه دو عضوی باشد محیط از نوع P می باشد. در چنین محیطی $\beta_1=1$ به عنوان جریمه و $\beta_2=0$ به عنوان پاداش در نظر گرفته می شود. در محیط Q، $\beta(n)$ می تواند به طور گسسته یک مقدار از مقادیر محدود در فاصله $[0,1]$ و در محیط از نوع S، $\beta(n)$ هر مقدار در فاصله $[0,1]$ را اختیار کند. c_i احتمال اینکه عمل α_i نتیجه نا مطلوب داشته باشد، می باشد. در محیط ایستا مقادیر c_i بدون تغییر می مانند، حال آنکه در محیط غیر ایستا این مقادیر در طی زمان تغییر می کنند.

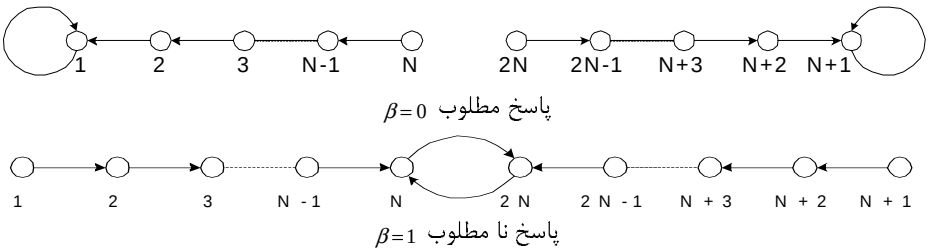


شکل ۱: ارتباط بین اتوماتای یادگیر و محیط

اتوماتاهای یادگیر به دو گروه با ساختار ثابت و با ساختار متغیر تقسیم بندی میگردند. در ادامه به شرح مختصری درباره اتوماتاهای یادگیر با ساختار متغییر[12] و اتوماتاهای یادگیر با ساختار ثابت[13] می پردازیم.

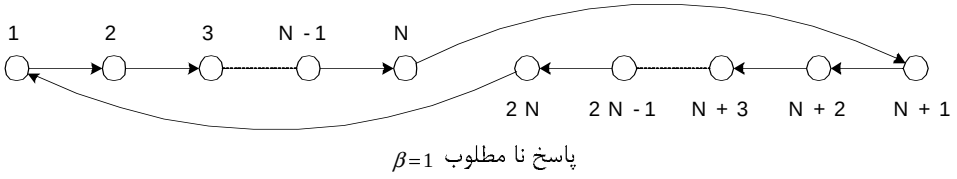
اتوماتای یادگیر با ساختار ثابت: اتوماتای یادگیر با ساختار ثابت توسط ۵ تایی $\{\alpha,\beta,F,G,\phi\}$ نشان داده میشود که $\alpha\equiv\{\alpha_1,\alpha_2,...,\alpha_r\}$ مجموعه اعمال اتوماتای یادگیر، $\beta\equiv\{\beta_1,\beta_2,...,\beta_m\}$ مجموعه ورودیهای اتوماتای یادگیر، $F:\phi\times\beta\to\phi$ تابعی که بر اساس پاسخ محیط، وضعیت جدید را می‌یابد، $G:\phi\to\alpha$ تابع خروجی که وضعیت کنونی را به خروجی بعدی می‌نگارد و $\phi(n)\equiv(\phi_1,\phi_2,...,\phi_k)$ مجموعه وضعیت‌های داخلی اتوماتای یادگیر میباشد چند نمونه از اتوماتاهای یادگیر با ساختار ثابت که در این مقاله از آنها استفاده کرده‌ایم در زیر معرفی میگردد

- **اتوماتای یادگیر** $L_{2N,2}$: این اتوماتا تعداد پاداش‌ها و جریمه‌های دریافت شده برای هر عمل را نگهداری کرده و تنها زمانی که تعداد جریمه‌ها بیشتر از پاداش‌ها می‌گردد، عمل دیگر را انتخاب می‌کند. نمودار تغییر وضعیت این اتوماتای یادگیر مطابق شکل ۲ می‌باشد.



شکل ۲: نمودار تغییر وضعیت اتوماتای یادگیر $L_{2N,2}$

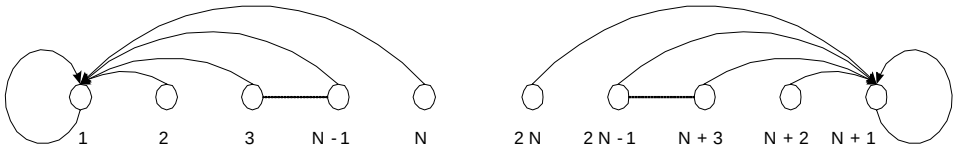
- **اتوماتای یادگیر** $G_{2N,2}$: در این اتوماتای یادگیر بر خلاف $L_{2N,2}$، عمل α_2 حداقل N بار انجام می‌گردد (پس از گرفتن N جریمه) تا اینکه در نهایت عمل α_1 دوباره انتخاب شود. گراف تغییر وضعیت این اتوماتای یادگیر برای پاسخ مطلوب مانند اتوماتای یادگیر $L_{2N,2}$ بوده و برای پاسخ نامطلوب مطابق شکل ۳ می باشد.



[12] Variable Structure Learning Automata
[13] Fixed Structure Learning Automata

شکل ۳: نمودار تغییر وضعیت اتوماتای یادگیر $G_{2N,2}$

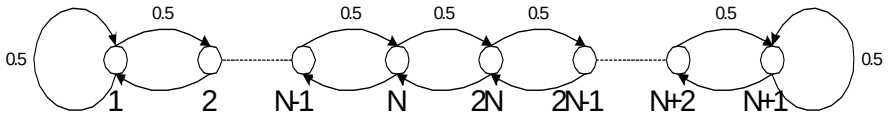
- **اتوماتای یادگیر *Krinsky***: این اتوماتای یادگیر زمانی که پاسخ محیط نامطلوب است، مانند اتوماتای یادگیر $L_{2N,2}$ رفتار می‌کند. اما برای پاسخ مطلوب هر وضعیت $\phi_i(i=1,2,3,...,N)$ به وضعیت ϕ_1 و هر وضعیت $\phi_i(i=N+1,N+2,...,2N)$ به وضعیت ϕ_{N+1} می رود. بنابراین همیشه N پاسخ نامطلوب متوالی لازم است تا اتوماتای یادگیر عمل خود را تغییر دهد. نمودار تغییر وضعیت این اتوماتای یادگیر برای پاسخ نامطلوب مانند اتوماتای یادگیر $L_{2N,2}$ بوده و برای پاسخ مطلوب مطابق شکل ۴ می باشد.



پاسخ مطلوب $\beta=0$

شکل ۴: نمودار تغییر وضعیت اتوماتای یادگیر *Krinsky*

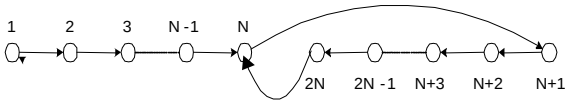
- **اتوماتای یادگیر *Krylov***: در این اتوماتای یادگیر زمانیکه پاسخ محیط مطلوب است، تغییر وضعیت مانند اتوماتای یادگیر $L_{2N,2}$ میباشد. اما زمانیکه پاسخ محیط نامطلوب می‌باشد، هر وضعیت $\phi_i(i\neq 1,N,N+1,2N)$ با احتمال 0.5 به وضعیت ϕ_{i+1} و با احتمال 0.5 به وضعیت ϕ_{i-1} مطابق شکل ۵ منتقل می شود.



پاسخ نا مطلوب $\beta=1$

شکل ۵: نمودار تغییر وضعیت اتوماتای یادگیر *Krylov*

- **اتوماتای یادگیر مهاجرت اشیاء**[۱۴]: نمودار تغییر وضعیت در این اتوماتای یادگیر برای پاسخ مطلوب مانند اتوماتای $L_{2N,2}$ بوده و برای پاسخ نامطلوب مطابق شکل ۳ می‌باشد.



پاسخ نامطلوب $\beta=1$

شکل ۶: نمودار تغییر وضعیت اتوماتای یادگیر مهاجرت اشیاء

۲-۲- اتوماتای یادگیر با ساختار متغیر

اتوماتای یادگیر با ساختار متغیر توسط ۴ تائی $\{\alpha,\beta,p,T\}$ نشان داده می شود که در آن $\alpha\equiv\{\alpha_1,\alpha_2,\cdots,\alpha_r\}$ مجموعه عملهای اتوماتای یادگیر، $\beta\equiv\{\beta_1,\beta_2,\cdots,\beta_m\}$ مجموعه ورودیهای اتوماتای یادگیر، $p\equiv\{p_1,p_2,\cdots,p_r\}$ بردار احتمال انتخاب هر یک از عملها، و $p(n+1)=T[\alpha(n),\beta(n),p(n)]$ الگوریتم یادگیری می باشد. در این نوع از اتوماتاهای یادگیر، اگر عمل α_i در مرحله n ام انتخاب شود و پاسخ مطلوب از محیط دریافت نماید، احتمال $p_i(n)$ افزایش یافته و سایر احتمالها کاهش می یابند، و برای پاسخ نامطلوب احتمال $p_i(n)$ کاهش یافته و سایر احتمالها افزایش می یابند. در هر حال، تغییرات به گونه ای صورت می گیرد تا حاصل جمع $p_i(n)$ ها همواره ثابت و مساوی یک باقی بماند. الگوریتم زیر یک نمونه از الگوریتمهای یادگیری خطی برای اتوماتای یادگیر با ساختار متغیر است.

---

[۱۴] Object Migrating Automata

الف- پاسخ مطلوب

$$p_i(n+1) = p_i(n) + a[1 - p_i(n)]$$
$$p_j(n+1) = (1-a)\,p_j(n) \qquad j \neq i \quad \forall j$$

ب- پاسخ نامطلوب

$$p_i(n+1) = (1-b)\,p_i(n)$$
$$p_j(n+1) = \frac{b}{r-1} + (1-b)\,p_j(n) \qquad j \neq i \quad \forall j$$

در روابط فوق، پارامتر پاداش و a پارامتر پاداش و b پارامتر جریمه می باشد. با توجه به مقادیر a و b سه حالت را می توان در نظر گرفت. زمانیکه a و b با هم برابر باشند، الگوریتم را، L_{RP} [۱۵] می نامیم. زمانیکه b از a خیلی کوچکتر باشد، الگوریتم را $L_{R\varepsilon P}$ [۱۶] می نامیم. زمانیکه b مساوی صفر باشد، الگوریتم را L_{RI} [۱۷] می نامیم. برای اطلاعات بیشتر در باره اتوماتاهای یادگیر با ساختار متغیر و اتوماتاهای یادگیر با ساختار ثابت مانند $L_{2N,2}$، $G_{2N,2}$، کرینسکی[۱۸] و کرایلو[۱۹] که در این مقاله از آنها استفاده شده است می توان به مراجع [۲۳] و[۱۵] مراجعه نمود.

۲-۳- الگوریتم انتشار خطا به عقب[۲۰]

این الگوریتم یک الگوریتم بازگشتی کاهش گرادیان است که برای آموزش شبکه‌های عصبی پیشخور مورد استفاده قرار می‌گیرد. قانون کاهش گرادیان که این الگوریتم بر اساس آن کار می‌کند به صورت زیر است:

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

که W بردار وزن، n تعداد تکرار، η نرخ یادگیری، α ممنتم و G گرادیان تابع خطا می‌باشد که به صورت زیر محاسبه می‌شود

$$G(n) = -\nabla E_p(n)$$

E_p برابر مجموع مربعات خطا میباشد و به ورت زیر محاسبه میشود.

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

بطوریکه $T_{p,j}$ و $O_{p,j}$ به ترتیب خروجی *خواسته شده*[۲۱] و *واقعی*[۲۲] برای الگوی p در نورون خروجی j هستند. کارایی این الگوریتم برای یک کاربرد خاص به میزان زیادی به توپولوژی شبکه(تعداد لایه‌ها، تعداد نورونها در هر لایه و اتصالات میان لایه ای) وابسته است.

۳- الگوریتم بقاء نورون

در الگوریتم بقاء نورون از یک اتوماتای یادگیر مهاجرت اشیاء برای تعیین تعداد نورونهای لایه مخفی یک شبکه سه لایه استفاده شده است. وظیفه این اتوماتای یادگیر تقسیم بندی نورونهای لایه مخفی به دو گروه روشن و خاموش می‌باشد. این اتوماتای یادگیر به صورت شش تایی $\langle \alpha, H, \Phi, \beta, F, G \rangle$ نشان داده می‌شود که در آن $\underline{\alpha} = \{\alpha_1, \alpha_1\}$ اقدامهای اتوماتا یادگیر می‌باشد. اتوماتای یادگیر دارای دو اقدام است: *اقدام شماره یک*، اقدام مناسب یا واحدهای روشن نامیده می‌شود. نورونهایی که در وضعیتهای این اقدام واقع شوند برای آموزش شبکه مورد استفاده قرار می‌گیرند. *اقدام شماره دو*، اقدام نامناسب یا واحدهای خاموش نام دارد. $\underline{H} = \{H_1, H_2, ..., H_n\}$ نورونهای مخفی هستند که روشن و خاموش کردن آنها به عهده اتوماتای یادگیر می‌باشد. اگر نورون H_i در اقدام شماره یک ظاهر شود به معنای روشن بودن آن و در غیر این صورت خاموش خواهد بود. $\underline{\Phi} = \{\Phi_1, \Phi_2, ..., \Phi_{2N}\}$ حالتهای اتوماتای یادگیر بوده و N عمق

---

[15] Linear Reward Penalty
[16] Linear Reward Epsilon Penalty
[17] Linear Reward Inaction
[18] Krinsky
[19] Krylov
[20] Back-propagation Algorithm
[۲۱] Desired
[۲۲] Actual

حافظه می‌باشد. حالتهای اتوماتای یادگیر به دو گروه تقسیم می‌شوند: $\{\Phi_1,\Phi_2,...,\Phi_N\}$ و $\{\Phi_{N+1},\Phi_2,...,\Phi_{2N}\}$. بر این اساس نورونهای روشن با مجموعه $ON=\{H_i \mid 1\le State(H_i)\le N\}$ و نورونهای خاموش با مجموعه $OFF=\{H_i \mid N+1\le State(H_i)\le 2N\}$ نشان داده می‌شوند.

نورونهایی که در وضعیتهای مربوط به این اقدام واقع شوند برای آموزش مورد استفاده قرار نمی‌گیرند. $\underline{\beta}=\{0,1\}$ ورودیهای اتوماتای یادگیر می‌باشد. در این مجموعه ۱ جریمه و ۰ پاداش را نشان می‌دهد.

نحوه عملکرد الگوریتم به این صورت است که در ابتدا تمامی نورونها روشن بوده هستند و در آموزش شرکت می‌کنند. نورونهایی که دارای عملکرد مناسب نیستند جریمه شده و نورونهای با عملکرد مناسب پاداش داده می‌شوند. برای ارزیابی عملکرد یک نورون، از متوسط انرژی نورونها استفاده می‌کنیم(شرح در بخشهای بعد). برای نحوه تعبیر چگونگی عملکرد یک نورون، دو قانون موجود می‌باشد: اگر برای تمامی الگوهای ورودی، مقدار فعالیت نورون تغییرات زیادی داشته باشد در این صورت نورون دارای عملکرد خوبی است و اگر برای تمامی الگوهای ورودی مقدار فعالیت دارای تغییرات کمی باشد نورون دارای عملکرد خوبی نیست.

۳-۱) تشخیص نحوه عملکرد نورون روشن

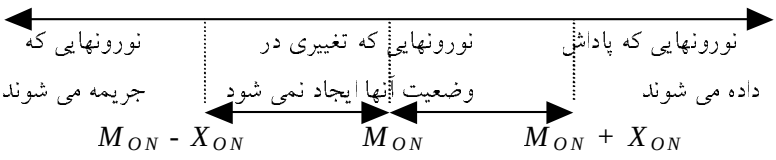
اگر فعالیت نورونی برای تمامی الگوها از یک مقدار آستانه کمتر باشد، نورون بد و اگر از یک مقدار آستانه بیشتر باشد، نورون خوب نامیده می‌شود. برای تعیین مقادیر آستانه، ابتدا واریانس مقدار فعالیت نورون برای تمامی الگوهای آموزش بصورت زیر محاسبه می‌شود:

$$\delta_l=\sqrt{\frac{\sum_{k=1}^{P}(|U_{lk}|-\mu_l)^2}{P}} \qquad l\in ON$$

که در آن، U_{lk} ، فعالیت نورون شماره l برای الگوی شماره K و P تعداد الگوهای آموزش می‌باشد. μ_l مقدار متوسط فعالیت نورون شماره l بوده که به صورت زیر تعریف می‌شود:

$$\mu_l=\frac{\sum_{k=1}^{P}|U_{lk}|}{P} \qquad l\in ON$$

پس از محاسبه واریانس نورونهای روشن، نورونهای روشنی که واریانس فعالیتهای آنها کمتر از یک مقدار آستانه باشد جریمه شده و نورونهایی که مقدار فعالیت آنها بزرگتر از یک مقدار آستانه دیگر باشد پاداش می‌بینند. نورونهای روشنی که واریانس فعالیت آنها بین دو مقدار آستانه قرار می‌گیرد جریمه یا پاداش داده نمی‌شوند(شکل۷).



شکل ۷: نحوه پاداش و جریمه نورونهای روشن

مقدار M_{ON} که مقدار متوسط واریانسهای نورونهای روشن می‌باشد بصورت زیر محاسبه می‌شود:

$$M_{ON}=\frac{\sum_{k\in ON}\delta_k}{|ON|}$$

پهنای X_{ON} بصورت زیر محاسبه می‌شود:

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

در معادله بالا ثابت λ_{ON} ضریب پهنای روشنی نامیده می‌شود. مقدار آستانه پایین $M_{ON}-X_{ON}$ و مقدار آستانه بالا $M_{ON}+X_{ON}$ می‌باشد.

۳-۲) نحوه تمایز بین نورونهای خاموش

نورونهای خاموش در آموزش شبکه شرکت نمی‌کنند. در الگوریتم بقاء نورون، مدت زمان خاموش بودن هر نورون برحسب تعداد epoch آموزشی، به عنوان پارامتری نگهداری می‌شود(n). فعالیت یک نورون خاموش برای یک الگو بر اساس آخرین مقدار فعالیت این نورون در زمان روشن بودن برای آن الگو محاسبه می‌شود. وقتی یک نورون برای مدت زیادی خاموش باشد، ارزش فعالیت نورون کاهش یافته و بتدریج باعث کم رنگ شدن نقش نورون خاموش می‌شود. به صورت روشن تر، فعالیت یک نورون خاموش را در زمانی خاص بصورت زیر محاسبه می‌کنیم:

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

در معادله بالا ثابت λ_d، ضریب کاهش فعالیت نامیده شده و n زمان را نشان می‌دهد. بنابراین مقدار فعالیت یک نورون خاموش به تدریج کاهش می‌یابد.

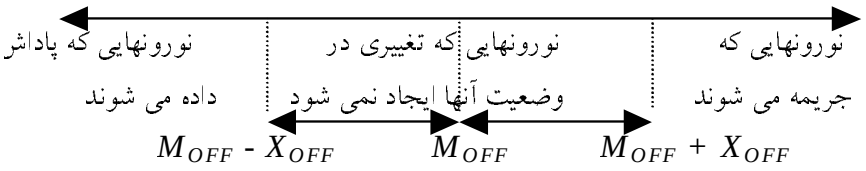
واریانس نورونهای خاموش بصورت زیر محاسبه می‌شود:

$$\delta_l = \sqrt{\frac{\sum_{k=1}^{P} (|U_{lk}| - \mu_l)^2}{P}} \qquad l \in OFF$$

که U_{lk} مقدار فعالیت نورون شماره l برای الگوی شماره K بوده و μ_l مقدار متوسط فعالیت نورون خاموش است که به صورت زیر بیان می‌شود:

$$\mu_l = \frac{\sum_{k=1}^{P} |U_{lk}|}{P} \qquad l \in OFF$$

پس از محاسبه واریانسِ نورونهای خاموش، نورونهایی که واریانس فعالیت آنها از یک مقدار آسنانه کمتر است پاداش دیده و نورونهایی که واریانس فعالیت آنها بین این دو مقدار آستانه می‌باشد نه جریمه و نه پاداش داده می‌شوند(کل ۸).



شکل۸: نحوه پاداش و جریمه نورونهای خاموش

مقدار M_{OFF} که مقدار متوسط واریانس نورونهای خاموش می‌باشد بصورت زیر محاسبه می‌شود:

$$M_{OFF} = \frac{\sum_{k \in OFF} \delta_k}{|OFF|}$$

پهنای X_{OFF} به صورت زیر محاسبه می‌شود:

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

در معادله بالا ثابت λ_{OFF} ضریب پهنای خاموشی نامیده می‌شود. مقدار آستانه پایین $M_{OFF} - X_{OFF}$ و مقدار آستانه بالا $M_{OFF} + X_{OFF}$ می‌باشد.

در [21] الگوریتم بقاء نورون اصلاح شده ارایه گردیده است. اصلاحات اعمال شده به الگوریتم بقا بشرح زیر است. :

۱) به جای اتوماتای یادگیر مهاجرت اشیاء که توسط الگوریتم بقا جهت تعیین تعداد نورونهای لایه مخفی در یک شبکه سه لایه استفاده می‌شد، در الگوریتم اصلاح شده از اتوماتای یادگیر L_{RP} (با پارامترهای a=b=0.1) که یک اتوماتای یادگیر با ساختار متغیر می‌باشد برای این منظور استفاده میشود.

۲) در الگوریتم بقاء، فعالیت یک نورون خاموش(خروجی نورون) در زمان n+1 بر حسب خروجی نورون در زمان n طبق رابطه زیر محاسبه می گردد:

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

در الگوریتم بقاء نورون اصلاح شده، بعد از هر epoch وزنهای ورودی به نورونهای خاموش و بایاس آنها را با ضریبی (λ) کاهش داده میشود که باعث تغییر در مقدار فعالیت نورون می گردد. اگر برای مدتی یک نورون خاموش بماند کلیه وزنهای نورون و بایاس آن به صفر نزدیک شده که با توجه به تابع فعالیت نورون ($f(x) = 1/(1+e^{-x})$)، باعث نزدیک شدن مقدار فعالیت نورون خاموش به مقدار ۰/۵ میشود. در این هنگام این چنین نورونی حذف گردیده و سپس به بایاس نورونهای لایه بعد مقدار $0.5 \times w_{ij}$ اضافه میشود. w_{ij} وزن مابین نورون خاموش(i) و نورون لایه بعد(j) می‌باشد.

این تغییر موجب کاهش محاسبات شده چرا که الگوریتم انتشار خطا و شبکه به حداقل تغییر نیازمند میباشد. بعد از هر epoch وزنها و بایاس واحدهای خاموش ا با ضریبی کاهش داده میشود و نورونهای خاموش در تصحیح وزنها و انتشار خطا شرکت داده نمیشود.

۳-۵) الگوریتم بقاء نورون تعمیم‌یافته

در این بخش روشی جهت تعمیم الگوریتم بقاء اصلاح شده برای شبکه‌هایی با بیش از چند لایه مخفی ارائه می‌گردد. تغییرات انجام گرفته بر روی الگوریتم بقا به قرار زیر میباشد.

۱) الگوریتم بقاء و الگوریتم بقاء اصلاح شده هر دو برای شبکه‌های سه لایه ارائه شده‌اند. در این الگوریتمها پارامترهای X_{ON}، M_{ON}، X_{OFF} و M_{OFF} برای تنها لایه میانی محاسبه شده و روشن یا خاموش شدن هر نورون میانی بر اساس مقایسه واریانس خروجی نورون(δ) با آستانه‌های $M_{ON} \pm X_{ON}$ و $M_{OFF} \pm X_{OFF}$ تعیین می‌گردد(طبق روابط بخشهای ۳-۱ و ۳-۲). در الگوریتم بقاء تعمیم‌یافته هر لایه مخفی شبکه، شامل پارامترهای X_{ON}، M_{ON}، X_{OFF} و M_{OFF} مربوط به خود بوده و برای هر لایه پارامترهای λ_{ON} , λ_{OFF} به طور مجزا تعیین می شوند. پارامتر δ برای هر نورون مخفی شبکه (واریانس خروجیهای نورون به ازای الگوهای ورودی) محاسبه شده و روشن یا خاموش بودن آن با توجه به پارامترهای لایه مخفی مربوط به آن نورون و طبق روابط بخشهای ۳-۱ و ۳-۲ محاسبه می‌شوند. به این ترتیب الگوریتم برای شبکه‌های با ساختار لایه‌ای دلخواه قابل استفاده می‌باشد.

۲) در شبکه با ساختار لایه‌ای دلخواه، چند روش برای فعال نمودن اتوماتاها در لایه‌های میانی می‌توان در نظر گرفت که به شرح زیرند:

الف) بعد از هر epoch، اتوماتاهای همه لایه‌ها همزمان فعال شوند..

ب) بعد از هر epoch، اتوماتای مربوط به یک لایه فعال شود(به ترتیب).

ج) بعد از هر epoch، اتوماتاهای لایه‌ها یک در میان(چند در میان) فعال شوند.

همانطور که مشاهده می‌شود برای فعال سازی اتوماتاها در لایه‌های میانی استراتژیهای مختلفی وجود دارد که بسته به شرایط ممکن است یکی از آنها مناسب باشد. جهت کسب اطلاعات بیشتر به [۲۲] مراجعه نمایید.

۴) نتایج شبیه سازیها

شبیه سازیها برای شبکه‌های با دو و سه لایه میانی انجام شده و برای فعال سازی اتوماتاها در لایه‌های میانی شبکه از روش (الف) در بخش ۳-۵ استفاده شده است. دو اتوماتای کرایلو و مهاجرت اشیاء در الگوریتم بقاء تعمیم‌یافته مورد آزمایش قرار گرفته‌اند. اتوماتای مهاجرت اشیاء که در الگوریتم بقاء سنتی استفاده می‌شد در شبکه‌های عصبی با دو و سه لایه میانی، نتایج پایین‌تری نسبت به اتوماتای کرایلو از خود نشان داده است.

دو الگوریتم مبتنی بر اتوماتای یادگیر و دو الگوریتم هرس تکراری و S&D در شبکه‌های با دو و سه لایه میانی و مسائل ارقام انگلیسی، Encoding، ارقام دست نویس فارسی، XOR سه بیتی و حروف چینی آزمایش شده‌اند. هر یک از این مسائل در ادامه شرح داده شده‌اند. در این مسائل، مولفه ۰ می‌بایست با مقدار ۱- جایگزین شود.

- **XOR**

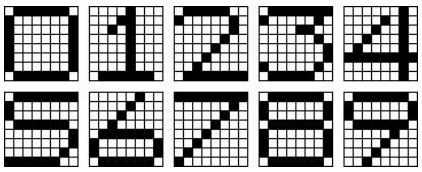
در مسئله XOR سه بیتی، الگوهای ورودی ۳ مولفه‌ای به شبکه ارائه می‌شود و در صورت فرد بودن تعداد مولفه‌های ۱ در الگو، خروجی شبکه فعال می شود.

- **ارقام انگلیسی**

ارقام انگلیسی به شکل ماتریسهای ۸×۸ در شکل ۹ نمایش داده شده‌اند. خانه‌های سیاه به معنای مولفه ۱ در ورودی و خانه‌های سفید به معنای مولفه ۱- هستند. این ارقام در قالب ورودیهای ۶۴ مولفه‌ای به شبکه ارائه می‌شوند. تابع تبدیل ورودی شبکه به خروجی می‌تواند به صورت یکی از توابع f_1 یا f_2 که در جدول ۱ ارائه شده باشد.

جدول ۱: توابع تبدیل در مسئله ارقام انگلیسی

x	$f_1(x)$	$f_2(x)$
۰	۰۰۰۰	۰۰۰۰۰۰۰۰۰۱
۱	۰۰۰۱	۰۰۰۰۰۰۰۰۱۰
۲	۰۰۱۰	۰۰۰۰۰۰۰۱۰۰
۳	۰۰۱۱	۰۰۰۰۰۰۱۰۰۰
۴	۰۱۰۰	۰۰۰۰۰۱۰۰۰۰
۵	۰۱۰۱	۰۰۰۰۱۰۰۰۰۰
۶	۰۱۱۰	۰۰۰۱۰۰۰۰۰۰
۷	۰۱۱۱	۰۰۱۰۰۰۰۰۰۰
۸	۱۰۰۰	۰۱۰۰۰۰۰۰۰۰
۹	۱۰۰۱	۱۰۰۰۰۰۰۰۰۰



شکل ۹: نحوه نمایش ارقام انگلیسی

- **Encoding**

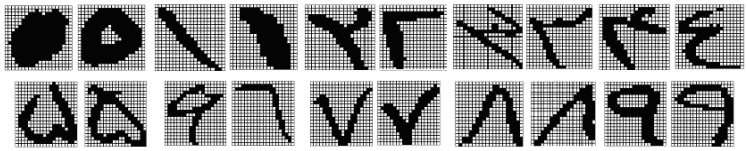
در این مسئله تنها یکی از مولفه‌های ورودی یک و بقیه صفر می‌باشد. خروجی، عدد باینری متناظر با مکان این مولفه را می‌دهد. تابع انکد کردن برای یک ورودی ۸ بیتی در جدول ۲ آمده است. در شبیه سازیها از انکدر ۸ بیت به ۳ بیت استفاده شده است.

جدول ۲: تابع انکدر ۸ بیتی

x	f(x)
۰۰۰۰۰۰۰۱	۰۰۰
۰۰۰۰۰۰۱۰	۰۰۱
۰۰۰۰۰۱۰۰	۰۱۰
۰۰۰۰۱۰۰۰	۰۱۱
۰۰۰۱۰۰۰۰	۱۰۰
۰۰۱۰۰۰۰۰	۱۰۱
۰۱۰۰۰۰۰۰	۱۱۰
۱۰۰۰۰۰۰۰	۱۱۱

- **ارقام دست نویس فارسی**

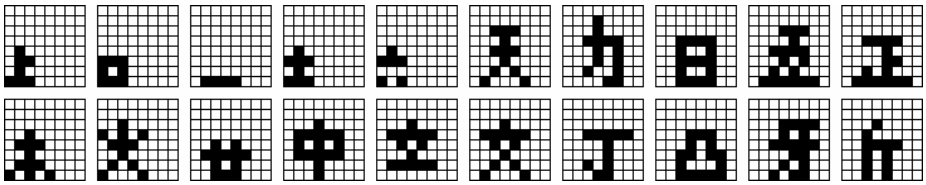
برای این مسئله، یک پایگاه داده شامل ۶۰۰ تصویر از ارقام دست نویس فارسی، مورد استفاده قرار گرفته است. این تصاویر پیش پردازش شده و به تصاویر سیاه و سفید با ابعاد ۲۰×۲۰ برای استفاده توسط شبکه تبدیل گردیده است. چند نمونه از تصاویر ۲۰×۲۰ شده این ارقام در شکل ۱۰ آمده است. در ابتدا تعدادی از تصاویر به عنوان نمونه‌های فاز آموزش در نظر گرفته می‌شوند و تصاویر باقیمانده برای فاز آزمایش استفاده می‌شوند. تابع تبدیل برای این مسئله مشابه تابع $f_2(x)$ در جدول ۱ است.



شکل ۱۰: چند نمونه از اعداد فارسی ۲۰×۲۰ شده.

- **حروف چینی**

۲۰ مورد از حروف چینی به شکل ماتریسهای ۸×۸ در شکل ۱۱ نمایش داده شده‌اند. این حروف به شبکه عصبی آموزش داده می‌شوند. خانه‌های سیاه به معنای مولفه ۱ در ورودی و خانه‌های سفید به معنای مولفه ۱- هستند. این حروف در قالب ورودیهای ۶۴ مولفه‌ای به شبکه ارائه می‌شوند. متناظر با هر حرف در خروجی شبکه مولفه‌ای در نظر گرفته می‌شود که در صورت ارائه آن حرف به شبکه، باید مولفه خروجی مربوط به آن فعال گردد.



شکل ۱۱: نحوه نمایش ۲۰ مورد از حروف چینی

جداول ۳ تا ۱۱ نتایج شبیه سازیها را نشان می‌دهند. نتایج موجود در این جداول مرتب بوده و مطلوبترین نتایج در سطرهای ابتدایی ذکر شده و به سمت پایین از اهمیت نتایج کاسته می شود. نقاطی که در جداول هاشور خورده، نرخ تشخیص پایین شبکه را به ازای استفاده از یک الگوریتم نشان می‌دهند.

در تمام مسائل به جز مسئله ارقام دست نویس فارسی و مسئله حروف چینی، آموزش شبکه‌ها با ۲۰ نورون میانی شروع می‌شود. در شبکه با دو لایه میانی، ۱۰ نورون به هر یک از دو لایه میانی تخصیص می‌یابد. در شبکه با سه لایه میانی، ۱۰ نورون به لایه میانی اول و ۵ نورون برای هر یک از لایه‌های دوم و سوم میانی در نظر گرفته می‌شود.

در مسئله ارقام دست نویس فارسی و شبکه با دو لایه میانی، ۲۰ نورون به هر یک از دو لایه میانی تخصیص می‌یابد. برای شبکه با سه لایه میانی، ۲۰ نورون در لایه میانی اول و ۱۵ نورون برای هر یک از لایه‌های دوم و سوم میانی در نظر گرفته می‌شود.

در مسئله حروف چینی و شبکه با دو لایه میانی، ۲۰ نورون به هر یک از دو لایه میانی تخصیص می‌یابد. برای شبکه با سه لایه میانی، ۲۰ نورون برای هر یک از لایه‌های اول و دوم میانی و ۱۵ نورون برای لایه سوم میانی در نظر گرفته می‌شود.

معیار انتخاب بهترین اجرا در تمام شبیه سازیها، کوچک بودن ساختار شبکه (تعداد نورونهای لایه میانی) می‌باشد و در صورت یکسان بودن ساختار چند شبکه، بیشترین نرخ تشخیص ملاک انتخاب قرار می‌گیرد.

۴-۱) مقایسه الگوریتمها در شبکه‌های با دو لایه میانی

برای مسائل بررسی شده در این بخش، مقادیر پارامترهای الگوریتمهای مبتنی بر اتوماتای یادگیر، برای لایه اول میانی $\lambda_{OFF} = 0.02, \lambda_{ON} = 0.01$ و برای لایه دوم میانی $\lambda_{OFF} = 0.02, \lambda_{ON} = 0.002$ در نظر گرفته شده است. مقدار λ برای هر دو لایه ۰,۰۵می‌باشد.

جداول ۳ و ۴، نتایج الگوریتمهای مختلف را برای مسئله ارقام انگلیسی و مسئله Encoding در شبکه با دو لایه میانی نشان می‌دهد. هر الگوریتم ۲۰۰ بار اجرا شده و میانگین ۱۰ مورد از بهترین اجراها برای هر الگوریتم در جداول آمده است.

جدول ۳: مسئله ارقام انگلیسی در شبکه با دو لایه میانی

نام روش	تعداد نورونهای میانی	تعداد epoch آموزشی	نرخ تشخیص	خطای حداقل مربعات
Krylov	۷,۲	۸۱۳,۶	۱۰۰	۰,۰۱۲۷
OMA	۹,۴	۸۳,۲	۱۰۰	۰,۰۱۸۱
Iterative	۱۳,۲	۲۶	۹۰	۰,۰۲۱۲
S&D	۱۶	۲۶,۶	۱۰۰	۰,۰۱۹۸

جدول ۴: مسئله Encoding در شبکه با دو لایه میانی

نام روش	تعداد نورونهای میانی	تعداد epoch آموزشی	نرخ تشخیص	خطای حداقل مربعات
Krylov	۱۰	۴۴۱,۳	۱۰۰	۰,۰۱۲۱
OMA	۱۱,۶	۳۰۹	۱۰۰	۰,۰۱۴۸
Iterative	۱۳,۶	۶۱,۴	۱۰۰	۰,۰۱۵۷
S&D	۱۵	۵۶	۸۱,۲۵	۰,۰۱۵۴

جدول ۵، ۶ و ۷، نتایج الگوریتمهای مختلف را برای سه مسئله ارقام دست نویس فارسی، مسئله XOR سه بیتی و مسئله حروف چینی در شبکه با دو لایه میانی نشان می دهد. هر الگوریتم ۱۰۰ بار اجرا شده و میانگین ۱۰ مورد از بهترین اجراها برای هر الگوریتم در جداول آمده است.

جدول ۵: مسئله ارقام دست نویس فارسی در شبکه با دو لایه میانی

نام روش	تعداد نورونهای میانی	تعداد epoch آموزشی	نرخ تشخیص	خطای حداقل مربعات
Krylov	۱۵,۴	۲۱۶,۳	۹۹,۴۴	۰,۰۰۷۷۱
OMA	۱۷,۵	۳۰۵,۹	۹۹,۸۴	۰,۰۱۲۱
S&D	۲۸,۴	۶۲,۶	۹۵,۹۲	۰,۰۲۱۱۶
Iterative	۲۹,۲	۶۵,۱	۹۳,۴	۰,۰۲۰۷۳

جدول ۶: مسئله XOR سه بیتی در شبکه با دو لایه میانی

نام روش	تعداد نورونهای میانی	تعداد epoch آموزشی	نرخ تشخیص	خطای حداقل مربعات
Krylov	۱۲	۱۰۷۲,۴	۱۰۰	۰,۰۱
OMA	۱۲,۲	۱۲۰۹	۱۰۰	۰,۰۱۰۴
Iterative	۱۲,۷	۳۳۲,۳	۱۰۰	۰,۰۱۳۶
S&D	۱۳	۴۵۲,۳	۱۰۰	۰,۰۱۴۲

جدول ۷: مسئله حروف چینی در شبکه با دو لایه میانی

نام روش	تعداد نورونهای میانی	تعداد epoch آموزشی	نرخ تشخیص	خطای حداقل مربعات
Krylov	۲۰,۶	۲۴۲۸,۶	۱۰۰	۰,۰۱۷۹۶
OMA	۲۳,۸	۲۱۹۰	۱۰۰	۰,۰۱۶۸۳
Iterative	۳۶,۶	۱۹۲,۹	۱۰۰	۰,۰۱۳۶۸
S&D	۳۷	۲۱۵,۴	۱۰۰	۰,۰۱۷۴۶

۴-۲) مقایسه الگوریتمها در شبکه‌های با سه لایه میانی

برای مسائل بررسی شده در این بخش، مقادیر پارامترهای الگوریتمهای مبتنی بر اتوماتای یادگیر، برای لایه اول میانی $\lambda_{OFF}=0.02,\lambda_{ON}=0.01$ و برای لایه‌های دوم و سوم میانی $\lambda_{OFF}=0.02,\lambda_{ON}=0.002$ در نظر گرفته شده است. مقدار λ برای هر سه لایه ۰،۰۵ می‌باشد.

جداول ۸، ۹ و ۱۰، نتایج الگوریتمهای مختلف را برای سه مسئله ارقام انگلیسی، مسئله Encoding **و** مسئله حروف چینی در شبکه با سه لایه میانی نشان می دهند. هر الگوریتم ۵۰ بار اجرا شده و میانگین ۱۰ مورد از بهترین اجراها برای هر روش در جداول آمده است.

جدول ۸: مسئله ارقام انگلیسی در شبکه با سه لایه میانی

نام روش	تعداد نورونهای میانی	تعداد epoch آموزشی	نرخ تشخیص	خطای حداقل مربعات
Krylov	۸,۱	۱۹۶۳,۶	۱۰۰	۰,۰۱۱۲
OMA	۱۱,۴	۱۱۸۶,۷	۱۰۰	۰,۰۱۴۷
Iterative	۱۵,۳	۱۹۱,۸	۹۱	۰,۰۱۹۶
S&D	۱۷,۷	۲۱۹,۹	۷۴	۰,۰۲۱

جدول ۹: مسئله Encoding در شبکه با سه لایه میانی

نام روش	تعداد نورونهای میانی	تعداد epoch آموزشی	نرخ تشخیص	خطای حداقل مربعات
Krylov	۱۱,۴	۲۶۹۰,۹	۱۰۰	۰,۰۱۲۲
OMA	۱۳,۷	۲۷۵۴,۲	۱۰۰	۰,۰۱۲۹
Iterative	۱۵,۶	۵۴۱,۴	۱۰۰	۰,۰۱۱۹
S&D	۱۶,۸	۵۳۵,۲	۸۳,۷۵	۰,۰۱۶۶

جدول ۱۰: مسئله حروف چینی در شبکه با سه لایه میانی

نام روش	تعداد نورونهای میانی	تعداد epoch آموزشی	نرخ تشخیص	خطای حداقل مربعات
Krylov	۳۲,۵	۳۳۷۸,۸	۱۰۰	۰,۰۱۹۷۲
OMA	۴۱,۶	۲۵۶۴	۱۰۰	۰,۰۲۰۸۸
Iterative	۵۲,۶	۶۱۳	۱۰۰	۰,۰۱۹۹۵
S&D	۵۳,۹	۶۳۴,۲	۱۰۰	۰,۰۱۹۱۹

جدول ۱۱، نتایج الگوریتمهای مختلف را برای مسئله ارقام دست نویس فارسی در شبکه با سه لایه میانی نشان می‌دهد. هر الگوریتم ۱۰ بار اجرا شده و میانگین این اجراها برای هر روش در جدول ۱۱ آمده است.

جدول ۱۱: مسئله ارقام دست نویس فارسی در شبکه با سه لایه میانی

نام روش	تعداد نورونهای میانی	تعداد epoch آموزشی	نرخ تشخیص	خطای حداقل مربعات
Krylov	۲۶,۴	۱۳۰۴	۱۰۰	۰,۰۲۳۵
OMA	۳۱	۲۵۳,۲	۹۹,۴۸	۰,۰۰۸۷
S&D	۴۴,۹	۱۵۱,۸	۹۳,۵۲	۰,۰۳۰۶
Iterative	۴۶,۶	۱۴۷,۶	۹۳,۳۲	۰,۰۳۰۹

۵) نتیجه گیری

در این مقاله، الگوریتمی بر اساس اتوماتای یادگیر برای تعیین ساختار شبکه‌های عصبی با ساختار لایه ای دلخواه ارائه گردیده است. این الگوریتم یک نسخه تعمیم‌یافته از الگوریتم بقاء می‌باشد که از اتوماتای یادگیر و الگوریتم یادگیری انتشار خطا به عقب برای تعیین ساختار شبکه استفاده می‌کند. الگوریتم بقاء تعمیم یافته برای شبکه‌های لایه‌ای، ساختاری نزدیک به بهینه یعنی شبکه‌هایی با ساختار کوچک ایجاد می‌کند. الگوریتم بقاء تعمیم‌یافته (با دو اتوماتای یادگیر کرایلو و مهاجرت اشیاء) با دو الگوریتم هرس به نام‌های S&D و تکراری مقایسه شده است. مسائل ارقام دست نویس فارسی، ارقام انگلیسی و XOR سه بیتی، Encoding و حروف چینی مسائل بررسی شده هستند که برتری الگوریتم بقاء تعمیم‌یافته‌ای که از اتوماتای یادگیر کرایلو برای تعیین ساختار شبکه استفاده می‌کند را نسبت سایر الگوریتمها در شبکه شامل دو و سه لایه میانی نشان می دهند..

مراجع

[1] Lin, J. H. and Vitter, J. S. (1991). “Complexity Results on Learning by Neural Nets.” Machine Learning, Vol. 6, PP. 211-230.

[2] Castellano, G., Fanelli, A. M. and Pelillo, M. (1997). “An Iterative Pruning Algorithm for Feed forward Neural Networks”, IEEE Transactions on Neural Networks, Vol.8, No.3, PP.519-531.

[3] Kruschke, J. H. (1989). “Improving generalization in backpropagation networks.” Proc. of Int. Joint Conf. on Neural Networks, Vol. I, PP. 443-447.

[4] Reed, R. (1993). “Pruning Algorithms---A survey” IEEE Trans. on Neural Networks, Vol. 4, No. 5, PP. 740-747.

[5] Sietsma, J. and Dow, R.J.F. (1991). ”Creating Artificial Neural Networks That Generalize”, Neural Networks, Vol.4, PP. 67-79.

[6] Beigy, H. and Meybodi, M. R. (1998). “A fast method for determining the number of hidden units in feedforward neural networks.” Proc. of CSIC-97, Tehran, Iran, PP. 414-420(In Persian).

[7] Kwok, T. Y. and Yeng, 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.

[8] 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.

[9] Nabhan, T. M. and Zomaya, A. Y. (1994). “Toward neural networks structures for function approximation.” Neural Networks, Vol. 7, No. 1, PP. 89-99.

[10] 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.

[11] Yao, X. and Liu, Y. (1997). “A new evolutionary system artificial neural networks.” IEEE Trans. on Neural Networks, Vol. 8, No. 3, PP. 694-713.

[12] Beigy. H. and Meybodi, M. R. (1999). “Optimization of topology of neural networks using learning automata.” Proc. of 4th Annual Int. Computer Society of Iran Computer Conf. CICC-98, Tehran, Iran, PP. 417-428(In Persian).

[13] Beigy, H. and Meybodi, M. R. (1999). “A learning automata based algorithm for determination of optimal number of hidden units in three layers feedforward neural networks.” Journal of Amirkabir, Tehran, Iran(In Persian).

[14] Meybodi, M. R. and Beigy, H. (1999). “Neural Network engineering using learning automata: determination of desired size for three layer feedforward neural network.” Technical Reports, Computer Eng. Dept. Amirkabir University of Technology, Tehran, Iran (In Persian).

[15] Narendra, K. S. and Thatachar, M. A. L. (1974). “Learning Automata: A Survey,” IEEE Trans. on Systems, Man, and Cybernetic, Vol. SMC-4. , PP. 323-334.

[16]Meybodi, M. R. and Beigy, H., “A Note on Learning Automata Based Schemes for Adaptation of BP Parameters”, Journal of Neurocomputing, Vol. 48, No. 4, pp. 957-974, October 2002.

[17]Beigy, H. and Meybodi, M. R. “Backpropagation Algorithm Adaptation Parameters Using Learning Automata”, International Journal of Neural System, Vol. 11, No. 11, No. 3, PP. 219-228, 2001.

[18]Meybodi, M. R. and Beigy, H., "New Learning Automata Based Algorithms for Adaptation of Backpropagation Algorithm Parameters", International Journal of Neural System, Vol. 12, No. 1, PP. 45-67, 2002.
[19]Adibi, P., Meybodi, M. R. and R. Safabakhsh, "Unsupervised Learning of Synaptic Delays based on Learning Automata in an RBF-Like Network of Spiking Neurons for Data Clustering", Journal of Neurocomputing, Elsevier Publishing Company, Accepted for publication.
[20] Mashoufi, B., Mehaj, M. B., Motamedi, A., and Meybodi, M. R., "Introducing an Adaptive VLR Algorithm Using Learning Automata for Multilayer Perceptron", IEICE Transactions on Information and Systems, Vol. E86-D, No. 3, pp. 495-609, March 2003.
[21] Anjidani, M.and Meybodi, M. R., "An Algorithm for designing Small Neural Networks with High Generalization Using Leaning Automata", Proceedings of 13th Iranian Electrical Engineering Conference, University of Zanjan, Zanjan, Iran, pp.326-332, May 10-12 2005
[22] Anjidani, M. and Meybodi, M. R., "Neural Network Engineering Using Learning Automata: Determination of the Number of Hidden Units for Multi-Layer Neural Networks and Adaptation of Vigilance Parameter of ART Neural Network", MS. Thesis, Amirkabir, Tehran, Iran, 2005.
[23] Narendra, K. S., and Thathachar, M. A. L., Learning Automata: An Introduction, Printice-Hall, 1989.