ELearn++[1]: یک الگوریتم جدید برای یادگیری افزایشی در شبکه های چندلایه پرسپترونی

علی صادقی نایینی[*]، حمید بیگی[†]

چکیده

هدف الگوریتم های یادگیری افزایشی در شبکه های چند لایه پرسپترونی، حفظ نتایج فازهای آموزشی قبلی و بهبود عملکرد شبکه با آموزش انحصاری آن بر روی نمونه های جدید است. الگوریتمهایی که تاکنون برای یادگیری افزایشی در یک شبکه چند لایه پرسپترونی پیشنهاد شده اند قابلیت تعریف کلاسهای جدید را در اختیار نمی گذارند. در این مقاله الگوریتمی برای یادگیری افزایشی در شبکه های چند لایه پرسپترونی مورد بررسی قرار می گیرد که این نقطه ضعف را رفع کرده است. این الگوریتم از ترکیب افزایشی تعدادی شبکه یادگیر ضعیف که هریک مربوط به تعدادی از نمونه های آموزشی است که در طول زمان به سیستم ارائه شده اند، یک شبکه یادگیر قوی می سازند و قابلیت پذیرش کلاسهای جدید معرفی شده توسط نمونه های تازه وارد را نیز دارا می باشند. در اینجا تاثیر پارامترهای مختلف بر میزان کارایی الگوریتم مورد بررسی قرار گرفته، و بوسیله نتایج بدست آمده، عملکرد الگوریتم مورد تحلیل قرار خواهد گرفت. به علاوه سه راه برای بهبود عملکرد چنین الگوریتمی پیشنهاد شده است. نتایج حاصل از پیاده سازیهای صورت گرفته، حاکی از موفقیت این روشها در بهبود عملکرد الگوریتم نسبت به نسخه اصلی می باشد تاجایی که استفاده همزمان از این روشها منجر به کاهش پنجاه درصدی خطا نسبت به نسخه اصلی الگوریتم گردیده است.

کلمات کلیدی

شبکه عصبی، چند لایه پرسپترونی، یادگیری، افزایشی، همجوشی، رای گیری اکثریت، توزیع احتمال، ترکیب فرضیات.

ELearn++: A novel incremental learning algorithm for MLP neural networks

Ali Sadeghi Naini
Amirkabir University of Technology
a_sadeghi@aut.ac.ir

Hamid Beigy
Sharif University of Technology
beigy@ce.sharif.edu

Abstract

The incremental learning algorithms in multilayer perceptron save the result of previous learning phases and enhance the performance of the network by training it on new samples exclusively. So far, introduced incremental learning algorithms for multilayer perceptron has not been capable of training the network on new samples which reflect a new class. In this article, an incremental learning algorithm is purposed which resolve this problem utilizing incremental combination of a set of weak learners' hypothesizes. First an original algorithm will be described. Next the influence of different parameters on efficiency of the algorithm is studied and exploring the results, the performance of the algorithm is analyzed. In addition, three approaches for the algorithm performance enhancement are proposed. The related results report the accomplished algorithm performance enhancement using these approaches in comparison with the original version of the algorithm. Applying these approaches simultaneously causes total error decreasing up to 50 percent comparing to the original algorithm result.

Keywords

Neural Network, Multilayer Perceptron, Learning, Incremental, Fusion, Weighted Majority, Hypothesis.

[*] دانشکده مهندسی کامپیوتر، دانشگاه صنعتی امیرکبیر، تهران، ایران، a_sadeghi@aut.ac.ir
[†] دانشکده مهندسی کامپیوتر، دانشگاه صنعتی شریف، تهران، ایران، beigy@ce.sharif.edu

۱- مقدمه

فاز یادگیری در شبکه های چند لایه پرسپترونی پر هزینه ترین مرحله در بکارگیری این شبکه ها به حساب می آید. در این فاز با ارائه مجموعه ای از داده ها آموزشی به شبکه، وزنهای مختلف شبکه تنظیم، و سپس در فاز آزمایش از آنها برای تخمین پاسخ شبکه به داده های ناآشنا استفاده خواهد شد. برای این منظور معمولا از الگوریتم انتشار خطا به عقب استفاده می شود که الگوریتمی کاملا برون خط است، بدین معنا که برای استفاده از این الگوریتم لازم است تمامی داده های آموزشی از ابتدا در اختیار الگوریتم قرار گیرد. اگر در کاربردی داده های آموزشی به طور قطعی و کامل از ابتدا مشخص نباشند و به مرور زمان بر تعداد آنها افزوده گردد، پس از هر بار به روز رسانی داده های آموزشی لازم است فاز یادگیری از ابتدا و با مجموعه جدید آموزشی شامل داده های آموزشی قدیمی و جدید پشت سر گذاشته شود. در این حالت نتایج فاز آموزش قبلی کاملا فراموش شده و از دست می رود، در نتیجه ما مجبور به پرداخت مکرر هزینه آموزش برای داده های آموزشی قدیمی هستیم. حال آنکه ممکن است در کاربردی داده های آموزشی در یک دنباله زمانی و تک تک به سیستم ارائه شوند. درمورد پروسه های غیر ایستا[۲] این مشکل بیشتر نمایان میشود. در برخی موارد مشکل حادتر می شود و آن هنگامی است که داده های آموزشی قدیمی دیگر در دسترس نباشند.

الگوریتم های یادگیری افزایشی[۳] برای حل این مشکل معرفی شده اند. هدف این الگوریتم ها حفظ نتایج فازهای آموزشی قبلی و بهبود عملکرد شبکه با آموزش آن تنها بر روی نمونه های جدید است. در واقع این الگوریتمها بدون آنکه دوباره بر روی نمونه های قدیمی، که ممکن است دیگر در دسترس نباشند، شبکه را آموزش دهند با رسیدن نمونه های جدید آن را با این نمونه ها نیز وفق داده و به روز می رسانند.

یک سیستم مبتنی بر یادگیری افزایشی سیستمی است که هنگام ورود نمونه یا نمونه های آموزشی جدید فرضیات و اطلاعات قبلی خود را بدون استفاده دوباره از نمونه های آموزشی قبلی به هنگام در می آورد. در واقع چنین سیستمی هنگام ارائه نمونه های جدید نتایج حاصل از فازهای قبلی آموزشی خود روی نمونه های قدیمی را از دست نمی دهد بلکه آنها را با توجه به نمونه های جدید بهبود می بخشد. به بیان دیگر چنین سیستمی تابع Y را بر اساس مجموعه نمونه آموزشی X1 می آموزد، سپس تابع بهبود یافته Z را بر اساس تابع Y و مجموعه نمونه آموزشی جدید X2 می آموزد و همینطور این روند را ادامه می دهد[۱] .

بر این اساس یک الگوریتم یادگیری افزایشی ایده آل برای شبکه های عصبی چند لایه پرسپترونی باید شش ویژگی زیر را دارا باشد[۲,۵]:

۱. [فشردگی[۴]]، از واحد ها و اتصالات افزایشی بر اساس یک رابطه خطی یا غیر خطی با تعداد نمونه های آموزشی استفاده نکند، در غیر اینصورت شبکه کاری غیر از ذخیره کامل تمام نمونه های آموزشی و پیدا کردن تابعی که از تمامی نمونه ها عبور می کند انجام نمی دهد.

۲. [رجوع نکردن به داده های آموزشی قدیمی]، با ورود نمونه های آموزشی جدید، شبکه فاز یادگیری افزایشی را فقط روی این نمونه ها، و نه روی مجموعه ای شامل این نمونه ها و نمونه های قدیمی، اجرا نماید.

۳. [شکل پذیری[۵]]، باید بتوان نمونه های آموزشی جدید را در هر زمانی به شبکه ارائه نمود و شبکه باید بتواند نتایج فازهای قبلی آموزش را با آموزش روی این نمونه های جدید گسترش دهد. نتایج جدید باید شامل ویژگی های نمونه های آموزشی قدیمی و جدید باشد، به ویژه اگر نمونه ها در یک دنباله زمانی و به مرور به شبکه ارائه شوند، عملکرد شبکه بر روی فضای مساله باید به تدریج بهبود یابد.

۴. [پایداری[۶]]، اگر الگویی که قبلا توسط شبکه دیده شده است به آن ارائه شود، پاسخ شبکه با یادگیری افزایشی در شرایط عدم وجود تناقض و اختلال[۷] (وجود تناقض و اختلال) باید (تقریبا) مطابق پاسخ قبلی و صحیح شبکه به آن الگو باشد.

۵. [مقاومت در برابر اختلال]، از آنجا که در یادگیری افزایشی تحلیل تمامی نمونه های آموزشی از قبل میسر نیست، وجود دو قابلیت ضروری است: اول آنکه شبکه با محلی کردن تاثیر نمونه های آموزشی دارای اختلال و تناقض در برابر آنها مقاومت کند، و دوم نتایج اشتباه و غیر قابل قبول ناشی از نمونه های متناقض تحت تاثیر آموزش بیشتر بر روی نمونه های صحیح قرار گرفته و محو شوند.

۶. [قدرت تعمیم[۸]]، شبکه آموزش دیده باید قابلیت کافی برای تعمیم داشته باشد تا بتواند به نمونه های ناآشنا پاسخ مناسب دهد.

آنچه مسلم است تامین این شش ویژگی در یک شبکه عصبی به طور همزمان و کامل کار ساده ای نیست و در برخی از موارد میسر نمی باشد. بنابراین با توجه به هر کاربرد سعی می شود در شبکه بین آنها توازن ایجاد شود. بر این اساس الگوریتمهای مختلفی برای یادگیری افزایشی در شبکه های چند لایه پرسپترونی پیشنهاد شده است. به عنوان نمونه الگوریتم ارائه شده در [۱] بدون نیاز به مجموعه آموزشی قبلی، ابتدا سعی می کند بدون تغییر در معماری شبکه و افزایش نورون ها و اتصالات آن، نمونه های جدید را به صورت افزایشی به شبکه آموزش دهد، و تنها در صورتی که شبکه بدلیل کم بودن تعداد نورون ها و اتصالات قادر به یادگیری افزایشی این نمونه ها نبود، نورون ها و اتصالات جدید را به شبکه می افزاید. به بیان دیگر این

الگوریتم برای یادگیری افزایشی نمونه های تازه وارد دو نوع تغییر را در شبکه اعمال میکند: ۱- تغییر وزنها که اگر موثر تشخیص داده شود اعمال می شود؛ ۲- افزایش نورون ها و اتصالات به شبکه که در صورت تاثیر نداشتن تغییر وزنها اعمال می شود. برای هر دوی این تغییرات باید محدودیت هایی در نظر گرفت تا شبکه و نتایج یادگیری افزایشی آن قابل اعتماد و پایدار باشند. در [۲] الگوریتمی برای یادگیری افزایشی یک شبکه چند لایه پرسپترونی ارائه شده است که از لحاظ نحوه عملکرد شبیه الگوریتم پیشنهادی در [۱] می باشد، یعنی در مرحله اول سعی می کند با تغییر وزنها و بدون بزرگ کردن شبکه نمونه های جدید را به صورت افزایشی به شبکه بیاموزد و تنها در صورتی نورون ها و اتصالات جدید را به شبکه می افزاید که مرحله اول آموزش ناموفق باشد. تفاوت این دسته از الگوریتم ها با الگوریتم قبلی در این است که این الگوریتم دارای روابطی برای یادگیری افزایشی است که به نمونه های قبلی آموزشی نیاز دارد، راه حلی که این الگوریتم به کار می برد استفاده از شبه نمونه هایی است که از پاسخ شبکه نیمه آموزش دیده در هر زمان، بدست می آید.

با آنکه این الگوریتمها و الگوریتمهای مشابهی که برای یادگیری افزایشی در یک شبکه چند لایه پرسپترونی پیشنهاد شده اند هر یک در حوزه کاربرد خود عملکرد نسبتا مناسبی داشته اند، اما همگی از یک نقطه ضعف اساسی رنج می برند: هیچ یک از این الگوریتمها قابلیت تعریف کلاس جدید را در اختیار نمی گذارند. به بیان دیگر با آنکه این الگوریتمها قابلیت ارائه تدریجی نمونه های آموزشی و در نتیجه یادگیری افزایشی را دارا می باشند، فرض می کنند که تمامی کلاسهای مساله طبقه بندی از ابتدا معلوم بوده و در اختیار آنها قرار می گیرد. در نتیجه با این الگوریتمها تنها می توان دانش شبکه را درمورد تعداد محدود و از ابتدا معینی از کلاسها به تدریج افزایش داد و اگر در کاربردی نیاز باشد تا شبکه قابلیت افزایش دانش خود را در مورد تعداد و توزیع کلاسها دارا باشد، این الگوریتمها از عهده ارائه راه حل ناتوان می مانند. در این مقاله الگوریتمی برای یادگیری افزایشی در شبکه های چند لایه پرسپترونی مورد بررسی قرار می گیرد که این نقطه ضعف را رفع کرده است. این الگوریتم که در [۳-۶]، Learn++ نامیده شده است، از ترکیب افزایشی تعدادی شبکه یادگیر ضعیف که هریک مربوط به تعدادی از نمونه های آموزشی است که در طول زمان به سیستم ارائه شده اند، یک شبکه یادگیر قوی می سازند و قابلیت پذیرش کلاسهای جدید معرفی شده توسط نمونه های تازه وارد را نیز دارا می باشند. در اینجا تاثیر پارامترهای مختلف بر میزان کارایی الگوریتم مورد بررسی قرار گرفته، و بوسیله نتایج بدست آمده، عملکرد الگوریتم مورد تحلیل قرار خواهد گرفت. به علاوه سه راه برای بهبود عملکرد این الگوریتم پیشنهاد شده است. نتایج حاصل از پیاده سازیهای صورت گرفته، حاکی از موفقیت این روشها در بهبود عملکرد الگوریتم نسبت به نسخه اصلی می باشد.

ادامه این مقاله بدین صورت زیر سازمان یافته است: ابتدا در بخش ۲ نسخه اصلی الگوریتم یادگیری افزایشی مورد بحث در این مقاله (Learn+) بررسی می شود، سپس در بخش ۳ روشهای پیشنهادی این مقاله برای بهبود کارایی این الگوریتم ارائه و الگوریتم پیشنهادی (ELearn++) بر این اساس معرفی خواهد شد. در بخشهای ۴ و۵ پیاده سازی صورت گرفته و نتایج حاصل از آن بررسی شده و در مورد این نتایج بحث و نتیجه گیری بعمل خواهد آمد.

۲- یاد گیری افزایشی مبتنی ترکیب فرضیات شبکه های یادگیر ضعیف

قبل از پرداختن به الگوریتم یادگیری افزایشی، الگوریتمی را معرفی می کنیم که فارغ از یادگیری افزایشی، برای ارتقا عملکرد یک شبکه ضعیف بوسیله تولید چندین فرضیه[۹] مختلف و سپس ترکیب آنها، [۳-۶]، توسعه یافته است. سپس نحوه استفاده از این الگوریتم در یادگیری افزایشی، و اصلاحات مورد نیاز در آن برای این منظور را بررسی خواهیم نمود.

۲-۱- الگوریتمی برای ایجاد یک شبکه قوی مبتنی بر ترکیب فرضیات تعدادی شبکه ضعیف

این الگوریتم برای ارتقا عملکرد یک شبکه یادگیر ضعیف، که عمل طبقه بندی را انجام می دهد، با تولید چندین فرضیه طبقه بندی ضعیف و سپس ترکیب آنها بوسیله رای گیری وزن دار روی کلاسهای پیشبینی شده به عنوان خروجی توسط هر فرضیه جداگانه، عمل می کند. این فرضیات با آموزش مجدد شبکه روی توزیع های مختلف انتخاب شده از پایگاه داده نمونه های آموزشی بدست می آید.

ورودی های این الگوریتم عبارتند از: دنباله ای از نمونه های برچسب خورده (داده های آموزشی، S)، یک الگوریتم یادگیری ضعیف، WeakLearn، و یک عدد صحیح، T، که مشخص کننده تعداد فرضیات (تعداد تکرار) تولید شده بوسیله، WeakLearn، می باشد. این الگوریتم، به صورت تکراری و با انتساب یک وزن به هر نمونه، توزیع S را بهنگام می کند. انتساب وزن در مورد خروجی یادگیر ضعیف، که روی زیر مجموعه انتخاب شده توسط این توزیع آموزش دیده، نیز صورت می گیرد. بهنگام سازی توزیع به گونه ای انجام می شود که تمرکز بیشتر روی نمونه های مشکل تر باشد.

در تکرار tام، این الگوریتم یک زیر مجموعه از داده های آموزشی را که مطابق توزیع D_t از پایگاه داده آموزشی اصلی، S=[(x1,y1),...,(xm,ym)]، انتخاب شده، به WeakLearn ارائه می کند. پس از آن،WeakLearn ، یک فرضیه (طبقه بندی کننده) را مانند، $h_t : X \to Y$، محاسبه می کند که کسری از مجموعه آموزشی را با توجه به D_t، بدرستی طبقه بندی می نماید. بنابراین هدف WeakLearn، یافتن فرضیه ای است که خطای آموزشی را کمینه کند:

$$\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i) \qquad (۱)$$

این الگوریتم برای عملکرد مناسب نیاز دارد که کارایی هر فرضیه(طبقه بند ضعیف) حداقل ٪۵۰ باشد، یعنی $\varepsilon_t < \frac{1}{2}$. توزیع اولیه، D_1، به صورت یکنواخت بر روی S انتخاب می شود، یعنی برای تمامی i ها $D_1(i) = \frac{1}{m}$. این مقدار دهی اولیه، منجر به شانس مساوی قرار گرفتن در زیر مجموعه آموزشی انتخاب شده، برای تمامی نمونه های S می گردد. رابطه بهنگام سازی این توزیع به شکل زیر است:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_{t+1}} \times \begin{cases} \beta_t & if\ ,h_t(x_t)=y_t \\ 1 & Otherwise \end{cases} \qquad (۲)$$

که $Z_t = \sum_i D_t(i)$، یک ثابت نرمال سازی است تا مطمئن شویم که D_{t+1}، یک توزیع معتبر خواهد بود و $\beta_t = \frac{\varepsilon_t}{(1-\varepsilon_t)}$. بدین ترتیب، نمونه های ساده تر که توسط h_t بدرستی طبقه بندی شده اند، از احتمال کمتر و نمونه های مشکل تر که نادرست طبقه بندی شده اند از احتمال بیشتری برای انتخاب در زیر مجموعه داده های آموزشی بعدی برخوردار خواهند شد.

پس از خاتمه Tامین تکرار، این الگوریتم، فرضیات ضعیف، h_t،...,h_1، را با محاسبه اکثریت وزن دار[۱۰] این فرضیات ضعیف ترکیب کرده، و یک فرضیه نهایی h_{final} را بصورت زیر بدست می آورد:

$$h_{final} = \arg\max_{y\in Y} \sum_{t:h_t(x)=y} \log(\frac{1}{\beta_t}) \qquad (۳)$$

برای یک نمونه مانند x ، h_{final} برچسب y را که، حاصل جمع وزنهای فرضیات ضعیفی را که این برچسب را پیشبینی کرده اند بیشینه کند، به عنوان خروجی تحویل می دهد. وزن فرضیه h_t، به صورت $\log(\frac{1}{\beta_t})$، تعریف شده تا وزن بیشتری به فرضیه با خطای کمتر داده شود.

نکته ای که باید به آن دقت شود این است که حداقل دقت هر شبکه یادگیر ضعیف باید ٪۵۰ باشد، اما استفاده از شبکه های یادگیر خیلی قوی به جای شبکه های یادگیر ضعیف در این الگوریتم توصیه نمی شود، زیرا این شبکه ها منجر به برازش بیش از حد[۱۱] روی داده ها می شوند.

۲-۲- Leran++ : یک الگوریتم یادگیری افزایشی

در الگوریتم بخش ۲-۱ نیازی به دانستن توزیع اصلی، $\underline{\boldsymbol{D}}$، که داده های آموزشی،S، از آن برداشته شده اند نیست. فرض کنید با ارائه مجموعه ای از داده های آموزشی،S، که از توزیع $\underline{\boldsymbol{D}}1 \subseteq \underline{\boldsymbol{D}}$، آمده اند به یک شبکه یادگیر، مجموعه ای از فرضیات تولید شده است. حال فرض کنید مجموعه جدیدی از داده ها که از توزیع $\underline{\boldsymbol{D}}2 \subseteq \underline{\boldsymbol{D}}$ آمده اند و برای شبکه ناآشنا هستند، به شبکه داده شود، اگر مجموعه داده قبلی که از توزیع $\underline{\boldsymbol{D}}1$ آمده اند، نمایش مناسبی از کل فضای نمونه ای[۱۲] $\underline{\boldsymbol{D}}$ باشند، شبکه روی نمونه های جدید نیز به خوبی کار می کند، در غیر اینصورت، می توان نتیجه گرفت مجموعه داده قبلی نمایش مناسبی از کل فضای نمونه ای $\underline{\boldsymbol{D}}$ نیست و $\underline{\boldsymbol{D}}1 \not\subset \underline{\boldsymbol{D}}2$. به بیان دیگر، شبکه روی نمونه های آن قسمت از فضای نمونه ای که نمونه های جدید از آن آمده اند، آموزش ندیده است. بنابراین می توان نمونه های جدید را به عنوان نمونه های سخت $\underline{\boldsymbol{D}}$ در نظر گرفت و شبکه را مجبور کرد نمونه های تازه وارد از $\underline{\boldsymbol{D}}2$ را با تولید فرضیات جدید برای این مجموعه جدید داده ها و سپس ترکیب تمام فرضیات تولید شده(برای $\underline{\boldsymbol{D}}1$ و $\underline{\boldsymbol{D}}2$) یاد بگیرد و به این ترتیب نمونه های ناآشنا را نیز بدرستی طبقه بندی کند. فرض اساسی آن است که توزیع $\underline{\boldsymbol{D}}$ ، توزیع $\underline{\boldsymbol{D}}2$ را نیز به خوبی $\underline{\boldsymbol{D}}1$ شامل می شود.

از آنجا که هدف الگوریتم ارائه شده در بخش ۲-۱ افزایش دقت طبقه بندی بود، آن الگوریتم روی توزیع های مختلف یک مجموعه آموزشی واحد کار می کرد و به این دلیل خطا بر روی نمونه های همان مجموعه آموزشی که بدرستی طبقه بندی نشده بودند، محاسبه می شد. به علاوه بهنگام سازی توزیع بر اساس فرضیه منفرد h_t و خطای نسبی آن، β_t صورت می گرفت. اما Learn++ نسخه متفاوتی از این الگوریتم را برای هر مجموعه داده جدید به کار می برد. در هر تکرار، t، روی مجموعه داده k$\underline{\boldsymbol{D}}$، زیر مجموعه های آموزشی (TRt) و آزمایشی (TEt) به صورت تصادفی و با توجه به D_t از مجموعه داده انتخاب می شوند. زیر مجموعه آموزشی در اختیار شبکه چند لایه پرسپترونی نسبتا کوچک (یادگیر ضعیف) قرار می گیرد و فرضیه h_t توسط آن تولید می شود. خطا روی اجتماع زیر مجموعه های آموزشی و آزمایشی محاسبه می شود. اگر خطا بزرگتر از ۰.۵ بود، h_t دور ریخته شده و زیر مجموعه های جدید آموزشی و آزمایشی تولید می شوند. سپس اکثریت وزن دار همه فرضیات تولید شده، H_t محاسبه می شود. این اکثریت وزن دار و خطای نسبی مربوط به آن برای بهنگام سازی توزیع، D_t، به کار برده می شوند. برای بدست آوردن خروجی شبکه به یک نمونه، از اکثریت وزن دار روی اکثریت وزن دار مربوط به هر مجموعه داده k$\underline{\boldsymbol{D}}$، استفاده می شود.

۲-۳- نقاط ضعف الگوریتم Learn++ و راههای بهبود آن

در نسخه اصلی الگوریتم Learn++ از رابطه $\log(\frac{1}{\beta_t})$ برای محاسبه امتیاز هر کلاس استفاده شده است. بدین ترتیب وزن بیشتری به فرضیه با خطای کمتر داده خواهد شد، اما اگر خطای یک فرضیه صفر باشد، که بسیار مطلوب است، این رابطه خطای تقسیم بر صفر را در

سیستم به دنبال خواهد داشت. در این حالت فرضیه مطلوب از رای گیری حذف خواهد شد. پیشنهاد ما برای رفع این مشکل استفاده از رابطه ساده $(1-\beta)$ برای محاسبه امتیاز هر کلاس است که منجر به چنین خطایی نخواهد شد.

پیشنهاد دیگری که به نظر می رسد موجب عملکرد سیستم گردد استفاده از WeakLearnerهایی با یک خروجی می باشد. در این حالت برای هر کلاس مساله یک شبکه وجود خواهد داشت که تعیین می کند یک داده به این کلاس تعلق دراد یا خیر. در نتیجه فارغ از تعداد کلاسهای مساله که ممکن است به تدریج بر تعداد آنها افزوده گردد، هر شبکه مامور شناسایی اعضای تنها یک کلاس خواهد بود.

روش دیگری که در اینجا برای بهبود عملکرد الگوریتم به کار گرفته شده است، هنجارسازی امتیازات کلاسها در رای گیری اکثریت وزندار می باشد. در الگوریتم Learn++ هنگامیکه کلاسهای جدیدی به سیستم معرفی می شوند، از آن مرحله به بعد WeakLearner ها دارای یک خروجی برای هر کلاس جدید خواهند بود. حال آنکه کلاسهای قدیمی در WeakLearner های مراحل قبلی نیز دارای خروجی بوده اند و لذا از شانس بیشتری برای پیروزی در رای گیری اکثریت برخوردار خواهند بود. در مرحله هنجارسازی امتیازات کلاسها، امتیاز کلاسهایی که جدیدا تعریف شده اند در ضریب تناسبی ضرب می شود تا با کلاسهای قدیمی که دارای امتیاز تجمعی بیشتری هستند در یک سطح قرار گیرند. انتظار می رود این هنجار سازی موجب کاهش خطای طبقه بندی گردد.

۳- ELearn++: نسخه بهبود یافته الگوریتم Learn++

با توجه به مطالب بیان شده در قسمت قبل، نسخه بهبود یافته الگوریتم Learn++ که در این مقاله ELearn++ نامیده شده است به صورت زیر است:

ورودی: برای هر مجموعه داده جدید گرفته شده از توزیع $\underline{\boldsymbol{D}}$k ، k=1,2,...,K

- دنباله ای از m نمونه: S=[(x1,y1),...,(xm,ym)]
- به ازاء هر کلاس موجود در مجموعه داده جدید یک شبکه چند لایه پرسپترونی نسبتا ضعیف(کوچک) با یک خروجی (WeakLearn)، در هر مرحله هر داده ورودی به کلاسی نسبت داده می شود که شبکه مربوط به آن کلاس در ازائ ارائه آن داده ورودی بزرگترین خروجی را تولید نماید.
- عدد صحیح Tk که نمایانگر تعداد تکرار است.

برای هر k=1,2,...,K مراحل زیر را انجام بده:

مقدار دهی اولیه احتمال انتخاب هر نمونه: $D_1(i)=\frac{1}{m}, \forall i$

برای هر t=1,2,..,Tk مراحل زیر را انجام بده:

۱. به طور تصادفی و با توجه به D_t ، زیر مجموعه های TRt و TEt (آموزش و آزمایش) را از S انتخاب کن.

۲. TRt را به WeakLearn ارائه کن.

۳. فرضیه خروجی شبکه، $h_t : X \to Y$ ، را دریافت کرده و خطا، $\varepsilon_t = \sum_{i:h_t(x_i)\neq y_i} D_t(i)$ ، را روی TRt +TEt محاسبه کن. اگر رابطه، $\varepsilon_t < \frac{1}{2}$ ، برقرار نبود، T را برابر t-1 قرار بده، h_t را دور بریز و به مرحله ۱ برگرد. در غیر این صورت خطای نسبی را محاسبه کن: $\beta_t = \frac{\varepsilon_t}{(1-\varepsilon_t)}$

۴. اکثریت وزن دار را و خطای عمومی را با توجه به همه فرضیات بدست آمده محاسبه کن:

$$H_t = \arg\max_{y\in Y} \sum_{t:h_t(x)=y} (1-\beta) \quad , \quad E_t = \sum_{i:H_t(x_i)\neq y_i} D_t(i)$$

۵. $B_t = \frac{E_t}{(1-E_t)}$ را محاسبه کن و توزیع D_t را بهنگام کن:

$$D_{t+1}(i) = \frac{D_t(i)}{Z_{t+1}} \times \begin{cases} B_t & \text{if } H_t(x_i)=y_i \\ 1 & \text{Otherwise} \end{cases} \quad , \quad Z_t = \sum_i D_t(i)$$

خروجی: اکثریت وزن دار را برای محاسبه خروجی نهایی (فرضیه نهایی) محاسبه کن:

$$h_{final}(x) = \arg\max_{y\in Y} \sum_k \left(\frac{C}{C_Y} * \sum_{t:h_t(x)=y} (1-\beta) \right)$$

که C تعداد کل مراحل، و C_Y تعداد مراحلی است که کلاس Y در آن وجود داشته است.($\frac{C}{C_Y}$ همان ضریب تناسب می باشد)

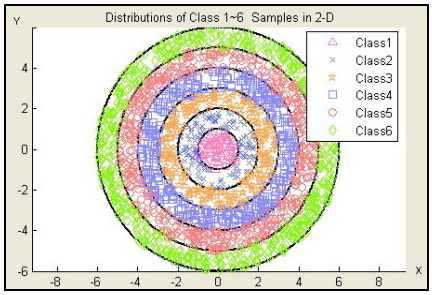
الگوریتم۱- الگوریتم ELearn++

در هر تکرار، t، روی مجموعه داده $\underline{\boldsymbol{D}}$k، زیر مجموعه های آموزشی (TRt) و آزمایشی (TEt) به صورت تصادفی و با توجه به D_t از مجموعه داده انتخاب می شوند. زیر مجموعه آموزشی در اختیار مجموعه ای از شبکه های چند لایه پرسپترونی نسبتا کوچک و با یک خروجی (یادگیر ضعیف) قرار می گیرد و فرضیه h_t توسط آن تولید می شود. خطا روی اجتماع زیر مجموعه های آموزشی و آزمایشی محاسبه می شود. اگر خطا بزرگتر از ۰.۵ بود، h_t دور ریخته شده و زیر مجموعه های جدید آموزشی و آزمایشی تولید می شوند. سپس اکثریت وزن دار همه فرضیات تولید شده، H_t محاسبه می شود. این اکثریت وزن دار و خطای نسبی مربوط به آن برای بهنگام سازی توزیع، D_t ، به کار برده می شوند. برای بدست آوردن پاسخ شبکه به یک نمونه جدید، از اکثریت وزن دار روی اکثریت وزن دار هنجار سازی شده مربوط به هر مجموعه داده $\underline{\boldsymbol{D}}$k، استفاده می شود.

۴- پیاده سازی و نتایج

مجموعه داده های بکار رفته جهت آموزش و آزمایش سیستم یادگیر در این پیاده سازی، همان مجموعه داده های بکار رفته در الگوریتم اصلی است تا امکان مقایسه نتایج فراهم آید. این مجموعه داده از پنج کلاس با ناحیه دایره ای و در یک فضای دو بعدی تشکیل شده است(شکل ۱). داخلی ترین دایره ناحیه کلاس ۱ و خارجی ترین دایره ناحیه کلاس ۵ است. ۶ مجموعه داده S1~S6 از این مجموعه داده انتخاب شده که S1,S2 شامل نمونه هایی از کلاس ۳،۱و۵ هستند. نمونه های S3,S4 علاوه بر کلاسهای قبلی کلاس ۴ و نمونه های

S5,S6 کلاس ۲ را نیز شامل می شوند. یک مجموعه داده به نام Test نیز از تمامی کلاسها تولید شده است. این مجموعه ها به ترتیب از S1 به الگوریتم داده شده اند و عملکرد الگوریتم پس از ارائه هر مجموعه داده جدید روی مجموعه داده Test سنجیده شده است. در نمودارهای خطا که در زیر مشاهده می کنید، محور عمودی میزان خطا و محور افقی تعداد مجموعه داده هایی را نشان می دهد که سیستم تاکنون بر روی آنها اموزش دیده است. وقتی کار الگوریتم روی هر یک از مجموعه های داده به پایان رسید، پس از آن و در مراحل بعد، این مجموعه داده مجددا به الگوریتم ارائه نشده است تا از اجرای یک یادگیری افزایشی مطمئن باشیم.



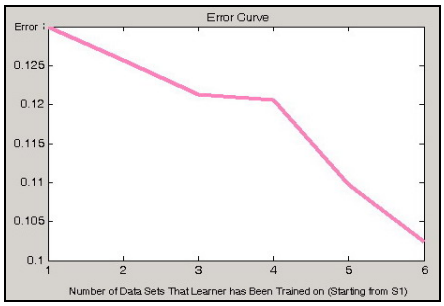
شکل ۱- توزیع داده های بکار رفته جهت بررسی عملکرد نسخه پیاده سازی شده الگوریتم Learn++

در ابتدای آزمایشات، تاثیر پارامترهای مختلف الگوریتم بر روی کارایی نسخه اصلی آن مورد بررسی قرار گرفت تا بدین طریق پارامترهای بهینه تعیین گردد. برای این منظور پارامترهای زیر هر یک با مقادیر متفاوت، که در پرانتز آمده اند، در الگوریتم بکار رفتند و خطای طبقه بندی الگوریتم در هر مرحله بدست آمده و ثبت شد: تعداد نورونهای لایه مخفی هر WeakLearner به ازاء هر نورون خروجی (۱، ۲، ۵، ۱۰)، تعداد Epoch برای هر WeakLearner (۵۰، ۱۰۰)، تعداد Weaklearner برای هر مجموعه داده جدید (۵، ۱۰)، تعداد داده آموزشی (۴۰، ۱۰۰)، تعداد داده آزمایشی (۲۰، ۵۰)، سطح آستانه خطا برای پذیرش هر WeakLearner(۰/۳، ۰/۵).

نتایج این آزمایشها نشان می دهد استفاده از WeakLearnerهایی با قدرت یادگیری بیشتر (تعداد نورونهای لایه مخفی و تعداد Epoch های بیشتر) تا حدودی موجب بهبود عملکرد کلی سیستم می گردد. هرچند استفاده از WeakLearnerهای بسیار بزرگ با آنکه فاز یادگیری را بسیار طولانی می کنند، موجب کاهش قابل توجهی درخطای نهایی سیستم نخواهد شد. از طرف دیگر استفاده از WeakLearnerهای بسیار کوچک نیز به علت آنکه به ندرت موجب دستیابی به حد آستانه خطای قابل قبول برای پذیرش WeakLearner می شوند، فاز یادگیری را طولانی می نماید. به علاوه استفاده از تعداد WeakLearnerهای بیشتر ومجموعه داده های آموزشی و آزمایشی بزرگتر برای آموزش هریک، در یادگیری یک مجموعه داده جدید نیز موجب بهبود عملکرد کلی سیستم می گردد.

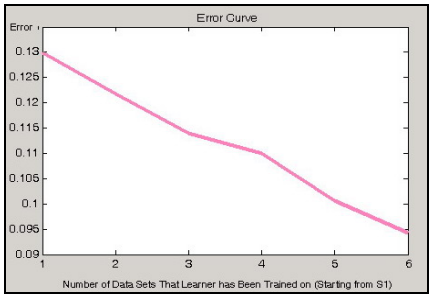
در اینجا نیز استفاده از تعدادWeakLearnerهای بسیار زیاد ومجموعه داده های آموزشی و آزمایشی بسیار بزرگ برای آموزش هریک، با آنکه موجب ارتقا چشمگیری در عملکرد سیستم یادگیری نمی شوند، فاز یادگیری را بسیار طولانی می کنند. مقادیر مناسب برای این پارامترها، مقادیری هستند که تعداد داده های یک مجموعه داده جدید را بپوشانند و فرصت تمرکز بیشتر بر روی نمونه های مشکل تر را نیز داشته باشند. انتخاب مقدار ۱۰ برای تعداد WeakLearnerها مناسب به نظر می رسد. اندازه مجموعه آموزشی مناسب نیز مقداری است که تعداد اعضای آن ضرب در تعداد WeakLearnerها حدود ۲ تا ۳ برابر تعداد اعضای مجموعه داده جدید باشد. اندازه مناسب برای مجموعه آزمایشی نیز حدود ۰/۳ تا نصف اندازه مجموعه داده آموزشی می باشد. با توجه به این موارد، می توان گفت قدرت یادگیری (اندازه) مناسب برای یک WeakLaner، باید با توجه به اندازه مجموعه آموزشی تعیین گردد. به گونه ای که هر WeakLearner نه آنقدر ضعیف باشد که به ندرت به آستانه خطای پذیرش دست یابد، و نه آنقدر قوی باشد که بیهوده فاز یادگیری را طولانی نماید.

بکارگیری سطوح متفاوتی از خطا برای پذیرش یک WeakLearnerنشان داد استفاده از سطح آستانه خطا برای پذیرش یک WeakLearner با مقادیر کمتر از ۰/۵ بدلیل آنکه موجب رد کردن بیشتر WeakLearnerها خواهد شد، فاز یادگیری را طولانی مینماید، هرچند موجب بهبود چشمگیری در عملکرد نهایی سیستم نخواهد شد. در آزمایشی دیگر یکبار از رابطه $\log(\frac{1}{\beta_t})$ و بار دیگر از رابطه $(1-\beta)$ برای محاسبه امتیاز هر کلاس استفاده شد. نتایج این آزمایش حاکی از برتری عملکرد الگوریتمی داشت که از رابطه دوم استفاده می کرد. بهترین نتایجی بدست آمده از نسخه اصلی الگوریتم مربوط به پارامترهایی با مقادیر زیر بودند: تعداد نورونهای لایه مخفی به ازاء هر نورون خروجی = ۵، تعداد Epochs = ۵۰، تعداد Weaklearner برای هر مجموعه داده جدید = ۱۰، تعداد داده آموزشی = ۱۰۰، تعداد داده آزمایشی = ۵۰، سطح آستانه خطا برای پذیرش: ۰/۵، استفاده از رابطه $(1-\beta)$ برای محاسبه امتیاز هر کلاس. شکل ۲ نمودار خطای نسخه اصلی الگوریتم را با مقادیر پارامترهایی که منجر به بهترین کارایی شدند نشان می دهد.



شکل۲. نمودار خطای الگوریتم Learn++ بر روی مجموعه داده آزمایشی مستقل

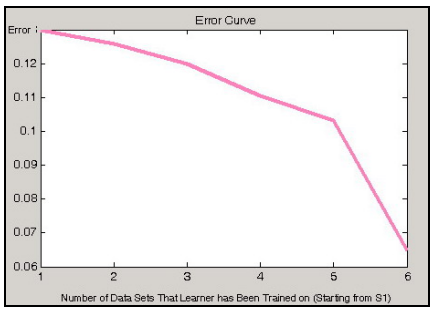
پس از این مرحله، و مقداردهی پارامترها با مقادیری که منجر به بهترین نتایج بر روی نسخه اصلی الگوریتم شده بود، الگوریتم با استفاده از WeakLearnerهایی با یک خروجی به کار گرفته شد. شکل ۳ نمودار خطای الگوریتم را در این حالت نشان می دهد.



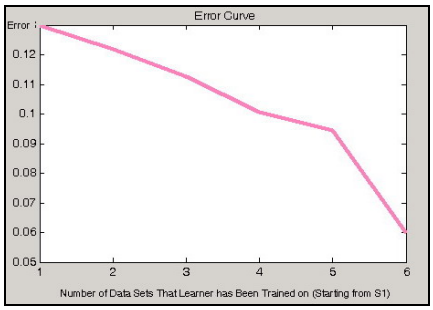
شکل ۳. نمودار خطای سیستم یادگیری افزایشی بر روی مجموعه داده آزمایشی مستقل، استفاده از WeakLearnerهایی با یک خروجی

در آزمایشی دیگر پس از مقدار دهی پارامترها با همان مقادیری که منجر به بهترین کارایی در نسخه اصلی الگوریتم شده بود، الگوریتم نسخه اصلی با اضافه کردن مرحله هنجار سازی امتیازات کلاسها اجرا شد. شکل ۴ نمودار خطای الگوریتم را در این حالت نشان می دهد.

در نهایت و در آزمایشی دیگر، ایده های استفاده از WeakLearnerهایی با یک خروجی و هنجارسازی امتیازات کلاسها با هم بکار گرفته شد. بهترین نتایج بدست آمده در آزمایشها متعلق به این حالت می باشد که در آن کمترین خطای طبقه بندی بدست آمد. شکل ۵ نمودار خطای الگوریتم را در این حالت نشان می دهد.



شکل ۴. نمودار خطای سیستم یادگیری افزایشی بر روی مجموعه داده آزمایشی مستقل، با استفاده از هنجارسازی امتیازات



شکل ۵. نمودار خطای نهایی الگوریتم ELearn++ بر روی مجموعه داده آزمایشی مستقل: کمترین خطای طبقه بندی در این حالت بدست آمده است

۵- بحث و نتیجه گیری

شکل نزولی نمودارهای خطا نشان دهنده موفقیت یادگیری افزایشی سیستم می باشد. در آزمایشهای صورت گرفته، تنها در مواردی که سیستم بر روی مجموعه آموزشی و در ابتدای فاز یادگیری به خطای صفر رسیده است، نمودار خطا بر روی این مجموعه آموزشی حالت صعودی (و البته با شیب کم)دارد، که این مساله نیز بدلیل وجود رای گیری اکثریت در سیتم طبیعی به نظر می رسد. مقایسه شکلهای ۲ و ۳ نشان می دهد استفاده از WeakLearnerهایی با یک خروجی موجب کاهش خطای نهایی سیستم یادگیری شده است. این کاهش خطا در حدود یک درصد بوده است. همچنین مقایسه شکلهای ۲ و ۴ نشان می دهد استفاده از هنجارسازی امتیازات برای رای گیری اکثریت وزندار موجب کاهش چشمگیری (حدود ۴ درصد) در خطای نهایی سیستم شده است.

با توجه به این موارد، به نظر می رسد تلاش برای بهبود عملکرد الگوریتم یاد گیری افزایشی Learn++ موفق بوده است. استفاده از رابطه $(1-\beta)$ برای محاسبه امتیازات به جای استفاده از رابطه $\log(\frac{1}{\beta_t})$، به تنهایی موجب کاهش اندکی در میزان خطا شده است. پس از آن استفاده از WeakLearnerهایی با یک خروجی، کاهش یک درصدی خطای نهایی سیستم یادگیری را موجب گردیده است. هنجار سازی امتیازات برای رای گیری اکثریت وزندار موجب کاهش قابل توجه ۴ درصدی در خطای نهایی سیستم شده است. در انتها، استفاده همزمان از تمامی این موارد (شکل ۵)، خطای نهایی الگوریتم یادگیری افزایشی ELearn++ را نسبت به Learn++ تقریبا نصف کرده است.

مراجع

[1] LiMin Fu, Hui-Huang Hsu, Jose C. Principle,"Incremental Backpropagation Learning Networks", IEEE Trans. On Neural Networks, Vol. 7, No.3, pp. 757-761, May 1996.

[2] Hown-Wen Chen, Von-Wun Soo, "Design of Adaptive and Incremental Feed-Forward Neural Networks", IEEE International Conference on Neural Networks, Vol. 1, pp. 479-484, 1993.

[3] R. Polikar, L. Upda, S. Upda, V. Honavar, "LEARN++: An Incremental Learning Algorithm for Multilayer Perceptron Neural Networks.", Proceedings of ICASSP 2000, vol. 6, pp. 3414-3417, 2000.

[4] R. Polikar, L. Upda, S. Upda, V. Honavar, "LEARN++: An Incremental Learning Algorithm for Supervised Neural Networks", IEEE Trans. On Systems, Man, and Cybernetics, Part c: Applications and Reviews, Vol. 31, No. 4, pp. 497-508, Nov. 2001.

[5] R. Polikar, J. Byorick, S. Krause, A. Marino, M. Moreton, "Learn++: a Classifier Independent Incremental Learning Algorithm for Supervised Neural Networks", IJCNN 2002, Proc. Of the 2002 Int. Joint Conference on Neural Networks, IEEE, Vol. 2, pp. 1742-1747, May 2002.

[6] R. Polikar, "Learn++: An Incremental Learning Algorithm Based On Psycho-physiological Model of Learning", Proc. Of the 23rd Annual EMBS Int. Conference, IEEE, Oct. 2001.

زیرنویس‌ها

[1] Enhanced Learn++
[2] Non stationary
[3] Incremental learning
[4] Compactness
[5] Plasticity
[6] Stability
[7] Noise
[8] Generalization
[9] Hypothesis
[10] Weighted Majority
[11] Over Fitting
[12] Instance Space