یک روش یادگیری برای شبکه‌های عصبی با استفاده از الگوریتمهای بهینه‌سازی تصادفی

کارولوکس

گروه مهندسی برق و کامپیوتر دانشکده فنی دانشگاه تهران

بهنام مهدوی

دانشکده مهندسی دانشگاه شهید چمران اهواز

چکیده:

صورتهای مختلف الگوریتمهای بهینه‌سازی تصادفی و کاربردهای متنوع آنها در مقالات مختلف مورد بحث قرار گرفته و می‌گیرد. در این مقاله یک گروه عمده از این الگوریتمها، بنام ژنتیک را معرفی نموده، سپس به یکی از کاربردهای ارزنده این الگوریتم به‌عنوان روشی برای یادگیری شبکه‌های عصبی خواهیم پرداخت. درخاتمه مزایای این الگوریتمها نسبت به روشهای متداول یادگیری (مانند روش پس انتشار خطا) را برخواهیم شمرد.

A Learning Method For Neural Networks Using Random Optimization Algorithms

C.Lucas, Ph.D.
Associate Professor,Tehran University

B.Mahdavi, M.Sc.
Lecturer of AHWAZ - University

ABSTRACT

There has been increased interest in and extensive discussion of the different forms of stochastic optimization algorithms in recent years. In this paper, a main group of these algorithms, called genetic algorithms, is introduced, and a magor application area of the genetic aptimization algorithms, namely the training of the neural networks, is discussed. The Conclusion includes a comparison of the proposed learning algorithm with conventional learing methods like the back propagation algorithm and an investigation of the advantages of genetic learning algorithm.

مقدمه :

الگوریتمهای تصادفی در سالهای اخیر توجه زیادی رابه‌خود جلب کرده‌اند و این به‌دلیل ارائه روشهای مؤثر در حل مسائل بهینه‌سازی مشکل و امکان رسیدن به نقاط بهینه مطلق می‌باشد.

همانگونه که می‌دانیم اکثر روشهای قطعی (Deterministic) دارای این اشکال عمده هستند که به‌محض رسیدن به اولین نقطه بهینه موضعی (Local) متوقف شده و توانائی خروج ازاین نقطه و حرکت به‌سوی نقطه بهینه مطلق (Global) را ندارند. بنابراین مدتهاست که کار بر روی الگوریتمهائی که بتوانند از نقاط بهینه موضعی فرار کنند شروع شده و تاکنون روشهای متفاوتی ارائه و مورد بررسی قرار گرفته است.

در این میان الگوریتمهای تصادفی به‌دلیل دارا بودن مکانیزم عملکرد ساده‌تر و نتیجتاً راحتی اجرا به‌کمک کامپیوتر مورد توجه بیشتری قرار گرفته‌اند.

دو گروه عمده دیگر از الگوریتمهای تصادفی،الگوریتمهای ژنتیک و تکاملی هستند که تاکید بیشتر در این مقاله بر روی الگوریتم ژنتیک خواهد بود. [۸] ، [۷] ، [۶] ، [۵] ، [۱]

الگوریتم ژنتیک اولین بار توسط Holland در Ann Arbor/Michigan مطرح گردید. این الگوریتم بهینه‌سازی توانا بر پایه اصول بیولوژیکی بنا نهاده شده است. در این الگوریتم جمعیتی از نقاط جدید با یکدیگر به رقابت می‌پردازند. نقاط جدید طبعاً خصوصیات نقاط تولیدکننده خود را به ارث می‌برند. Aarts.Eiben و Van Hee [۱]. همگرائی الگوریتم ژنتیک را با استفاده از روش آنالیز زنجیره بی‌نهایت مارکوف اثبات نموده‌اند.

یکی از کاربردهای ارزنده الگوریتمهای تصادفی، در مسئله یادگیری شبکه‌های عصبی است. شبکه‌های عصبی به‌دلیل پردازش موازی دارای قابلیت پردازش بالای اطلاعات می‌باشند. محاسباتی از قبیل حرکت یا ردیابی چشم یا تشخیص گفتار محاسبات بسیار پیچیده‌ای هستند که سیستمهای بیولوژیکی بطور عادی آن را انجام می‌دهند، در صورتی‌که این اعمال خارج از توان پرقدرت‌ترین کامپیوترهای موجود می‌باشد. شبکه‌های عصبی با توسل به مدلسازی سیستمهای بیولوژیکی این امکان را بوجود آورده است که بتوان مسائلی ازاین قبیل را حل نمود. هم‌اینک شبکه‌های عصبی در علوم مختلف مانند سیستمهای هوشمند مورد استفاده وبا مطالعه جدی قرار دارد.

ساختار کلی شبکه‌های عصبی تشکیل شده از عناصر غیرخطی یا عصبها (نورونها) واتصالاتی که این عصبها را به‌یکدیگر متصل می‌سازد که به هر یک از این اتصالات وزن مشخص اختصاص داده می‌شود.

هر شبکه عصبی از سه نوع لایه تشکیل می‌شود :

۱ - لایه ورودی.

۲ - لایه پنهانی.

۳ - لایه خروجی .

یک مسئله مهم در ارتباط با شبکه‌های عصبی، مسئله یادگیری است. دو نوع روش یادگیری برای شبکه‌های عصبی وجود دارد یکی با معلم و دیگری بدون معلم. در روش یادگیری با معلم، اشکال ورودی و خروجی مطلوب در دسترس است که با اعمال ورودی به شبکه ومقایسه خروجی شبکه با خروجی مطلوب، وزنهای موجود در شبکه را به‌گونه‌ای تغییر می‌دهیم تا اختلاف خروجی شبکه با خروجی مطلوب به حد قابل قبولی برسد. این اشکال را، اشکال مطلوب نیز می‌نامند. در روش با معلم با اعمال اشکال ورودی مطلوب به شبکه، وزنهای شبکه به‌گونه‌ای تغییر می‌یابند که پس از پایان یادگیری در صورت اعمال هر ورودی دلخواه، خروجی شبکه به سمت یکی ازاشکال مطلوب که دارای انطباق بیشتری با ورودی اعمال شده می‌باشد همگرا خواهد شد.

یکی از معروفترین روشهای یادگیری با معلم، روش پس‌انتشار خطا می‌باشد.[۲]

۱ - الگوریتم ژنتیک

به‌دست آوردن راه‌حل‌های آسانتر جهت بهینه‌سازی سیستمهای پیچیده همواره مد نظر بوده است. بسیاری از ابزارهای ریاضی در حل اینگونه مسائل ناتوان می‌باشد و از طرفی بررسی و جستجوی جامع تمامی حالتهای ممکن (تمامی نقاط واقع در فضای جوابها) نیز خارج از تصور است، بنابراین فقط نمونه‌برداری امکان‌پذیر می‌باشد. یکی از الگوریتمهای تصادفی که اساس کار آن بر مبنای نمونه‌برداری استوار است، الگوریتم ژنتیک می‌باشد.

در این الگوریتم نقاط برروی مجموعه [0،1] تعریف می‌شوند یعنی اعداد به‌صورت رشته‌های باینری و بطول K بیان می‌شوند. الگوریتم ژنتیک روشی است برمبنای جستجو در فضای $[0،1]^k$. با استفاده از تابع توزیع احتمال P(t)، که این تابع در طول اجرای الگوریتم مرحله به مرحله به‌سمت محدوده‌ای که جواب در آن قرار دارد بایاس می‌شود.

ادعای فوق را می‌توان بسادگی مشاهده نمود اگر که، مجموعه $[0،1]^k$ را به‌صورت یک فضای K بعدی در نظر گرفته و با استفاده از تبدیل فوق صفحه مورد بررسی قرار دهیم.

در روش تبدیل فوق صفحه، فضا به‌صورت فوق صفحه‌هائی با ابعاد مختلف در نظر گرفته می‌شود که مقدار متوسط تابع هدف (تابع معیار یا کیفیت) بر روی هریک از این فوق صفحه‌ها متفاوت می‌باشد و با احتمال زیاد می‌توان گفت، نقطه جواب بر روی فوق صفحه‌ایست که مقدار متوسط تابع هدف بر روی آن

زیاد باشد.

با استفاده از روش تبدیل فوق صفحه می‌توان نشان داد که در حین اجرای الگوریتم ژنتیک، تابع توزیع احتمال P(t) بگونه‌ای تغییر می‌یابد که احتمال نمونه‌برداری از فوق صفحه‌هائی که دارای مقدار متوسط تابع هدف بهتری می‌باشند، افزایش می‌یابد.[۳]

۲- مکانیزم عملکرد الگوریتم ژنتیک

مراحل مختلف اجرای الگوریتم ژنتیک ارائه شده توسط Holland [۳] ، عبارت است از :

۱ - ابتدا تعداد m نقطه را بطور تصادفی انتخاب کرده (این m نقطه تشکیل مجموعه B(t) را می‌دهند) و مقدار تابع هدف U. را به‌ازاء هریک از m نقطه متعلق به مجموعه B(t) به‌دست می‌آوریم :

U_i i = i....,m = مقدار تابع هدف به‌ازاء نقطه iام

البته لازم به تذکر مجدد است که نقاطی را که مورد ارزیابی قرار می‌دهیم بر روی مجموعه $[0،1]^k$ تعریف می‌شوند. یعنی نقاط را به‌صورت اعداد باینری و بطول K بیان می‌کنیم. برای راحتی، m را نیز یک عدد زوج در نظر می‌گیریم .

۲ - مقادیر U_i به‌دست آمده از مرحله یک را نرمالیزه می‌کنیم

$$\hat{U}(x_j) = \frac{U(x_j)}{\sum_{i=1}^{m} U_i} \qquad j = 1,...,m$$

که در آن منظور از U(Xj) ، مقدار تابع هدف به‌ازاء نقطه Xj می‌باشد.

۳ - در این مرحله که در واقع همان مرحله انتخاب است، از میان M نقطه متعلق به B(t) ، نقاطی را که مقدار تابع هدف را بهتر می‌کنند با احتمال بیشتری نسبت به سایر نقاط، دوباره تکرار می‌نمائیم. برای این منظور به هر نقطه Xj ، تابع توزیع احتمال P(Xj,t) را متناسب با مقادیر نرمالیزه شده‌ای که در مرحله قبل محاسبه شده است، نسبت داده و با استفاده ازاین تابع توزیع احتمال، m نقطه جدید (مجموعه B′(t)) را ازروی m نقطه قدیمی تولید می‌نمائیم.

۴ - قبل ازتشریح این مرحله که در واقع مرحله تولید نقاط جدید می‌باشد، لازم است ابتدائاً اپراتورهای ژنتیکی را تعریف نمائیم. از این اپراتورها جهت تغییر نقاط موجود در مجموعه B′(t) و تشکیل مجموعه جدید B″(t) ، استفاده می‌شود.

الف - اپراتور تحول

همانطور که قبلاً گفتیم، نقاط Xj به‌شکل اعداد باینری با طول K هستند (یعنی هر عدد به‌صورت رشته‌ای از صفر و یک بطول K می‌باشد). ابتدا یک عدد تصادفی بین 1 تا K با توزیع احتمال یکنواخت تولید کرده، این عدد را L می‌نامیم L∈{1،...،K}. سپس در عدد باینری مورد نظر که K عنصر دارد. L امین عنصر را مشخص کرده، درصورت صفر بودن به یک و در صورت یک بودن به صفر تبدیل می‌کنیم.

ب - اپراتور تقاطع

قبلاً گفتیم که برای راحتی M (تعداد نقاط) را زوج انتخاب می‌کنیم. M نقطه موجود را به‌طور تصادفی به گروههای دوتایی تقسیم می‌کنیم. بنابراین M/2 زوج نقطه خواهیم داشت. فرض کنیم بخواهیم اپراتور تقاطع را برروی یکی ازاین زوجها اعمال کنیم. همانند اپراتور تحول، عدد تصادفی L بین 1 تا K، با توزیع احتمال یکنواخت را تولید کرده، سپس در هر دوعدد باینری موردنظر که هریک K عنصر دارند، L امین عنصر را مشخص کرده و عناصر واقع در سمت راست Lامین عنصر را در هر دو عدد باینری با یکدیگرجابجا می‌کنیم.

ج - اپراتور برگردان

همانند اپراتورهای قبلی، دراینجا نیز یک عدد تصادفی L بین ۱ تا K ، {L∈{۱،...،k با توزیع احتمال یکنواخت تولید کرده، سپس در عدد باینری مورد نظر که K عنصر دارد، L امین عنصر را مشخص کرده، عناصر واقع در سمت راست آنرا معکوس می‌کنیم، یعنی اگر صفر باشد به یک و اگر یک باشد به صفر تبدیل می‌شود.

با توجه به تعریف سه اپراتورژنتیکی فوق، در این مرحله نقاط مجموعه B′(t) را به‌توسط اپراتورهای فوق تغییر داده ، M نقطه جدید (مجموعه B″(t)) تولید کرده به مرحله ۲ برمی‌گردیم (اپراتورهای تقاطع و تحول بیش از اپراتور برگردان مورد استفاده قرار می‌گیرند).

D.Whitley et al ، [۴] ، شکل دیگری از الگوریتم ژنتیک را با اسم GENITOR ارائه می‌دهند. مرحله اول و دوم این روش همانندروش قبل می‌باشد. تفاوتی که مرحله سوم این روش با روش قبل دارد در اینست که بجای تولید M نقطه جدید از روی M نقطه موجود، فقط دو نقطه متفاوت انتخاب شده، سپس با استفاده از اپراتور تقاطع با دو نقطه تقاطع، این دو نقطه را تبدیل به دو نقطه جدید می‌کنیم. به‌عنوان مثال :

110 1001 111 ⟶ 110 1101 111

010 1101 001 ⟵ 010 1001 001

سپس به‌طور تصادفی یکی ازاین دو نقطه به‌دست آمده را حذف

کرده و نقطه دیگر را جایگزین کم‌رتبه‌ترین نقطه در مجموعه M نقطه‌ای قبلی می‌کنیم (منظور از کم‌رتبه‌ترین نقطه، نقطه‌ای است که تابع هدف به‌ازاء این نقطه، کمترین مقدار خود را نسبت به سایر نقاط در جمعیت داشته باشد). سپس مقدار تابع هدف را به‌ازاء نقطه جدید محاسبه و نقاط موجود را دوباره به‌ترتیب رتبه مرتب کرده، الگوریتم را تکرار می‌کنیم.

۳- الگوریتم ژنتیک ویادگیری شبکه‌های عصبی

قبل از ورود به جزئیات روش ژنتیک برای یادگیری بد نیست به این نکته توجه کنیم که استفاده ازالگوریتم ژنتیک در یادگیری شبکه‌های عصبی نه‌تنها از نقطه نظر شبکه‌های عصبی حائز اهمیت می‌باشد بلکه از جهت تست الگوریتم ژنتیک نیز ارزشمند می‌باشد. تا قبل از معرفی الگوریتم ژنتیک به‌عنوان روشی برای یادگیری، محققین از توابع تست استاندارد شامل مسائلی با کدگذاری کمتر از ۵۰ بیت برای تست الگوریتم ژنتیک استفاده می‌کردند و حال آنکه در مسئله یادگیری کدگذاری حتی بالغ بر بیش از صدها وهزارها بیت می‌باشد. بنابراین نتایج به‌دست آمده نه تنها دلالت بر چگونگی استفاده از الگوریتم ژنتیک در یادگیری شبکه‌های عصبی دارد بلکه نشان‌دهنده نحوه تعمیم الگوریتم ژنتیک، برای دستیابی به جوابهای دقیق، به مسائلی با ابعادی بزرگتر از آنچه تاکنون وجود داشته می‌باشد همانطور که قبلاً نیز گفتیم مسئله یادگیری در واقع همان بهینه‌سازی وزنهای یک شبکه عصبی است. فرض می‌کنیم که ساختار شبکه عصبی مورد نظر مشخص می‌باشد یعنی تعداد عصبها و نحوه اتصال بین آنها و لایه‌های مختلف مشخص است. قبلاً گفتیم که در الگوریتم ژنتیک اعداد در مود باینری بیان می‌شوند. بنابراین می‌بایست مسئله را به‌صورت یک رشته باینری کدگذاری نمود. برای این منظور می‌بایست برای هریک از وزنهای شبکه رنج محدودی را در نظر گرفت. به‌عنوان مثال می‌توان به هر وزن یک رشته باینری بطول هشت بیت اختصاص داد (یک بیت را هم به‌عنوان بیت علامت انتخاب می‌کنیم) که در این‌صورت محدوده تغییرات مقادیر وزنها ۱۲۷ تا ۱۲۷- خواهند بود. الگوریتم ژنتیک جمعیتی از رشته‌های باینری که هر کدام مجموعه‌ای از وزنها برای تمام شبکه می‌باشند را ایجاد می‌کنند. بنابراین در این‌حالت هر رشته، از کنار هم قرار دادن مقادیر کد شده وزنهای شبکه تشکیل می‌گردد به‌منظور یادآوری، در اینجا یکبار دیگر روند کلی اجرای الگوریتم ژنتیک را بیان خواهیم نمود:

در ابتدا جمعیتی از نقاط (در اینجا منظور از نقطه همان مجموعه وزنهای کد شده شبکه است که به‌صورت یک رشته در کنار یکدیگر قرار گرفته‌اند) به‌طور تصادفی انتخاب می‌شود. بعد از تولید هر یک از رشته‌های باینری، یک مفسر، این اطلاعات باینری را به‌شکل وزنهای اتصالات موجود در شبکه به‌کار می‌گیرد. سپس شبکه را همانند روش پس انتشار خطا به‌کار می‌اندازیم به این ترتیب که ورودیهای مورد نظر را به شبکه اعمال کرده و مجموع مربعات خطا در خروجی شبکه را که نشان‌دهنده عملکرد شبکه و معیاری جهت ارزش‌گذاری به وزنهای به‌کار رفته می‌باشد را محاسبه می‌کنیم. همانطور که قبلاً گفتیم الگوریتم ژنتیک از این مقادیر (مجموع مربعات خطاها) جهت رتبه‌بندی رشته‌ها استفاده شده و در مراحل بعدی، سیاست تولید نقاط جدید از مقادیر این رتبه‌ها استفاده خواهد نمود.

قبلاً الگوریتم genitor را به‌عنوان حالت خاصی از الگوریتم ژنتیک معرفی نمودیم. در آزمایشات به‌عمل آمده بر روی سه مدار:

۱- XOR

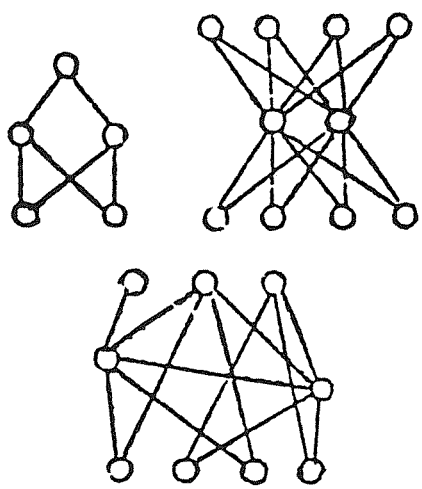
۲- جمع کننده دوبیتی

۳- کدگذار 4-2-4

از الگوریتم GENITOR برای یادگیری استفاده شد و در سه مورد نیز الگوریتم GENITOR جواب قابل قبول را به‌دست آورد. نوع تابع غیرخطی به‌کار رفته در شبکه‌ها یکسان نمی‌باشد و عمدتاً از دونوع تابع غیرخطی، سیگموئید و باینری استفاده شده است. از تابع سیگموئید برای لایه‌های پنهانی و خروجی و از تابع باینری برای لایه ورودی.

در شکل (۱) شبکه‌های به‌کار رفته برای هریک از مثالهای فوق نمایش داده شده است.

علت استفاده از تابع سیگموئید در لایه خروجی، اینست که پیوستگی خطا در گره‌های خروجی، از نظر عملکرد الگوریتم بهتر است چراکه در این‌صورت الگوریتم بهتر می‌تواند تفاوت بین عملکرد نقاط مختلف را تشخیص دهد. همانطور که قبلاً



شکل ۱ چهار شبکه بکار رفته برای تست الگوریتم ژنتیک

گفتیم حفظ تنوع گونه‌ها یک مسئله مهم در استفاده از الگوریتم ژنتیک می‌باشد و به همین دلیل نیز در آزمایشات فوق از روش اپراتور تحول تطبیقی استفاده می‌نمائیم.[۴] نتایج حاصل ازانجام آزمایشات فوق در جدول (۱) زیر نشان داده شده است.

جدول ۱

تعداد نقاط (M)	متوسط تکرار	مسئله
۲۰۰	۲۰۰۰	کدگذار ۴-۲-۴
۷۰	۱۰۰	مدار XOR
۳۰۰	۱۵۰۰۰	جمع کننده دوبیتی (می‌نیمال)

۴- مزایا و معایب الگوریتمهای تصادفی نسبت به روشهای یادگیری متداول

روش یادگیری شبکه‌های عصبی با استفاده از الگوریتمهای تصادفی (که یک روش یادگیری با معلم می‌باشد) دارای فوائد بسیاری نسبت به سایر روشهای یادگیری متداول می‌باشد. این الگوریتمها بسادگی قابل گسترش می‌باشند وجوابهای مطلق را به‌دست می‌آورند. شاید بتوان گفت که بزرگترین فایده آنها در اینست که هیچ شرطی و فرضی بر روی فضای جستجو قائل نمی‌شوند. به‌عبارت دیگر در هنگام یادگیری الگوریتمهای تصادفی هیچ احتیاجی به شناخت نحوه استفاده وزنها در شبکه ندارند و همینطور هیچ احتیاجی به عمل گرادیان‌گیری نیست.

البته با وجود تمام محسنات ذکر شده، از معایب این الگوریتمها سرعت پائین آنها نسبت به روشهای دیگر یادگیری (مانند پس‌انتشارخطا) می‌باشد.

۵- نتیجه‌گیری

با توجه به مزایای الگوریتمهای بهینه‌سازی تصادفی، نسبت به روشهای قطعی (یافتن نقاط بهینه مطلق، سادگی در ساختار عدم وابستگی به فرم مسئله) و با توجه به گسترش مسائل بهینه‌سازی در علوم مختلف و لزوم دستیابی به روشهای مطمئن‌تر، دقیق‌تر وعمومی‌تر برای حل آنها، می‌توان امیدوار بود که الگوریتمهای تصادفی نقش بیشتری را در این زمینه برعهده گیرند. کمااینکه روند رو به گسترش بکارگیری آنها در حوزه‌های مختلف، موید این مطلب می‌باشد. در این مقاله به یکی از این کاربردها به‌عنوان روشی برای یادگیری شبکه‌های عصبی پرداختیم.

مراجع :

1- *Aarts, E.H.L, Eiben, A.E, VanHee, K.M. "Global Convergence of Genetic Algorithms: an Infinite Markov Chain Anaslysis" PPSN, First International Workshop on Parallel Problem Solving form Nature, October 1-3, 1990.*

2- *Lippman, R.P. "An Introduction to Computing with Neural Nets". IEEE Acoustic, Speech, and Signal Processing Magazine, vol. 4, April 1987, PP.4-22.*

3- *Holland, J.H. "Searching Nonlinear Functions for High Values" Applied Mathematics & Computation 32: 255-274 (1989).*

4- *Whitley, D,Starkweather. T and Boyart, C "Genetic Alyorithms and Neural Network: Optimiziny Connections and Connectivity" Parallel Computiny 14 (1990), 347-361 North-Holland.*

5- *Hoffmeister, F. and Back, T. "Genetic Algorithms and Evolution Strategy: Similarities and Differences" PPSN, First International Workshop on Parallel Problem Solving from Nature, october 1-3, 1990.*

6- *Schwefel, H.P. "Collective Learning in Evolutionary Procsses" Proc. of the Int'l Conf. on Evolution and Nonlinear Economics, Austin/Texas, April 1989.*

7- *King, R.M. "ESP: Placement by Simulated Evolution" IEEE Transaction on Computer-Aided Design, vol.8, NO.3, March 1989.*

8- *Rechenberg, I. "Evolutionsstrategie" Problemata Series Number 14. Frommann-Holzboog, Stuttgart-Bad, Cannstatt, 1973.*