دانشکده مهندسی کامپیوتر

پایان‌نامه کارشناسی ارشد هوش مصنوعی

رده‌بندی توزیع‌شده با استفاده از روش بهینه‌سازی گرادیان نزولی تصادفی

نگارنده: بهناز سادات میرهادی تفرشی

استاد راهنما

دکتر هدی مشایخی

استاد مشاور

دکتر محسن بیگلری

دی ۱۳۹۸

تقدیم به پدر و مادر مهربانم

سپاس‌گزاری

به نام خدایی که هنرمندی‌اش بارز است و دیده از درکش عاجز، خدایی که زینت آدمی را علم قرار داد و اسباب رسیدنش را حلم، خدایی که زمین و زمان از اوست و ستایش خلقی بر درگه اوست، خدایی که اندیشه‌مان وامدار از کرامتش باشد و قلم فرسایی‌مان از سر عنایتش. پس سزد که نامش، همواره متصل به زبان باشد و یادش همیشه به نهر دل روان. و رواست: ترنم قلمم این زمزمه که خدایا دلی ده حقیقت‌شناس زبانی سزاوار حمد و سپاس

همچنین از استاد گرامی، سرکار خانم دکتر هدی مشایخی بسیار سپاسگزارم که در تمامی دشواری‌های این مسیر، راهنمایی‌هایشان چاره‌ساز کارم بود و نیز از آقای دکتر بیگلری بابت کمک‌های بی‌دریغشان قدردانی به عمل می‌آورم.

بهناز سادات میرهادی تفرشی
دی ۱۳۹۸

تعهد نامه

اینجانب **بهناز سادات میرهادی تفرشی** دانشجوی کارشناسی ارشد رشته **گروه کامپیوتر مهندسی کامپیوتر** دانشگاه شاهرود، نویسنده پایان‌نامه با عنوان **رده‌بندی توزیع‌شده با استفاده از روش بهینه‌سازی گرادیان نزولی تصادفی** ، تحت راهنمایی **هدی مشایخی** متعهد می‌شوم:

- تحقیقات در این پایان‌نامه توسط اینجانب انجام شده است و از صحت و اصالت برخوردار است.
- در استفاده از نتایج پژوهش‌های دیگر پژوهش‌گران، به مرجع مورد استفاده استناد شده است.
- مطالب این پایان‌نامه، تا کنون توسط خود، یا فرد دیگری برای دریافت هیچ نوع مدرک یا امتیازی در هیچ‌جا ارایه نشده است.
- حقوق معنوی این اثر، به دانشگاه صنعتی شاهرود تعلق دارد، و مقالات مستخرج با نام ‘‘ دانشگاه صنعتی شاهرود’’ یا “ Shahrood University of Technology ” به چاپ خواهد رسید.
- حقوق معنوی تمام افرادی که در به‌دست آوردن نتایج اصلی پایان‌نامه تاثیرگذار بوده‌اند، در مقالات مستخرج از پایان‌نامه رعایت می‌گردد.
- در تمام مراحل انجام این پایان‌نامه، در مواردی که از موجود زنده (یا بافت‌های آنها) استفاده شده است، ضوابط و اصول اخلاقی رعایت شده است.
- در تمام مراحل انجام این پایان‌نامه، در مواردی که به حوزه اطلاعات شخصی افراد دسترسی یافته (یا استفاده شده است)، اصل رازداری و اصول اخلاق انسانی رعایت شده است.

بهناز سادات میرهادی تفرشی
دی ۱۳۹۸

مالکیت نتایج و حق نشر

- تمام حقوق معنوی این اثر و محصولات آن (مقالات مستخرج، کتاب، برنامه‌های رایانه‌ای، نرم‌افزارها و تجهیزات ساخته شده) متعلق به دانشگاه صنعتی شاهرود می‌باشد. این مطلب باید به نحو مقتضی، در تولیدات علمی مربوطه ذکر شود.
- استفاده از اطلاعات و نتایج موجود در این پایان‌نامه بدون ذکر منبع مجاز نمی‌باشد.

چکیده

امروزه الگوریتم‌های رده‌بندی به جهت پردازش داده‌ها، با مجموعه داده‌های کلان روبرو هستند. از این رو، جهت رده‌بندی این داده‌ها که حجم عظیمی از محاسبات را دارند، انجام محاسبات با سرعت بالا اهمیت به سزایی پیدا می‌کند. الگوریتم‌های رده‌بندی به منظور بهینه‌سازی تابع هدف مسئله مورد نظر، از روش‌های بهینه‌سازی متعددی استفاده می‌کنند. در میان این روش‌ها، روش گرادیان نزولی تصادفی یکی از پرکاربردترین و محبوب‌ترین الگوریتم‌های بهینه‌سازی می‌باشد. این روش، یک تقریب از روش گرادیان نزولی است که با انتخاب تصادفی یک نمونه، گرادیان تابع هدف را به طور مکرر محاسبه می‌کند. اما به دلیل محاسبات اضافی و چالش در انتخاب نرخ یادگیری مناسب و همچنین نوسانات زیاد در هنگام رسیدن به نقطه کمینه تابع هدف، واریانس بالایی را ایجاد نموده و در نتیجه همگرایی ضعیفی را به همراه دارد. بنابراین به جهت آمد بهبود در الگوریتم، روشی با نام گرادیان نزولی تصادفی بازگشتی ارائه شده است. در این روش در هر دوره از الگوریتم با استفاده از یک فرمول بازگشتی واریانس ایجاد شده به جهت نمونه‌برداری تصادفی مورد محاسبه قرار گرفته و پارامتر مدل رده‌بندی بروزرسانی می‌گردد. بنابراین در حجم بالای داده‌ها، رده‌بندی با استفاده از این روش بهینه‌سازی دچار کاهش سرعت در پردازش مجموعه داده موردنظر می‌شود.
الگوریتم پیشنهادی به منظور افزایش سرعت در رده‌بندی داده‌ها با حجم بالا، روشی را جهت رده‌بندی توزیع شده با استفاده از گرادیان نزولی تصادفی مورد بررسی قرار داده است. در این روش، یک مسئله رده‌بندی رگرسیون لجستیک با استفاده از الگوریتم بهینه‌سازی گرادیان نزولی تصادفی، جهت افزایش سرعت در یک سیستم توزیع شده، پیاده‌سازی شده است. در الگوریتم مذکور، پارامترهای مسئله رده‌بندی با تعامل بین گره‌های کارگر و سرور بروزرسانی می‌شوند، به طوریکه داده‌ها بین گره‌های کارگر پخش شده و از اشغال شدن حجم شبکه جلوگیری می‌شود. گره‌های کارگر یک کپی از پارامتر سراسری مسئله رده‌بندی را با یک درخواست از سرور، دریافت کرده و عملیات بروزرسانی را روی این پارامترها انجام می‌دهند. در نهایت پارامترهای بروزرسانی شده به سمت سرور فرستاده می‌شوند. گره سرور این پارامترها را دریافت کرده و عمل تجمیع را روی آن‌ها انجام می‌دهد و به عنوان پارامتر سراسری ذخیره می‌کند. در روش پیاده‌سازی شده محاسبه کاهش واریانس به صورت بازگشتی و نمونه برداری تصادفی به جهت همگرایی خطی تابع هدف در سیستم توزیع شده، موجب بهبود دقت الگوریتم نیز می‌شود. در این پژوهش با استفاده از چهار مجموعه داده حجیم، الگوریتم پیشنهادی به صورت توزیع شده در محیط اسپارک پیاده‌سازی شده است. نتایج بدست آمده از مقایسه روش پیشنهادی با حالت متمرکز نمایانگر آن است که در چهار داده آموزشی با افزایش تعداد گره‌های کارگر در هر آزمایش انجام شده، تقریبا با حفظ نرخ همگرایی، سرعت اجرای الگوریتم پیشنهادی حداقل دو برابر نسبت به حالت متمرکز افزایش یافته است.

کلمات کلیدی: رده بندی، رگرسیون لجستیک، گرادیان نزولی، گرادیان نزولی تصادفی، کاهش واریانس، سیستم توزیع شده، کلان داده، اسپارک

فهرست مطالب

فهرست تصاویر

فهرست جداول

فصل ۱

مقدمه

یکی از مهمترین بخش‌های هر الگوریتم رده‌بندی[۱] در یادگیری ماشین، بهینه سازی[۲] است. گرادیان نزولی[۳] یکی از قدرتمندترین الگوریتم‌های بهینه سازی است که در بسیاری از مسائل رده‌بندی مورد استفاده قرار می‌گیرد. این روش هم در الگوریتم‌های رده‌بندی و هم در الگوریتم‌های یادگیری عمیق به دفعات مورد استفاده قرار گرفته است[۱]. به همین دلیل شناخت آن گام مهمی در فهمیدن منطق بسیاری از روش‌ها و الگوریتم‌های رده‌بندی در مبحث یادگیری ماشین می‌باشد. امروزه الگوریتم‌های رده‌بندی اغلب پیچیده هستند و پارامترهای فراوانی دارند. بدین جهت، افزایش سرعت یادگیری در مجموعه داده‌های آموزشی با حجم بالا[۴]، اهمیت ویژه‌ای خواهد داشت[۲].

با توجه به اینکه الگوریتم‌های بهینه‌سازی پس از محاسبات با تکرار زیاد، جهت بروزرسانی پارامترها همگرا می‌گردند، با درنظر گرفتن پیچیدگی مدل، حجم بالا در داده‌های آموزشی و همچنین نیاز به محاسبات فراوان، ارائه یک روش رده‌بندی به منظور اجرای الگوریتم با سرعت بالا، بسیار ضرروری است[۳، ۲].

---

[1] Classification
[2] Optimization
[3] Gradient Descent
[4] Big Data

در روش‌های رده‌بندی، پردازش در داده‌های حجیم با زمان اجرای بالایی صورت می‌پذیرد[۴]. در این پژوهش سعی شده است تا برای نخستین بار یک رده بندی توزیع‌شده با استفاده از بهینه سازی گرادیان نزولی تصادفی بازگشتی، جهت پردازش داده‌هایی با حجم بالا به صورت کارا و موثر، ارائه شود.

در این پژوهش از ابزار آپاچی اسپارک[1] جهت افزایش سرعت برای رده‌بندی داده‌هایی با حجم بالا استفاده شده است. همچنین می‌توان گفت برخلاف روش نگاشت_کاهش[2] که از دیسک به عنوان مکان ذخیره‌سازی داده‌های میانی استفاده می‌کند، اسپارک برنامه‌ها را به صورت یک مجموعه توزیع‌شده ارائه می‌دهد و جهت پردازش داده‌ها از یک حافظه مشترک استفاده می‌کند[۵، ۶].

۱.۱ تعاریف اولیه

در این قسمت به شرح کوتاهی از تعاریف اولیه و الگوریتم‌های کاربردی که اساس الگوریتم پیشنهادی را تشکیل می‌دهند، پرداخته شده است. همچنین نمای مختصری از چگونگی عملکرد این الگوریتم‌ها بیان می‌شود. در نهایت به نحوه عملکرد سیستم‌های توزیع‌شده در رده بندی و تاثیر معماری این سیستم‌ها در الگوریتم‌های بهینه‌سازی پرداخته می‌شود.

۱.۱.۱ رده بندی

رده‌بندی یکی از زیر شاخه‌های اساسی یادگیری ماشین و داده کاوی است. در علم داده کاوی، رده بندی همان فرآیند تحلیل داده‌ها است که طی آن مدل توصیف کننده داده و کلاس های تمییز دهنده آن مشخص می‌شوند. درحقیقت رده‌بندی، نسبت دادن یک داده جدید مشاهده شده به کلاس‌های اصلی تشکیل دهنده مجموعه داده است که از طریق اجرای الگوریتم های یادگیری بر روی داده‌های آموزشی موجود صورت می گیرد. اساس آن داده‌های جمع‌آوری شده از اعمال گذشته هستند. اعمالی که بر اساس دانش فرد خبره برچسب گذاری شدند. در اصطلاحات یادگیری ماشین، رده بندی نمونه ای از یادگیری نظارت شده می‌باشد. یعنی یادگیری در جایی که مجموعه ای از مشاهدات به درستی شناسایی شده در نظر گرفته شود. الگوریتمی که رده‌بندی را پیاده سازی می‌کند، به عنوان رده‌بندی کننده شناخته می‌شود. اصطلاح رده‌بندی گاه به عملکرد ریاضیاتی که توسط یک الگوریتم رده‌بندی انجام می‌شود،

[1] Apache Spark
[2] Map-Reduce

اشاره دارد که داده‌های ورودی را به یک دسته رده‌بندی می‌کند[۷].

۲.۱.۱ گرادیان نزولی

روش گرادیان نزولی اولین بار در سال ۱۸۴۷ توسط کاشی ارائه شد [۹]. الگوریتم گرادیان نزولی یک الگوریتم بهینه سازی تکرار شونده مرتبه اول برای یافتن کمترین مقدار یک تابع است [۸]. برای پیدا کردن کمترین[۱] مقدار محلی از یک تابع با استفاده از گرادیان نزولی، گام‌هایی متناسب در جهت منفی شیب (یا شیب تقریبی) از نقطه فعلی در نمودار برداشته می‌شود. در عوض، اگر قدم‌هایی متناسب با جهت مثبت شیب برداشته شود، حداکثر[۲] مقدار محلی آن تابع بدست می‌آید، که با عنوان گرادیان صعودی[۳] شناخته می‌شود[۹].

۳.۱.۱ گرادیان نزولی تصادفی

الگوریتم گرادیان نزولی تصادفی[۴] روشی مبتنی بر تکرار برای بهینه‌سازی یک تابع مشتق‌پذیر می‌باشد. این تابع مشتق‌پذیر را تابع هزینه یا تابع هدف می‌نامند. این الگوریتم یک تقریب تصادفی از روش گرادیان نزولی می‌باشد. در حقیقت گرادیان نزولی تصادفی الگوریتمی در اختیار ما قرار می‌دهد که طی چند حلقه تکرار مقدار کمینه یک تابع و مقادیری را که به ازای آن‌ها تابع کمینه مقدار خود را می‌گیرد، بدست بیاوریم. به تازگی مقاله‌ای [۱۰]، ابداع این روش را به هربرت رابینز و ساتِن مونرو برای انتشار مقاله‌ای در باب گرادیان نزولی تصادفی در سال ۱۹۵۱ نسبت داده است. تفاوت گرادیان نزولی تصادفی با گرادیان نزولی استاندارد در این است که برخلاف گرادیان نزولی استاندارد که جهت بهینه‌سازی تابع هدف از تمام داده‌های آموزشی استفاده می‌کند، گرادیان نزولی تصادفی از گروهی از داده‌های آموزشی که به‌طور تصادفی انتخاب می‌شود برای بهینه‌سازی استفاده می‌کند. این روش در مسائل یادگیری ماشین از جمله مسئله رده‌بندی کاربرد فراوانی دارد[۱۰].

۴.۱.۱ سیستم توزیع‌شده

[1] Minimum
[2] Maximum
[3] Gradient Ascent
[4] Stochastic Gradient Descent

یك سیستم توزیع‌شده[۱] مجموعه‌ای از سیستم‌های مستقل است که از دید کاربر به صورت سیستمی یکتا[۲]و منسجم[۳] به نظر می‌آید. سیستم توزیعی به صورت میان افزار[۴] سازمان دهی شده است. لایه میان افزار روی چندین ماشین گسترده شده و برای تمامی برنامه های کاربردی واسط یکسانی را ارائه می‌دهد[۱۱].

در این نوع سیستم‌ها نیاز است چهار هدف مهم برآورده شود تا سیستم توزیع‌شده ارزش ساخته شدن را داشته باشد. اولین هدف قابلیت دسترسی[۵] آسان به منابع است. هدف اصلی سیستم توزیع‌شده این است که دسترسی آسانی برای کاربران به منابع دور، به روشی کنترل شده و کارآمد را فراهم نماید. دومین هدف شفافیت[۶] در سیستم توزیعی می‌باشد. در واقع یك سیستم توزیعی که از دید کاربران و نرم افزارهای کاربردی خود، بصورت سیستم کامپیوتری منفرد جلوه کند، اصطلاحاً شفاف نامیده می شود. سومین هدف باز بودن[۷] سیستم توزیعی است. سیستم توزیع‌شده باز، سیستمی است که سرویس‌ها را مطابق قواعد استاندارد توصیف کننده نحو[۸] و معنا[۹] ارائه می‌دهد. در نهایت، آخرین هدف توسعه پذیری[۱۰] سیستم از سه جنبه اندازه[۱۱]، جغرافیا[۱۲] و راهبری[۱۳] می‌باشد [۱۳][۱۲] .

[1] Distributed Systems
[2] Single
[3] Coherent
[4] Middleware
[5] Accessible
[6] Transparency
[7] Open
[8] syntax
[9] semantic
[10] Scable
[11] size
[12] geographically
[13] administratively

۵.۱.۱ محاسبات توزیع‌شده

محاسبات توزیع‌شده یک فناوری بسیار گسترده است. بیش از سه دهه است که از این فناوری در پردازش داده‌ها[1] استفاده می‌شود. به بیان ساده، محاسبات توزیع‌شده، به محاسبه از طریق رایانه‌های مستقل توزیع‌شده می‌پردازند که تنها از طریق شبکه ارتباط برقرار می‌کنند [۱۴].

سیستم‌های محاسباتی توزیع‌شده معمولاً متفاوت از سیستم‌های محاسبات موازی[2] یا سیستم‌های حافظه مشترک[3] (چندین رایانه یک حافظه مشترک دارند که برای ارتباط بین پردازنده‌ها استفاده می‌شوند) می‌باشند [۱۵]. سیستم‌های حافظه توزیع‌شده از چندین کامپیوتر با محاسبه توزیع‌شده بین رایانه‌های متصل (گره‌ها)[4]، برای حل یک مشکل مشترک، استفاده می‌کنند [۱۶].

۲.۱ بیان مسئله

امروزه در الگوریتم‌های رده‌بندی به دلیل دسترسی به حجم زیاد داده‌ها، پردازش در این الگوریتم‌ها با زمان محاسباتی زیاد مواجه می‌شود [۱۷]. گرادیان نزولی یک الگوریتم بهینه‌سازی تکرارپذیر است که به کمک آن می‌توان مقدار کمینه یک تابع مورد نظر را محاسبه کرد. در اینجا تابع مورد نظر، همان تابع هدف است[۱۸].

همان‌طور که در تعریف اولیه گرادیان نزولی مطرح شد، روش گرادیان نزولی راهی برای یافتن کمینه یک تابع هدف است که بر اساس بردار پارامترهای مدل نوشته می‌شود. به این صورت که پارامترها در خلاف جهت گرادیان تابع نسبت به پارامترهای مدل تغییر می‌کنند. نرخ یادگیری اندازه‌ی گام[5] تغییر پارامترها برای یافتن کمترین مقدار را تعیین می‌کند. به عبارت دیگر، در جهت شیب سطح ایجاد شده توسط تابع به سمت پایین حرکت کرده تا به یک دره برسد. به این ترتیب در هر گام، موقعیت بعدی را براساس موقعیت فعلی تنظیم می‌کند تا الگوریتم از مسیر خارج نشود. به این شیوه مشخص است که گام بعدی به گام قبلی بستگی دارد. در این حالت کاهش طول گام‌ها با کاهش شیب هماهنگ می‌شود. این فرآیند تا

[1] Data Processing
[2] Parallel
[3] Shared Memory
[4] Nodes
[5] Steps

زمانی که دیگر شیبی وجود نداشت باشد، ادامه خواهد داشت. در این هنگام به دره که هدف الگوریتم بود، دست یافته می‌شود. بنابراین در الگوریتم گرادیان نزولی این گام ها با تعیین نرخ یادگیری در الگوریتم برداشت می‌شوند [۲۱][۱۹].

انتخاب نرخ یادگیری مناسب می‌تواند دشوار باشد. یک نرخ یادگیری ثابت برای همه‌ی تغییرات ایجاد شده در پارامترها استفاده می‌شود. اگر داده‌ها خیلی پراکنده باشند و ویژگی‌های مورد بررسی تعداد رخدادهای متفاوتی داشته باشند، نباید برای همه‌ی آنها به یک اندازه تغییر ایجاد شود. نرخ یادگیری بسیار اندک منجر به همگرایی[۱] بسیار کند شده، در حالی که میزان نرخ یادگیری خیلی زیاد می‌تواند مانع همگرایی شود و یا حداقل عملکرد الگوریتم را با ضرر مواجه کند و حتی منجر به واگرایی شود. در نتیجه می‌توان گفت، نرخ یادگیری کوچک زمان بیشتری را برای پردازش داده‌های آموزشی صرف کرده، همچنین انتخاب نرخ یادگیری بزرگ در یک مدل، دقت را برای هر داده آزمون تضمین نمی‌کند. جهت تعیین نرخ یادگیری مناسب در طول آموزش میزان نرخ یادگیری را تنظیم می‌کنند. در واقع این امر موجب مقاوم کردن الگوریتم می‌شود[۱۸]. مقاوم کردن[۲] یعنی کاهش میزان یادگیری مطابق با یک میزان از پیش تعریف شده یا وقتی تغییر در هدف بین دوره‌ها کمتر از یک آستانه است. با این حال، این مقادیر و آستانه‌ها باید از قبل مشخص شوند و از این رو قادر به سازگاری با مشخصات یک مجموعه داده نیستند[۲۰]. علاوه بر این، نرخ یادگیری یکسان برای همه بروزرسانی‌های پارامتر مدل اعمال می‌شود[۲۱].

یکی دیگر از چالش های مهم، به حداقل رساندن خطای عملکرد الگوریتم در حالت غیر محدب[۳] می‌باشد. برای الگوریتم رده‌بندی در مدل یادگیری ماشین، جلوگیری از گرفتار شدن در حداقل‌های محلی[۴]اهمیت ویژه‌ای دارد[۱۰].

دافین و همکاران، استدلال می‌کنند که مشکل در حقیقت نه از کمینه محلی بلکه از نقاط زین[۵]، یعنی نقاطی که شیب هم با جهت مثبت و هم با جهت منفی وجود دارد، می‌باشد. در واقع نقاط زینی نقاط بحرانی هستند که مقدار تابع در آن‌ها صفر است، اما این نقاط بیشینه و یا کمینه تابع نمی‌باشند. این نقاط زین معمولاً توسط یک صفحه با همان خط احاطه می شوند، که اجتناب از ارائه گرادیان نزولی تصادفی را بسیار سخت می‌کند، زیرا شیب در همه ابعاد نزدیک به صفر است[۱۶].

[1] Convergence
[2] Annealing
[3] non-convex error
[4] Minimum Local
[5] Saddle Point

از طرفی در گرادیان نزولی تصادفی در هنگام انتخاب نمونه‌ها به صورت تصادفی، ارتعاشات[1] زیادی در مسیر رسیدن به کمینه تابع بوجود می‌آورد، این مسئله موجب ایجاد واریانس بالا در محاسبه گرادیان تابع مورد نظر می‌شود. این امر باعث کاهش همگرایی در الگوریتم خواهد شد[۲۰].

علاوه براین عدم پشتیبانی الگوریتم بهینه سازی گرادیان نزولی تصادفی از سیستم‌های توزیع‌شده موجب طرح الگوریتم پیشنهادی این پژوهش و بحث و بررسی بر روی آن گردید[۱۴]. محاسبات توزیع‌شده برای آموزش در الگوریتم‌های رده‌بندی در مجموعه داده های بزرگ ضروری است[۸]. در این پژوهش بر موازی سازی داده‌ها تمرکز شده است. گره‌ها به صورت یک مدل مشترک مدل، داده‌های آموزش را در قسمت های مختلف از طریق روش ارائه شده بروزرسانی می‌کنند.

بنابراین در راستای حل مشکلاتی مانند محاسبات اضافی در داده‌های حجیم، همچنین مشکلات واریانس بالا در انجام محاسبات در هر دوره برای داده‌های آموزشی در گرادیان نزولی تصادفی، در رده بندی با استفاده از الگوریتم گرادیان نزولی روشی پیشنهاد می‌شود. در الگوریتم پیشنهادی مشکلات مطرح شده را در صورت امکان برطرف کرده تا کارایی بالایی در الگوریتم را حاصل نماید.

۳.۱ اهمیت مسئله

امروزه پردازش داده‌های حجیم در الگوریتم‌های رده‌بندی، موجب محاسباتی در سطح بالاتر می‌شوند. به طور مثال می‌توان گفت امروزه در شبکه‌های عمیق بالغ بر میلیون‌ها پارامتر در هر لایه شبکه با مقادیر زیادی داده آموزش داده می‌شوند. همچنین در یک مسئله رده‌بندی با داده‌های کلان این نکته حائز اهمیت است که الگوریتم‌های مورد استفاده در رده‌بندی، برپایه تکرار به جهت بروزرسانی پارامترها بنا شده‌اند. بنابراین در این الگوریتم‌ها دورهای زیادی از محاسبات تکراری برای بروز رسانی پارامترهای مدل مورد نیاز می‌باشند[۱۹].

از این رو با در نظر گرفتن پیچیدگی به وجود آمده در این الگوریتم‌های رده‌بندی به سبب حجم بالای داده، ارائه روش موثر برای آموزش مقیاس بزرگی از داده‌ها را می‌توان از وظایف یادگیری ماشین دانست. بنابراین با توجه به روش ارائه شده با عنوان رده‌بندی توزیع‌شده با استفاده از بهینه سازی گرادیان نزولی تصادفی سعی بر کاهش زمان اجرای الگوریتم و بهبود

[1] Fluctuation

آن شده است. الگوریتم ارائه شده ترکیبی از پروتکل ارتباطی آسنکرون[1] و کاهش واریانس می‌باشد. در این روش پیشنهادی، الگوریتمی ارائه شده است که در آن به جهت بهینه‌سازی، از یک فاکتور نرخ یادگیری مناسب استفاده شده است.

۴.۱ هدف مسئله

هدف اصلی از انجام این پژوهش طراحی یک سیستم جهت رده بندی توزیع‌شده است که با بهینه‌سازی گرادیان نزولی تصادفی انجام می‌پذیرد. چنان که از توضیحات پیشین استنباط می‌شود در گرادیان نزولی تصادفی محاسبات دچار نوسانات زیادی می‌شوند. در واقع این نوسانات شدید نشان‌دهنده واریانس بالا در اجرای الگوریتم می‌باشد. که این پژوهش قصد کاهش این نوسانات را دارد. همچنین باید این نکته را مطرح کرد که کاهش واریانس موجب کاهش همگرایی در الگوریتم پیشنهادی نشود.

بنابراین تکنیک مورد استفاده در روش ارائه شده محاسبات را با استفاده از سیستم توزیعی، به گونه‌ای انجام می‌دهد که از کاهش همگرایی در الگوریتم بهینه‌سازی و افزایش زمان اجرای الگوریتم جلوگیری شود. همان‌گونه که مطرح شد در سیستم‌های توزیع‌شده بروزرسانی پارامترها بر روی گره‌ها صورت می‌پذیرند. بنابراین این تقسیم محاسبات موجب کاهش همگرایی نمی‌شود. در اینجا این نکته حائز اهمیت است که تکنیک مورد استفاده برای کاهش واریانس بر روی یک خوشه انجام می‌پذیرد و نه بر روی گره‌ها، زیرا در غیر این صورت همگرایی در الگوریتم حاصل نمی‌شود. بنابراین با ارائه این الگوریتم سعی بر نگاه داشتن نرخ همگرایی به صورت خطی شده است. همچنین بالا بردن سرعت در محاسبات از اهمیت بالایی برخوردار می‌باشد. از طرفی کاهش زمان اجرای الگوریتم فراهم می‌شود، که این امر به خودی خود در پردازش داده‌ها در حجم بالا موجب سهولت در پردازش داده‌های مورد نظر می‌گردد.

۵.۱ روش پیشنهادی

همان‌طور که در مطالب گذشته بیان شد، امروزه در مسائل رده‌بندی به جهت افزایش حجم داده‌های پردازشی، محاسبات پیچیده و با حجم بالایی بوجود می‌آید. از این رو به منظور سرعت بخشیدن به انجام محاسبات مذکور، به سیستمی جهت افزایش سرعت محاسبات

[1] Asynchronous Serial Interface Protocol

نیازمند هستیم.

در الگوریتم رده‌بندی برای بهبود عملکرد تابع هدف مسئله مورد نظر از روش های بهینه سازی استفاده می‌شود. می‌توان از مهمترین این روش‌ها گرادیان نزولی را نام برد. این روش مهمترین و پرکاربردترین روش در میان الگوریتم‌های بهینه‌سازی است. با این وجود این روش مشکلاتی مانند محاسبات اضافی، سرعت پایین در انجام محاسبات و در نهایت نرخ همگرایی کند را به همراه دارد. بنابراین به منظور کاهش میزان محاسبات الگوریتم گرادیان نزولی تصادفی ، روش دیگری ارائه شده است. در این روش با تغییر اندک قبل از محاسبه گرادیان در هر دوره، محاسبات اضافی از بین می‌روند. اما به دلیل افزایش نوسانات در هنگام انتخاب نرخ یادگیری در این الگوریتم، واریانس بالایی به‌وجود می‌آید.

برای حل مشکل واریانس بالا در الگوریتم، از روشی با نام گرادیان نزولی تصادفی همراه با کاهش واریانس استفاده می‌شود. در این الگوریتم گرادیان بهینه برای هر دوره را از گرادیان کل در آن دوره کم می‌کند. بنابراین واریانس حاصل از محاسبات در روش گرادیان نزولی تصادفی از بین می‌رود. اما این روش در حالت‌هایی که مسئله رده‌بندی تحت تحدب قوی باشد عملکرد خوبی نخواهد داشت. از این رو الگوریتم گرادیان نزولی تصادفی بازگشتی ارائه شده است. با توجه به مفروضات تعیین شده در این روش که در فصل سوم به تفصیل به آن پرداخته می‌شود، با حفظ همگرایی خطی مشکل عملکرد نامطلوب در روش گرادیان نزولی تصادفی همراه با کاهش واریانس حل می‌شود. علاوه بر این سرعت همگرایی را حفظ می‌کند. اما در این الگوریتم به دلیل ماهیت آن، که به گونه‌ای طراحی شده است که تنها روی یک گره کارایی دارد، از یک سیستم توزیع‌شده پشتیبانی نمی‌کند.

بنابراین در این پژوهش به منظور بهینه سازی یک تابع هدف در مسئله رده بندی از روش گرادیان نزولی تصادفی بازگشتی،استفاده شده است. با توجه به طراحی این الگوریتم که مشکلاتی از قبیل افزایش واریانس و یا پایین آمدن نرخ همگرایی تحت شرایط خاص، مانند تحدب قوی، را حل کرده، استفاده شده است. همچنین برای نخستین بار این الگوریتم در یک سیستم توزیع‌شده، جهت افزایش سرعت پردازش داده‌ها در حجم بالا، ارائه شده است. روش پیاده‌سازی شده به جهت کاهش واریانس به صورت بازگشتی و نمونه برداری غیر یکنواخت در یک سیستم توزیع‌شده، موجب بهبود دقت الگوریتم نیز می‌شود.

۶.۱ روش ارزیابی

در الگوریتم ارائه شده، از یک معادله پایه رگرسیون لجستیک[1] به عنوان مسئله رده‌بندی استفاده شده است. در واقع تابع هدف مورد نظر در این روش را یک تابع رگرسیون قرار داده‌ایم. بنابراین با استفاده از یک روش بازگشتی همراه با کاهش واریانس، گرادیان برای این تابع محاسبه می‌شود.

همچنین این محاسبات به جهت تسریع در انجام عملیات محاسبه گرادیان در هر دوره و کارایی بیشتر الگوریتم، بر روی یک سیستم توزیع‌شده، پیاده‌سازی شده‌است. روش پیشنهادی، با استفاده از ابزار اسپارک و با چهار نوع مجموعه داده مختلف پیاده سازی شده است. نتایج حاصل به تفضیل در فصل پیاده‌سازی و ارزیابی روش پیشنهادی ارائه می‌شود.

۷.۱ مروری بر فصول

در این پایان نامه پس از بیان مقدماتی جهت آشنایی خوانندگان با مفاهیم اولیه پژوهش، در فصل دوم به بررسی دقیق مفاهیم و همچنین شرح پژوهش‌های پیشین که در راستای کارایی بهتر الگوریتم گرادیان نزولی تصادفی، رده بندی در سیستم‌های توزیع‌شده و کارایی بالاتر در الگوریتم گرادیان نزولی بر پایه توزیع شدگی می‌پردازیم. سپس در فصل سوم الگوریتم پیشنهادی را به عنوان رده‌بندی توزیع‌شده با استفاده از بهینه‌سازی گرادیان نزولی تصادفی ارائه می‌شود. در فصل چهارم آزمایشات و نتایج بدست آمده و میزان بهبود الگوریتم گزارش خواهد شد. در نهایت در فصل پنجم نتیجه‌گیری نهایی از انجام پژوهش و پیشنهادات آینده ارائه می‌شود.

[1] Logistic Regression

فصل ۲

ادبیات پژوهش و کارهای پیشین

۱.۲ مقدمه

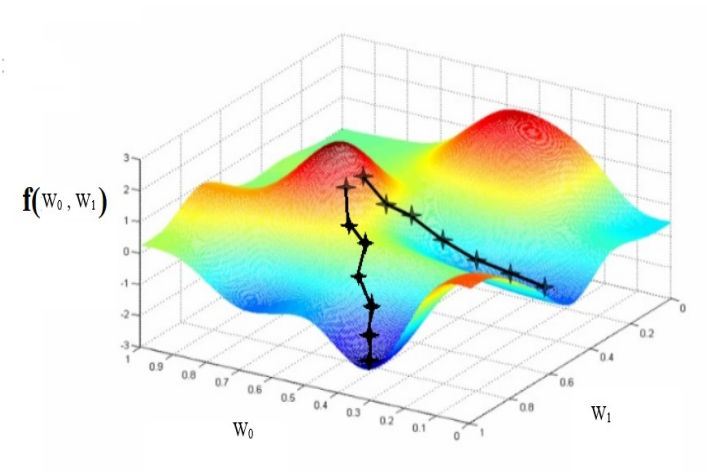
در این فصل ابتدا مفاهیم پایه و اولیه ضروری در روش بهینه‌سازی گرادیان نزولی تصادفی در رده‌بندی بیان می‌شود و سپس به تشریح سیستم‌های توزیع‌شده و کاربردهای آن‌ها در رده‌بندی پرداخته شده تا پس از آشنایی با ادبیات اولیه پژوهش، به بررسی پژوهش‌های پیشین در راستای الگوریتم پیشنهادی پرداخته و چالش‌های پیش رو در آنها را مورد بحث و بررسی قرار می‌دهیم.

۲.۲ گرادیان نزولی

استفاده از گرادیان نزولی[1] یکی از پرکاربردترین راهکارها، جهت بهینه سازی پارامترها برای کاهش خطای رده‌بندی در روش‌های مبتنی بر جداساز است. همان‌طور که در شکل ۱.۲ مشاهده می‌شود، الگوریتم گرادیان نزولی یک فرایند تکرار شونده است که کمترین مقدار تابع را بدست می‌آورد. این الگوریتم کمک می‌کند تا با استفاده از مشتقات تابع، تصمیماتی

[1] Gradient Descent(GD)

کاربردی و موثر اتخاذ شود. مشتق یک تابع، به عنوان شیب نمودار در یک نقطه خاص محاسبه می‌شود. شیب با کشیدن یک خط مماس به نمودار در یک نقطه توصیف می‌شود. بنابراین، اگر بتوان این خط مماس را محاسبه کرد، ممکن است بتوان جهت مورد نظر برای رسیدن به حداقل‌های یک تابع را بدست آورد. بنابراین از این شیب نزولی به عنوان استراتژی بهینه سازی استفاده می‌شود.[۱۸]



شکل ۱.۲: گرادیان نزولی[۱۸]

در این روش انبوه محاسبات اضافی و غیرضروری برای مجموعه داده‌های بزرگ انجام می‌پذیرد، زیرا در این الگوریتم گرادیان داده‌های یکسانی را به طور مکرر قبل از ایجاد هر تغییر محاسبه می‌کند. اگر این روش راهی برای یافتن مینیمم تابعی مانند $f(w)$ در نظر گرفته شود، این تابع براساس پارامترهای مدل یعنی $w \in \mathbb{R}$ نوشته شده است. به این صورت که پارامترها در خلاف جهت تابع تغییر می‌کنند و با توجه به فرمول (۱.۲) بروزرسانی می‌شوند. در این فرمول $f(w)$ را تابع هدر یا هزینه می‌نامند.

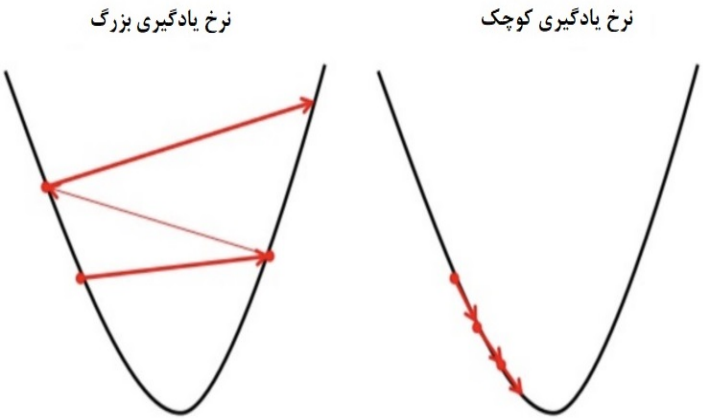
$$f(w) := \frac{1}{n}\sum_{i=1}^{n} f_i(w) \tag{۱.۲}$$

سپس با استفاده از فرمول (۲.۲) پارامتر مدل را بروزرسانی می‌کنیم. روش گرادیان نزولی ساده، که به نام گرادیان نزولی انبوه هم شناخته می‌شود، گرادیان تابع هدر نسبت به بردار پارامتر w را برای کل مجموعه داده‌های ورودی سیستم محاسبه می‌کند.

$$w_1 = w_0 - \eta * f_i(w) \tag{۲.۲}$$

۱.۲.۲ نرخ یادگیری

در فرمول (۲.۲)، η را نرخ یادگیری [۱] می‌نامیم. گرادیان یک تابع دارای ارزش بردار است و به عنوان یک بردار، دارای جهت و اندازه می‌باشد. الگوریتم گرادیان نزولی شیب را در یک عدد (میزان یادگیری یا اندازه مرحله) ضرب می‌کند تا نقطه بعدی را تعیین کند. انتخاب نرخ یادگیری مناسب می‌تواند دشوار باشد. نرخ یادگیری بسیار اندک منجر به همگرایی با سرعت بسیار پایین می‌شود، در حالی که یک میزان یادگیری خیلی زیاد می تواند مانع همگرایی شود و باعث شود عملکرد تابع ضرر ایجاد کرده و حتی واگرایی [۲] داشته باشد. بنابراین اگر نرخ یادگیری کوچک انتخاب شود، زمان همگرایی الگوریتم افزایش می‌یابد و اگر نرخ یادگیری بزرگ انتخاب شود، ممکن است مانند شکل ۲.۲ الگوریتم در حلقه نامتناهی بیافتد و هرگز همگرا نشود. [۱۸]



شکل ۲.۲: میزان نرخ یادگیری [۲۲]

۲.۲.۲ تابع هدر / تابع هزینه

در مسائل بهینه سازی در رده‌بندی و همچنین در نظریه تصمیم، تابع هدر [۳] / تابع هزینه[۴] تابعی است که یک رویداد یا مقادیر یک یا چند متغیر را بر روی یک مقدار واقعی نگاشت می کند که بطور شهودی نمایانگر مقداری از ’’هزینه’’ مرتبط با آن رویداد است. در واقع یک مسئله بهینه سازی در پی به حداقل رساندن تابع هدر است[۲۳].

[1] Learning Rate
[2] Divergence
[3] Loss Function
[4] Cost Function

تابع هدر یا تابع هزینه کارایی الگوریتم رده‌بندی را ارزیابی می‌کند. تابع هدر، میزان خطا را برای یک داده آموزشی محاسبه کرده در صورتی که تابع هزینه میزان میانگین تمام توابع هدر برای کل داده‌های آموزشی را محاسبه می‌کند. در واقع تابع هزینه میزان خوب بودن الگوریتم رده‌بندی را مشخص می‌کند. بنابراین هدف در الگوریتم رده‌بندی بدست آوردن کمترین مقدار تابع هزینه است. به این دلیل که خطای کمتری بین مقدار واقعی و مقدار پیش بینی شده را به ما نشان می‌دهد و این به معنی کارایی خوب الگوریتم رده‌بندی در مدل یادگیری می‌باشد[۲۴].

۳.۲ گرادیان نزولی تصادفی

در میان روش‌های بهینه سازی روش گرادیان نزولی بسیار کارآمد است. الگوریتم گرادیان نزولی تصادفی به جهت انتخاب تصادفی نمونه‌ها برای به‌روزرسانی پارامترهای مدل نام‌گذاری شده‌است. در این الگوریتم تخمین وزن مدل در تکرارهای بسیار، تابع هدف مسئله را در هر مرحله به حداقل میزان ممکن می‌رساند.

در الگوریتم گرادیان نزولی تصادفی، به جای کل مجموعه داده‌های مربوط به هر تکرار، چند نمونه به طور تصادفی انتخاب می‌شوند. روش گرادیان نزولی می‌تواند بسیار کند باشد. در ضمن این روش به ما اجازه نمی‌دهد که مدل را به طور لحظه‌ای تغییر دهیم. این بدان معناست که نمی‌توان داده‌های جدید را حین کار به آن اضافه کرد[۱۸].

امروزه الگوریتم‌های رده‌بندی جهت پردازش داده‌ها با حجم بالایی از داده مواجه می‌شود. از این رو جهت دسته‌بندی بهتر داده‌ها از الگوریتم گرادیان نزولی تصادفی [1]استفاده می‌شود. با توجه به این موضوع که استفاده از این الگوریتم همگرایی خوبی را فراهم می‌کند، اما در بسیاری از موارد در مواجهه با داده‌ها با حجم بالا به کندی عمل می‌کند. بنابراین اجرای غیر همزمان از الگوریتم گرادیان نزولی تصادفی برای داده‌ها با حجم بالا، سرعت الگوریتم را افزایش می‌دهد. اما باید بدانیم که در این حالت ارتباط غیر بهینه بین اجزای اجرا کننده الگوریتم می‌تواند باعث همگرایی ضعیف شود[۱۸].

الگوریتم گرادیان نزولی تصادفی تقریبی از روش بهینه سازی گرادیان نزولی است که هر بار بر روی یک نمونه به جای کل داده‌ها عمل می‌کند. روش گرادیان نزولی تصادفی این محاسبات اضافی را با ایجاد یک تغییر در هر گرادیان گیری حذف می‌کند. روش گرادیان نزولی تصادفی تغییرات را با واریانس بالایی ایجاد می‌کند، بنابراین با استفاده از روش کاهش واریانس میزان تغییرات بهبود پیدا می‌کنند. الگوریتم گرادیان نزولی تصادفی گام به گام، به سمت کمترین

[1]Stochastic Gradient Descent(SGD)

میزان تابع حرکت می‌کند. با توجه به نحوه عملکرد این روش، گرادیان نزولی تصادفی بسیار سریعتر است و می‌تواند یادگیری لحظه‌ای هم داشته باشد. این روش یک الگوریتم محبوب برای آموزش طیف گسترده‌ای از مدل‌ها در الگوریتم‌های رده‌بندی به عنوان مثال رگرسیون لجستیک می‌باشد. روش گرادیان نزولی تصادفی تغییرات را با واریانس بالایی ایجاد می‌کند و این مسئله باعث بوجود آمدن نوسانات زیاد در مسیر یافتن کمترین مقدار تابع می‌شود. استفاده از این الگوریتم همگرایی خوبی فراهم می‌کند ولی میتواند به خصوص برای مجموعه داده‌های بزرگ کند باشد. بنابراین با گذشت زمان و لزوم استفاده از داده ها در مقیاس بزرگ ارائه الگوریتمی جدید به منظور افزایش سرعت الزامی است[۱۸] [۲۰].

۴.۲ رده‌بندی

رده‌بندی یکی از زیر شاخه‌های اساسی یادگیری ماشین و داده‌کاوی است. رده‌بندی علمی است که بر اساس داده‌های قبلی که دارای برچسب هستند، مدلی جهت پیش‌بینی داده‌های جدید می‌سازد. هدف از رده‌بندی این است که ابتدا با استفاده از مجموعه کوچکی از داده‌ها، یک مدل مناسب ساخته و سپس برمبنای مدل ایجاد شد، داده‌های را که در آینده مشاهده می‌شود به درستی رده‌بندی کرد.
رده‌بندی را می‌توان به عنوان یک روش با نظارت که در آن هر نمونه به یک کلاس با ویژگی خاص می‌باشد، دانست. هر نمونه شامل دو قسمت، مجموعه مقادیر ویژگی پیش‌بینی کننده و مقدار ویژگی هدف می‌باشد. اولین مورد به منظور پیش‌بینی ارزش بعدی استفاده می‌شود. ویژگی‌های پیش بینی کننده باید برای پیش بینی یک کلاس مناسب باشند. در رده‌بندی مجموعه نمونه‌های استخراج شده به دو مجموعه انحصاری متقابل و جامع که به آن‌ها مجموعه آموزش و آزمون گفته می‌شوند، تقسیم می‌شود. دانش کشف شده به وسیله الگوریتم رده‌بندی را می‌توان از طریق راه‌های مختلف بیان کرد که شامل بسیاری از الگوریتم‌ها و روش‌ها می‌شوند[۲۵].

بایاس و واریانس در رده‌بندی

مسائلِ حوزه‌ی رده‌بندی یک خروجیِ موردِ انتظار دارند و آن بدست آوردن برچسب‌ها با دقت بالا، شناسایی و تفکیک آن‌ها از یکدیگر می‌باشد. گاهی در رده‌بندی ممکن است خطوط رده بند با دقت بالایی داده ها را از هم تفکیک کنند که در این هنگام گفته می‌شود خطِ رده بند دارای پیچیدگی[1] زیادی است. معنیِ این خطِ پیچیده این است که رده‌بندی که این خط را رسم کرده، دارای بیش برازش است. یعنی کاملاً بر روی داده‌های آموزشیِ موجود، درست

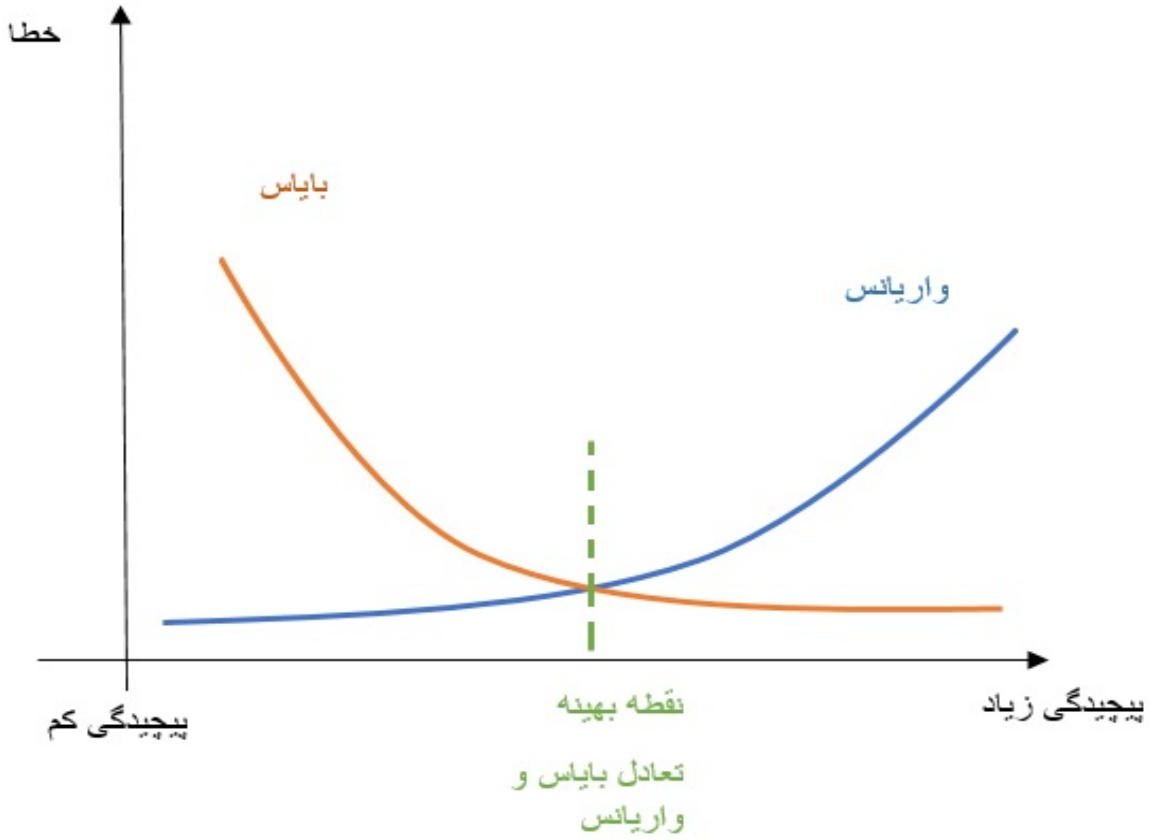
[1] Complexity

عمل می‌کند ولی اگر یک یا چند داده‌ی جدید ببیند، نمی‌تواند به درستی این داده‌ی جدید را رده‌بندی کند (برچسب بزند). یعنی خطای بالایی در رده‌بندی دارد. این مدلِ پیچیده (خطِ پیچیده) خطای رده‌بندی را بالا می‌برد (اگر چه که بر روی داده‌های آموزشیِ موجود خوب عمل می‌کند) و در نهایت واریانس[1]بالا را به همراه دارد[۲۶].

حال اگر خط تفکیک کننده خطی بسیار ساده باشد، این خط ساده بر روی داده‌های آموزشی (داده‌های موجود) نیز خطا دارد اما با این وجود نسبت به داده‌های جدید، خطای خیلی بالایی نخواهد داشت. یعنی هم بر روی داده‌های آموزشی و هم بر روی داده‌های جدید یک خطای نسبی دارد. در این جا مقدار بایاس[2]زیادی داریم یعنی خطِ ساده، چه در داده‌های آموزشی و چه در داده‌های جدید یک خطای نسبی دارد. بنابراین بایاس بالا سادگیِ زیادِ طبقه‌بند را در نتیجه دارد و واریانس بالا نیز پیچیدگیِ بیش از حد را به همراه دارد. حال به مفهومی به اسم تعادلِ بین بایاس و واریانس که به تعادل بین بایاس و واریانس[3]مشهور است، پرداخته می‌شود. در واقع این مفهوم بدین معناست که باید یک تعادلِ معقولی بین بایاس و واریانس یعنی بین سادگیِ زیادِ رده‌بند و پیچیدگیِ زیادِ آن برقرار شود تا یک الگوریتم رده‌بندی خوب بدست بیاید[۲۷].

تفسیر شکل ۳.۲ بسیار ساده است. با زیاد شدن پیچیدگیِ یک مُدل (به طور مثال یک الگوریتمِ رده‌بندی که از داده‌ها یادگیری را انجام داده باشد)، واریانس زیاد می شود (خط آبی) و با کم شدن پیچیدگی بایاس زیاد می‌شود (خط قرمز) و زیاد شدن هر دو باعث بالا رفتن مقدار خطای کل می‌شود (محور عمودی). همان طور که مشاهده می‌شود در نقطه سبز رنگ (که یک پیچیدگی معقول دارد) مقدار بایاس و واریانس در حد معقولی قرار می‌گیرد. از یک رده‌بند خوب انتظار می‌رود مدلی بسازد که یک پیچیدگی معقول (در نقطه‌ای نزدیک به نقطه سبز) به دست بیاورد. بدین مفهوم که نه زیاد پیچیده و نه زیاد ساده باشد. برای رسیدن به این نقطه سبز راهکارهای مختلفی وجود دارد. یکی از آن‌ها این است که داده‌های آموزشیِ به دو بخش آموزش و تست تقسیم شوند. رده‌بند بر اساس بخش داده‌های آموزشی شناخته شود و بعد از آن یک ارزیابی با توجه به داده‌های جدا شده بخشِ تست انجام پذیرد تا این موضوع که آیا بایاس یا واریانس رخ داده است یا خیر مشخص شود.

[1] Variance
[2] Bias
[3] Bias and Variance Tradeoff



شکل ۳.۲: تعادل بایاس و واریانس [۲۶]

۵.۲ گرادیان نزولی تصادفی همراه با کاهش واریانس

از آنجا که گرادیان نزولی تصادفی فقط یک تخمین تصادفی (توسط یک دسته کوچک از نمونه ها یا حتی یک نمونه واحد) از شیب دسته‌ای است، باید هنگام بروزرسانی در طول مسیر توجه داشت، به منظور اطمینان از همگرایی، نرخ یادگیری به دلیل واریانس نمونه گیری تصادفی، مجبور به کاهش به میزان صفر نشود. به طور معمول نرخ یادگیری را $\eta_t = O(\frac{1}{t})$ انتخاب می‌کنند. میزان نرخ یادگیری کوچک منجر به حل همگرایی زیر خطی $O(\frac{1}{t})$ می‌شود[۲۸].

همانطور که در جدول ۱.۲ مشاهده می‌شود در گرادیان نزولی باید تمام گرادیان‌ها را به تعداد سایز داده ورودی محاسبه کرد و این کار موجب کند شدن الگوریتم شده، و در داده‌های

جدول ۱.۲: مشکلات اساسی در گرادیان نزولی و گرادیان نزولی تصادفی [۲۸]

	پیچیدگی زمانی	نرخ یادگیری	نرخ همگرایی (محدب قوی)
گرادیان نزولی	محاسبه گرادیان‌ها به تعداد داده ورودی	ثابت	$O(\log t)$
گرادیان نزولی تصادفی	محاسبه یک گرادیان	$O(\frac{1}{t})$	$O(\frac{1}{t})$

با حجم بالا محاسبات دشوار می‌شود. همچنین با توجه به نرخ یادگیری انتخاب شده در گرادیان نزولی تصادفی، ایجاد واریانس بالا در الگوریتم موجب کاهش همگرایی نسبت به الگوریتم سنتی گرادیان نزولی می‌شود.

خوشبختانه، می‌توان روش‌هایی طراحی کرد که واریانس گرادیان تصادفی را کاهش می‌دهند. این الگوریتم با نام گرادیان نزولی تصادفی همراه با کاهش واریانس شناخته شده است[1] این الگوریتم‌ها این امکان را فراهم خواهند کرد که بتوان در مواقع لزوم نرخ یادگیری بزرگتری را در هنگام استفاده از روش بهینه سازی گرادیان نزولی تصادفی مورد استفاده قرار داد.

در واقع هدف از پیدایش الگوریتم گرادیان نزولی تصادفی همراه با کاهش واریانس، کاهش واریانس بوجود آمده طی بهینه سازی گرادیان نزولی تصادفی، اصلاح نرخ همگرایی کاهش یافته به دلیل بروز واریانس بالا در روش گرادیان نزولی تصادفی و در نهایت رهایی از دخیره سازی تمامی گرادیان‌ها می‌باشد[۲۹].
در جدول ۲.۲ مراحل الگوریتم کاهش واریانس نشان داده شده است:

جدول ۲.۲: مراحل الگوریتم گرادیان نزولی تصادفی همراه با کاهش واریانس[۲۹]

۱	مقدار دهی اولیه پارامتر مدل w_0 در یک حلقه خارجی $k = 1, \ldots$
۲	$\tilde{w} = w_{\tilde{t-1}}$ محاسبه $\tilde{\mu} = \nabla f(\tilde{w})$ $w_0 = \tilde{w}$ در یک حلقه داخلی $t = 1, \ldots, m$
۳	انتخاب نمونه i_t به صورت تصادفی و بروزرسانی پارامتر مدل یادگیری $w_t = w_{t-1} - \eta(\nabla f_{i_t}(w_{t-1}) - \nabla f_{i_t}(\tilde{w}) + \tilde{\mu})$
۴	پایان الگوریتم

[1] Stochastic Variance Reduced Gradient(SVRG)

تئوری الگوریتم:

در این روش با فرض اینکه گرادیان $L-smooth$ و تابع هدف $F(w)$ را به اندازه μ محدب قوی باشد الگوریتم پیشنهاد شده است و در نهایت $w^* = argmin_w F(w)$ قرار می‌دهیم. بنابراین در این الگوریتم انتظار یک همگرایی هندسی را خواهیم داشت.

$$\mathbb{E}F(\tilde{w}_k) - \mathbb{E}F(w^*) \leq \alpha^k (F(\tilde{w}_0) - F(w^*)) \tag{۳.۲}$$

همچنین اگر m را یک مقدار نسبتا بزرگ در نظر بگیریم:

$$\alpha = \frac{1}{L\eta m(1-2\mu\eta)} + \frac{2\mu\eta}{1-2\mu\eta} < 1 \tag{۴.۲}$$

در واقع m به همان میزان از n انتخاب می‌شود. بنابراین جهت میزان دقیق‌تر همگرایی خواهیم داشت:

$$\mathbb{E}F(\tilde{w}_k) - \mathbb{E}F(w^*) \leq \alpha^{\frac{t}{n}} (F(\tilde{w}_0) - F(w^*)) \tag{۵.۲}$$

کاهش واریانس در الگوریتم گرادیان تصادفی اجازه می دهد تا از یک نرخ یادگیری ثابت استفاده شود و درنهایت نرخ همگرایی خطی حاصل می‌شود.می‌توان خاطر نشان کرد که این روش نیازی به ذخیره‌سازی گرادیان‌ها در طول محاسبات نداشته و همچنین در حالت غیر محدب نیز کارایی بالایی دارد[۳۰].

۶.۲ سیستم‌های توزیع‌شده

گرادیان نزولی تصادفی با تکنیک های کاهش واریانس برای آموزش پارامترهای مدل‌های مختلف مسئله رده‌بندی عملکرد مطلوبی دارد، اما نمی‌تواند از سیستم های توزیع‌شده بطور جزئی پشتیبانی کند[۳۱].

با توجه به پیچیدگی مسائل رده‌بندی، اندازه بزرگ داده‌های آموزش و انبوه محاسبات، یک روش آموزش کارآمد برای انجام یک کار یادگیری در مقیاس بزرگ از اهمیت حیاتی برخوردار است. بنابراین با ایجاد یک سیستم توزیع‌شده در الگوریتم گرادیان نزولی تصادفی همراه با کاهش واریانس، می‌توان سرعت این الگوریتم را افزایش داد[۳۲].

یک سیستم توزیع‌شده مجموعه ای از فرآیندهای مستقل است. این سیستم‌ها حجم عظیمی از داده‌ها را نگهداری می‌کنند، سرعت بالا و همچنین قابلیت اطمینان بالایی نیز دارند. یکی از پارامترهای مهم الگوریتم گرادیان نزولی تصادفی نرخ یادگیری است که به تسریع بر یادگیری داده‌ها تاثیر می‌گذارد. به این صورت که اگر نرخ یادگیری بسیار زیاد باشد نقاط کمینه را نادیده می‌گیرد و اگر کم باشد الگوریتم در یک بهینه محلی گیر می‌کند، که در آن موقعیت هم با افزایش نرخ یادگیری می‌توان از بهینه های محلی عبور کرد.

بنابر این یکی دیگر از فواید استفاده سیستم‌های توزیع‌شده در روش گرادیان نزولی تصادفی این است که در هر تکرار نرخ یادگیری را تغییر داده و الگوریتم را سرعت می‌بخشیم. در یک سیستم توزیع‌شده مواردی مانند شفافیت،انعطاف‌پذیری و قابلیت اطمینان لزوما باید طراحی مورد توجه واقع شوند.

در دسترس بودن یک فاکتور مهم مرتبط با سیستم‌های توزیع‌شده است. طراحی نباید به گونه‌ای باشد که نیاز به اجرای همزمان اجزای اساسی در این سیستم‌ها باشد. افزونگی بیشتر داده‌ها باعث افزایش در دسترس بودن شده اما ناسازگاری را بیشتر می‌کند. قدرت تحمل نقص[۱]باعث پوشاندن خطاهای ایجاد شده توسط کاربر می‌شود[۳۳].

۷.۲ محیط پیاده‌سازی

آپاچی اسپارک[۲] یک چارچوب رایانش توزیع‌شده است. اسپارک یک رابط برنامه‌نویسی کاربردی برای برنامه‌نویسی تمام خوشه‌ها با موازی‌سازی داده‌های ضمنی و تحمل خطا فراهم می‌کند. اسپارک قادر است چندین پتابایت داده را که بر روی خوشه‌هایی از هزاران سرور فیزیکی یا مجازی توزیع‌شده‌اند پردازش نماید. برخی ویژگی‌های منحصر بفرد اسپارک موجب شده است که این برنامه نسبت به برخی دیگر از گزینه‌های مشابه هدوپ[۳] برای پردازش داده‌ها مانند برنامه نگاشت_کاهش بهتر عمل کند. در واقع می‌توان گفت که اسپارک از همان ابتدا

[1] Fault tolerance
[2] Apache Spark
[3] Hadoop

به گونه‌ای طراحی و بهینه شده بود که عملیات پردازش را در حافظه انجام دهد و این مزیت نسبت به برنامه نگاشت – کاهش که داده‌ها را بر روی دیسک نوشته و از روی دیسک نیز برای پردازش فراخوانی می‌کند موجب سرعت فوق‌العاده بالاتری شده است. توانایی اسپارک برای ذخیره داده‌ها در حافظه موقت و اجرای پردازش‌ها و جستارهای تکراری آن را برای پیکربندی الگوریتم‌های یادگیری ماشین مناسب ساخته تا بتوان مجموعه فعالیت‌های تکراری را به راحتی و با سرعت بالا بر روی داده‌های جدید و عظیم نیز اجرا کرد. اسپارک با داشتن موتور اجرای بسیار پیشرفته خود از جریان غیرخطی داده‌ها و پردازش درون حافظه‌ای پشتیبانی می‌کند و بدین ترتیب سرعت عملیات افزایش می‌یابد[۳۴].

۸.۲ کارهای پیشین

امروزه به دلیل فراوانی داده‌ها با حجم بالا در مسائل رده‌بندی با مشکلاتی روبرو هستیم، که در فصل قبل به آن اشاره شد. بنابراین به جهت رفع مشکلات مطرح شده رده‌بندی برای داده‌های با حجم بالا ارائه یک روش جهت این امر الزامی است. استفاده از گرادیان نزولی یکی از پرکاربردترین راهکارها جهت بهینه سازی پارامترها برای کاهش خطای رده‌بندی در روش‌های مبتنی بر جداساز است. در این روش انبوه محاسبات اضافی و غیر ضروری برای مجموعه داده‌های بزرگ انجام می‌پذیرد، زیرا در این الگوریتم گرادیان داده‌های یکسانی را به طور مکرر قبل از ایجاد هر تغییر محاسبه می‌کند. روش گرادیان نزولی تصادفی این محاسبات اضافی را با ایجاد یک تغییر با هر گرادیان‌گیری حذف می‌کند.

روش گرادیان نزولی تصادفی تغییرات را با واریانس بالایی ایجاد می‌کند، بنابراین با استفاده از روش کاهش واریانس میزان تغییرات را بهبود می‌بخشد. الگوریتم گرادیان نزولی تصادفی گام به گام، به سمت کمترین میزان تابع حرکت می‌کند. امروزه الگوریتم‌های رده‌بندی با حجم بالایی از داده‌ها مواجه است، از این رو جهت رده‌بندی بهتر داده‌ها از الگوریتم گرادیان نزولی تصادفی استفاده می‌کنیم. با این که استفاده از این الگوریتم همگرایی خوبی را فراهم می‌کند، اما در بسیاری از موارد در مواجهه با داده‌ها با حجم بالا به کندی عمل می‌کند.

بنابراین اجرای غیر همزمان از الگوریتم گرادیان نزولی تصادفی برای داده‌ها با حجم بالا سریعتر است. اما باید بدانیم که در این حالت ارتباط غیر بهینه بین اجزای اجرا کننده الگوریتم می‌تواند باعث همگرایی ضعیف شود. بنابراین ما نسل جدیدی از الگوریتم الگوریتم گرادیان نزولی تصادفی را با عنوان گرادیان نزولی تصادفی توزیع‌شده معرفی می‌کنیم. این الگوریتم ترکیبی از پروتکل ارتباطی آسنکرون و کاهش واریانس می‌باشد.

روش گرادیان نزولی به منظور یافتن کمترین مقدار تابع در یک همسایگی مورد استفاده قرار می‌گیرد. در این الگوریتم برای ایجاد هر تغییر باید گرادیان کل داده‌ها محاسبه شود. روش گرادیان می‌تواند بسیار کند باشد و برای مجموعه‌ای از داده ها که به قدری بزرگ هستند، که در حافظه فضای کافی برای آنها وجود ندارد، قابل قبول نمی‌باشد. در ضمن این روش به ما اجازه نمی‌دهد که مدل را لحظه‌ای تغییر دهیم، این بدان معناست که نمی‌توان داده‌های جدید را حین کار به آن اضافه کرد. روش گرادیان نزولی انبوه محاسبات اضافی و غیر ضروری برای مجموعه داده‌های بزرگ انجام می‌دهد زیرا گرادیان داده‌های یکسانی را به طور مکرر قبل از ایجاد هر تغییر محاسبه می‌کند[۱۷].

گرادیان نزولی تصادفی[1] یک روش بهینه‌سازی جهت یافتن کمترین مقدار توابع است. تحقیقات در مورد گرادیان نزولی تصادفی به‌تازگی در یادگیری ماشین برای بهینه‌سازی توابع ضرر محدب و آموزش شبکه‌های عصبی عمیق به‌دست‌آمده‌است. این الگوریتم به دلیل نوع انتخاب داده‌ها به صورت تصادفی برای به‌روزرسانی داده‌ها نام‌گذاری شده‌است. [۳۵] . این نظریه بیان‌می‌کند که توابع به‌صورت گرادیان تصادفی یک تابع، با معادله $w = w - \mu \nabla f(w, x, y)$ محاسبه می‌شوند. در این فرمول پارامتر وزن بروزرسانی می شود و سپس در تابع گرادیان قرار داده شده که یک نرخ یا ضریب یادگیری بر میزان این تابع در هر مرحله تاثیر می‌گذارد. [۳۶].

با فراگیری راه‌حل‌های بر پایه داده‌ها در مقیاس بزرگ برای چالش‌های صنعتی و ظهور خوشه‌های مسائل رده‌بندی سطح بالا با تعداد کم کاربر، توزیع الگوریتم گرادیان نزولی تصادفی به منظور افزایش سرعت آن رویکرد قابل تأملی است [۳۷].

روش گرادیان نزولی تصادفی یک الگوریتم ترتیبی است، بدین‌معنا که گام به گام به سمت کمترین مقدار تابع حرکت می‌کند. همچنین در این الگوریتم برای هر مقدار x یک مقدار y در نظر گرفته می‌شود. بنابراین الگوریتم با گام‌های کوچکی که به سمت هدف برداشته می‌شود، باعث ایجاد ارتعاشات بسیار زیادی در مسیر رسیدن به کمینه تابع شده و باعث کندی همگرایی می‌شود. بنابراین روشی با نام گرادیان نزولی تصادفی همراه با کاهش واریانس ارائه شده است که در این روش کاهش واریانس گرادیان تصادفی اجازه می‌دهد تا از نرخ یادگیری ثابت استفاده شود و در نهایت همگرایی خطی حاصل می‌شود. این روش به دلیل تعداد محاسبات بالا در هر دوره، با این که به ذخیره کردن گرادیان‌های قبلی احتیاج ندارد، در حالت غیر محدب همگرایی مطلوبی حاصل نمی‌کند [۳۸].

هر چند گرادیان نزولی تصادفی ساده مقیاس‌پذیر است اما باعث افزایش واریانس می‌شود، به این جهت که گرادیان نزولی تصادفی باید گام‌به‌گام کاهش یابد تا واریانس کنترل شود.

[1] Stochastic Gradient Descent(SGD)

انتخاب نرخ یادگیری مناسب می تواند سخت باشد. اگر نرخ یادگیری زیادی کوچک باشد باعث می‌شود همگرایی به شدت کند باشد، و از طرف دیگر اگر زیادی بزرگ باشد برای همگرایی مزاحمت ایجاد کرده و باعث شود تابع ضرر در اطراف مقدار کمینه نوسان کند یا حتی واگرا شود. با توجه به این‌که مجموعه آموزشی محدود است، تعدادی تکنیک برای کاهش واریانس معرفی شده‌است. این روش ضریب گرادیان را طوری تنظیم می‌کند که واریانس را در هر مرحله کاهش می‌دهد. یکی دیگر از چالش‌های کلیدی کمینه کردن توابع ضرر پرکاربرد در سیستم های عصبی که به شدت غیرمحدب هستند، پیشگیری از توقف کردن در کمینه‌های نسبی است که نسبت به کمینه‌های اصلی بهینه نیستند. چالش اصلی، کمینه‌های نسبی نیستند بلکه نقاط زینی اند؛ یعنی نقاطی که شیب در یکی از جهت ها مثبت و در جهت دیگر منفی است. این نقاط زینی معمولا با سطحی صاف پر از نقاط مشابه احاطه شده‌اند که خروج از آن‌ها را برای روش گرادیان نزولی تصادفی دشوار می‌کند زیرا گرادیان در همه‌ی جهات نزدیک صفر است. [۳۹، ۴۰].

استفاده از این الگوریتم گرادیان نزولی تصادفی به دلیل ارتعاشات و همچنین تغییر نرخ یادگیری در هر مرحله از بروزرسانی پارامترها همگرایی خوبی فراهم نمی‌کند و می‌تواند به خصوص برای داده‌های بزرگ کند عمل کند. در مسایل رده‌بندی پیچیده با استفاده از روش گرادیان نزولی تصادفی به دلیل استفاده از داده‌هایی با حجم بالا و افزایش محاسبات نیازمند کاهش نوسانات در این الگوریتم بوده، بنابراین روشی با رویکرد کاهش واریانس ارائه شد، اما این روش در حالت تحدب قوی باعث کاهش همگرایی نسبت به گرادیان نزولی تصادفی دارد. بنابراین روش دیگری با نام گرادیان نزولی تصادفی بازگشتی[1] جهت بهبود عملکرد در حالت تحدب قوی ارائه شده است. در این روش با محاسبه گرادیان در دو حلقه خارجی و داخلی به صورت بازگشتی، دیگر نیازی به نگه‌داشتن گرادیان از دوره‌های قبلی وجود ندارد و این امر موجب بهبود عملکرد الگوریتم می‌شود [۴۱].

بنابراین می‌توان گفت اجرای غیر همزمان (آسنکرون) گرادیان نزولی تصادفی به دلیل این که در هر دوره بروزرسانی پارامترها به صورت آسنکرون الگوریتم را پیش می‌برد، می‌تواند از ارتعاشات کاسته و الگوریتم با سرعت بیشتری اجرا شود، اما ارتباط غیر بهینه بین اجزای اجراکننده الگوریتم می‌تواند باعث همگرایی ضعیف شود. به‌علاوه، می‌توان گرادیان نزولی تصادفی را تنها برروی یک دستگاه بدون نیاز خوشه‌های پردازشی بزرگ هم به صورت موازی اجرا کرد. این الگوریتم را می‌توان با عنوان الگوریتم گرادیان نزولی تصادفی موازی معرفی کرد [۴۲، ۴۳].

امروزه گرادیان تصادفی نزولی به صورت گسترده‌ای برای آموزش پارامترهای مدل در

[1] StochAstic Recursive grAdient algoritHm(SARAH)

سیستم‌های توزیع‌شده مورد استفاده قرار می‌گیرد. ارائه نسخه‌ی توزیع‌شده از گرادیان نزولی تصادفی توسط گرادیان نزولی تصادفی بارشی[۱] در یک سیستم پارامتر سرور مشخص شده است. گرادیان نزولی تصادفی بارشی یک نوع از گرادیان تصادفی است که داده‌های آموزشی را به تعداد زیادی زیر مجموعه تقسیم می‌کند و یک نسخه از مدل را در هر یک از این زیر مجموعه‌ها اجرا می‌کند . در این روش دو نکته وجود دارد، اول این‌که نسخه‌های مدل مستقل اجرا می‌شوند، دوم این‌که پارامترهای مدل به‌طور مستقل اجرا می‌شوند. این الگوریتم نرخ همگرایی را تضمین نمی‌کند[۴۶، ۴۵، ۴۴].

بنابراین جهت بهبود الگوریتم از یک پروتکل ارتباطی با نام پورت سریال سنکرن[۲] استفاده شده است. از این پروتکل ارتباطی جهت ارتباط میان گره‌ها در بروزرسانی پارامترهای مورد استفاده قرار می‌گیرد. این پروتکل هم در تئوری و هم در حالت تمرینی قدرت خود را ثابت کرده است [۴۷، ۴۸، ۴۹].

بنابراین نسخه دیگری با عنوان Petuum ارائه شد که این روش به منظور کاهش واریانس با استفاده از تعیین نرخ یادگیری مناسب همگرایی را افزایش می‌دهد. می‌توان گفت در این روش تمرکز کار بیشتر بر روی پروتکل ارتباطی بوده است. از آنجا که در سیستم‌های توزیع‌شده با گرادیان نزولی تصادفی پارامترهای زیادی برای بروزرسانی وجود دارند بنابراین از یک سیستم پارامتر سرور جهت بروزرسانی استفاده می‌شود که در این سیستم‌ها گره‌ها با یکدیگر در ارتباطند. بنابراین نسخه Petuum یک پروتکل ارتباطی جهت ارتباط گره‌ها با یکدیگر پیشنهاد می‌کند. در این روش پارامترها با تضعیف نرخ یادگیری بروزرسانی می‌شوند. این کاهش نرخ یادگیری باعث می‌شود به سختی تابع هدر را کاهش دهیم بنابراین همگرایی کاهش می‌یابد [۵۲، ۵۰، ۵۱].

در این پژوهش به جهت مشکلات محاسبات در مسائل رده‌بندی با حجم بالای داده مانند سرعت پایین و عدم پشتیبانی الگوریتم گرادیان نزولی تصادفی بازگشتی از سیستم‌های توزیع‌شده، یک مسئله رده‌بندی را در یک سیستم توزیع‌شده با استفاده از الگوریتم گرادیان نزولی تصادفی بازگشتی، که خود بهبود یافته الگوریتم گرادیان نزولی است،پیاده‌سازی کرده و مورد ارزیابی قرار داده شده است. شرح پژوهش انجام شده و نتایج حاصل از ارزیابی عملکرد روش پیشنهادی به ترتیب در فصل سوم و چهارم ارائه خواهد شد.

در جدول ۳.۲ می‌توان به جمع بندی کلی از پژوهش‌های پیشین نگاهی اجمالی داشت .

[1] DownPour SGD
[2] Synchronous Serial Port

جدول ۳.۲: مقایسه زمانی الگوریتم پیشنهادی و الگوریتم پایه

روش	شرح مختصر	معایب
۱ـ گرادیان نزولی	گرادیان تابع هزینه را نسبت به بردار پارامتر w برای کل مجموع داده‌های ورودی محاسبه می‌کند[۱۷]	کندی در الگوریتم و عدم تغییر لحظه‌ای
۲ـ گرادیان نزولی تصادفی	این روش با ایجاد یک تغییر قبل از محاسبه گرادیان در هر دوره محاسبات اضافی را حذف کرده و سرعت الگوریتم را بالاتر می‌برد[۳۵]	افزایش نوسانات در الگوریتم موجب واریانس بالا می‌شود.
۳ـ گرادیان نزولی تصادفی همراه با کاهش واریانس	این روش با استفاده از دو حلقه یکی حلقه خارجی و دیگری حلقه داخلی به محاسبه گرادیان پرداخته و با استفاده از محاسبه گرادیان بهینه هر دوره و کم کردن این میزان از گرادیان کل در مرحله قبل موجب کاهش واریانس بوجود آمده شده و از طرفی نیاز به ذخیره سازی گرادیان‌ها در دوره‌های قبلی ندارد[۳۸]	در حالت غیر محدب عملکرد خوبی نداشته و همگرایی کندی را موجب می‌شود.
۴ـ گرادیان نزولی تصادفی بازگشتی	در دو حلقه یکی خارجی و دیگری داخلی گرادیان تصادفی را به صورت بازگشتی محاسبه می‌کند[۴۱]	عدم پشتیبانی از سیستم توزیع‌شده
۵ـ گرادیان نزولی تصادفی موازی	گرادیان نزولی تصادفی را تنها بر روی یک دستگاه بدون نیاز خوشه‌های پردازشی بزرگ هم به صورت موازی اجرا می‌کند[۴۳، ۴۲]	موازی سازی روی خوشه صورت می‌گیرد نه روی نودها
۶ـ روش گرادیان نزولی تصادفی DownPour	داده‌های ورودی را به تعدادی زیرمجموعه تقسیم می‌کند و یک نسخه از مدل را در هر زیر مجموعه اجرا می‌کند [۴۶، ۴۵، ۴۴]	عدم تضمین نرخ همگرایی در الگوریتم
۷ـ روش Petuum	بهره‌گیری از یک پروتکل ارتباطی جهت بروزرسانی پارامترها [۵۲، ۵۰، ۵۱]	سیاست کاهش نرخ یادگیری که موجب کاهش همگرایی می‌شود.

۹.۲ جمع‌بندی

در این فصل ابتدا به تشریح مفاهیم اولیه در گرادیان نزولی و چالش‌های موجود در این الگوریتم پرداخته شد. سپس روش بهبود یافته گرادیان نزولی تصادفی و مشکلات آن را مورد بررسی قرار داده و در نهایت به شرح الگوریتم گرادیان نزولی تصادفی همراه با کاهش واریانس به تفصیل پرداخته شد. بعد از پرداختن به تعاریف اولیه در الگوریتم بهینه سازی سیستم‌های توزیع‌شده و کاربردهای آن‌ها را مورد بحث قرار گرفت. در نهایت با توجه به بررسی پژوهش‌های پیشین وجود مشکلاتی مانند عدم پشتیبانی از سیستم‌های توزیع شده در بهبود الگوریتم گرادیان نزولی تصادفی، نگهداری نرخ همگرایی خطی و همچنین افزایش سرعت در بروزرسانی پارامترها در سیستم توزیع شده، الگوریتم پیشنهادی ارائه شده است. بنابراین با توجه به مشکلات مطرح شده در فصل سوم، الگوریتم پیشنهادی در یک سیستم توزیعی به جهت افزایش سرعت عملکرد الگوریتم و همچنین نمونه‌برداری تصادفی برای حفظ همگرایی، مورد بررسی قرار گرفته‌است.

فصل ۳

روش پیشنهادی جهت رده‌بندی توزیع‌شده

همانطور که در فصول گذشته بیان شد، گرادیان نزولی تصادفی یک تقریب از روش بهینه سازی گرادیان نزولی است که بر مبنای تکرار محاسبه گرادیان یک تابع مشتق پذیر، به جهت کمینه کردن آن تابع (تابع هدف) مورد استفاده قرار می‌گیرد. همچنین در فصل دوم به چالش‌های موجود در این روش بهینه‌سازی اشاره شد. یکی از مشکلات اساسی این الگوریتم بهینه سازی، نوسانات شدید آن در هنگام نزدیک شدن به نقطه بهینه است. این نوسانات شدید، حاکی ازایجاد واریانس بالا در این الگوریتم است. بنابراین باید روشی در این الگوریتم ارائه شود که موجب کاهش واریانس در محاسبات شده و همچنین همگرایی الگوریتم را حفظ کند.

از طرفی دیگر انتخاب نرخ یادگیری صحیح در الگوریتم گرادیان نزولی تصادفی یکی از مشکلات پیش رو در الگوریتم می‌باشد. نرخ یادگیری در واقع نشان دهنده اندازه هر گام برای نزدیک شدن به کمترین میزان تابع هدف الگوریتم می‌باشد. بنابراین انتخاب صحیح این پارامتر در کارایی الگوریتم تأثیر به سزایی خواهد داشت. بنابراین می‌توان به این مورد اشاره کرد که رده‌بندی داده‌ها به صورت توزیع‌شده، با استفاده از روش گرادیان نزولی تصادفی سرعت

محاسبات را افزایش می‌بخشد. بنابراین استفاده از یک سیستم توزیع‌شده در این رده‌بندی موجب افزایش سرعت در انجام محاسبات در هر دوره از تکرار الگوریتم خواهد شد.

از این رو ، الگوریتم معرفی شده با نام گرادیان نزولی تصادفی بازگشتی مشکلاتی از قبیل افزایش واریانس در روش گرادیان نزولی تصادفی و همچنین رفع عملکرد نامناسب در الگوریتم گرادیان نزولی تصادفی همراه با کاهش واریانس را در یک حالت تک گره برطرف می‌کند. این ماهیت الگوریتم بازگشتی که تنها با یک گره در یک خوشه محاسبات را انجام می‌دهد در استفاده از مجموعه داده‌های کلان موجب کندی پردازش در مسئله رده‌بندی می‌شود. بنابراین با ارائه یک سیستم توزیعی که بتوان الگوریتم بازگشتی را در آن پیاده‌سازی کرد، به منظور افزایش سرعت در مسئله رده‌بندی، الگوریتم پیشنهادی در ادامه معرفی خواهد شد.

۱.۳ تعاریف اولیه در الگوریتم پیشنهادی

همان‌طور که در فصل‌های قبل نیز به آن اشاره شد الگوریتم گرادیان نزولی روشی جهت بهینه کردن توابعی مانند تابع (۱.۳) که دارای کمترین و بیشترین مقدار محلی هستند، می‌باشد.

$$P(w) := \frac{1}{n}\sum_{i=1}^{n} f_i(w) \qquad (۱.۳)$$

در این معادله f_i به صورت $f_i \quad , i \in [n] = 1, ..., n$ تعریف می‌شود.

این تابع یک تابع محدب، که تحت استمرار لیپشیتز[1] ایجاد شده است، می‌باشد. لازم به ذکر است که در مسائل کمینه سازی باید شروط تحدب برقرار باشد. بنابراین یکی از این شروط را که تحدب قوی[2] نام دارد مطرح می‌کنیم.
گرادیان تابع f_i با یک پارامتر $L > 0$ یک استمرار لیپشیتز نامیده می‌شود اگر:

$$\| \quad \nabla f(x) - \nabla f(y) \quad \| \quad \leq L \quad \| \quad x - y \quad \| \quad for \quad all \quad x, y \in dom(f) \qquad (۲.۳)$$

در تجزیه و تحلیل ریاضی، استمرار لیپشیتز که بر اساس نام رادواف لیپشیتز نام‌گذاری شده است، یک حالت قوی از استمرار یکنواخت[3] برای توابع است.
یک تابع استمرار لیپشیتز با توجه به اینکه با چه سرعتی تغییر می‌کند، محدود می‌شود. در

[1] Lipschitz Continuous
[2] Strong Convexity
[3] Uniform Continuous

واقع مقداری واقعی به گونه‌ای برای هر جفت از نقطه‌ها در نمودار تابع وجود دارد که مقدار مطلق شیب خط اتصال دهنده تابع از این مقدار بیشتر نمی‌باشد. این محدوده کوچک را ثابت لیپشیتز[1] می‌نامند. همچنین برای تابع f_i اگر در آن مقدار $\mu > 0$ را درنظر بگیریم، به ازای هر $\alpha \in [0,1]$ رابطه (۳.۳) برقرار می‌باشد:

$$f(\alpha x_1 + (1-\alpha \quad x_2)) \leq \alpha f(x_1) + (1-\alpha) f(x_2) - \frac{1}{2}\alpha(1-\alpha)\mu \| \quad x_1 - x_2 \quad \|^2 \qquad (۳.۳)$$

می‌توان مسائلی از این قبیل را در دسته مسائل کاربردی نظارتی در نظر گرفت. اگر مدل الگوریتمی یادگیری ماشین را به صورت رابطه (۴.۳) درنظر گرفته‌شود:

$$min_w F(w) \qquad F(w) = \quad \frac{1}{n}\sum_{i=1}^{n} f_i(w) + R(w) \qquad (۴.۳)$$

در واقع هدف از الگوریتم پیشنهادی یافتن کمترین مقدار بهینه برای این مدل یادگیری ماشین می‌باشد.

۱.۱.۳ تابع هزینه

در مسئله رده‌بندی در رابطه (۴.۳) باید به دنبال کمترین میزان برای تابع $f_i(w)$ بود. به این تابع، تابع هزینه گفته می‌شود. در واقع در الگوریتم های یادگیری ماشین هدف یافتن کمترین میزان تابع هزینه می‌باشد. در اینجا برای تابع $f_i(w)$ که $1 < i < n$ بیانگر ضرر عینی مربوط به i امین نمونه در تابع می‌باشد. w پارامتر مدل را نشان می‌دهد که این پارامتر در هر دوره الگوریتم ما نیاز به به‌روزرسانی دارد. n در اینجا نشان دهنده سایز داده در داده آموزشی می‌باشد. می‌توان اذعان داشت که تابع هزینه در اینجا برابر است با محاسبه تابع هزینه در هر دوره به اضافه تابع تنظیم که کمترین مقدار بدست آمده از این محاسبات، نتیجه مورد نظر در این بهینه سازی می‌باشد. تابع $f(w)$ برای کمینه شدن با پارامترهایی به صورت مکرر به‌روزرسانی می‌شود که این فرآیند را یادگیری می‌نامیم.

۲.۱.۳ تابع تنظیم

زمانیکه الگوریتم‌های رده‌بندی پیچیده برای مسائل ساده در نظر گرفته می‌شود، محاسبات الگوریتم بعد از چندین دوره تکرار دچار بیش برازش می‌شود که در این شرایط یکی از راه‌های جلوگیری از بیش برازش کاهش مقدار تابع هزینه می‌باشد. یکی از این راه حل‌های پیشنهادی این است که بخشی از الگوریتم کنار گذاشته شود و راهی دیگر برای رفع این مسئله صفر کردن

[1] Lipschitz Constant

برخی از وزن ها از طریق آیتم تنظیم می‌باشد که در این روش وزن‌های کوچک تشویق به بزرگ شدن و وزن ها بزرگ جریمه شده و به صفر میل می‌کنند. روش‌های مرسوم برای این مسئله روش‌های نرم L1 و نرم L2 می‌باشند که تحت لاسو و ریج[1] ارائه شده‌اند. در روش L1 از جمع قدرمطلق وزن‌ها و در روش L2 از جمع مجذور وزن‌ها استفاده می‌شود. البته روش سومی هم وجود داره که از ترکیب دو روش L1 و L2 بدست می‌آید.

در اینجا از تابع هزینه به این مقصود استفاده می‌شود که مقدار w بهینه با کاهش $f(w)$ بدست آورده می‌شود. در اصل این پارامتر تنظیم نشان دهنده دانش اولیه‌ای از مسئله می‌باشد که در اختیار ما قرار می‌گیرد. بنابراین با افزایش تابع تنظیم تابع بهینه سازی به جای کاهش کل تابع $f(w)$ می‌تواند w کوچکتری را انتخاب می‌کند. این عمل باعث جلوگیری از بیش برازش در تابع هزینه می‌شود. به عنوان مثال اگر یک مسئله که شامل راه حل‌های تنک می‌باشد (که پارامترهای آن شامل تعداد صفر زیادی می‌باشد) مورد بررسی قرار داده شود، میزان تابع تنظیم به طور معمول به صورت نرم L_1 به شکل $R(w) = \|w\|_1$ نمایش داده می‌شود.
در سال‌های اخیر روش‌های پیشرفته بهینه سازی جهت حل مسئله (۴.۳)پیشنهاد شده‌اند.

۳.۱.۳ گرادیان نزولی تصادفی

هنگامی که تابع هدف ساده و محدب است، روش‌های بهینه سازی مرسوم مثل گرادیان نزولی در حالتی که n تعداد نمونه‌های آموزشی و تابع f_i بزرگ باشد، به طور معمول غیر عملی می‌باشند. در حالت کلی به‌روزرسانی در گرادیان نزولی طبق فرمول (۵.۳) صورت می‌پذیرد.

$$w_{t+1} = w_t - \eta_t \nabla P(w_t) \quad , \quad t = 0, 1, 2, ... \tag{۵.۳}$$

تحت فرضیه‌ای مبنی بر محدب قوی بودن بر روی P و با یک انتخاب صحیح برای η_t، گرادیان نزولی در یک نرخ خطی همگرا می‌شود. با این حال اگر n بزرگ باشد محاسبه $\nabla P(w_t)$ در هر تکرار با مشکل روبرو می‌شود.
به عنوان جایگزین گرادیان نزولی تصادفی یک روش برای حل این مشکلات شناخته شده است. در هر مرحله از الگوریتم گرادیان نزولی تصادفی یک نمونه یکنواخت از i را به صورت تصادفی انتخاب کرده، و طبق فرمول (۶.۳) پارامتر مدل را به‌روزرسانی می‌کند.

$$w_{t+1} = w_t - \eta_t \nabla F_i(w_t) \quad , \quad t = 0, 1, 2, ... \tag{۶.۳}$$

می‌توان با بررسی دو الگوریتم نتیجه گرفت که همگرایی در روش گرادیان نزولی تصادفی نسبت به گرادیان نزولی کندتر می‌باشد. در واقع می‌توان گفت این روش به دلیل واریانس شدید که در

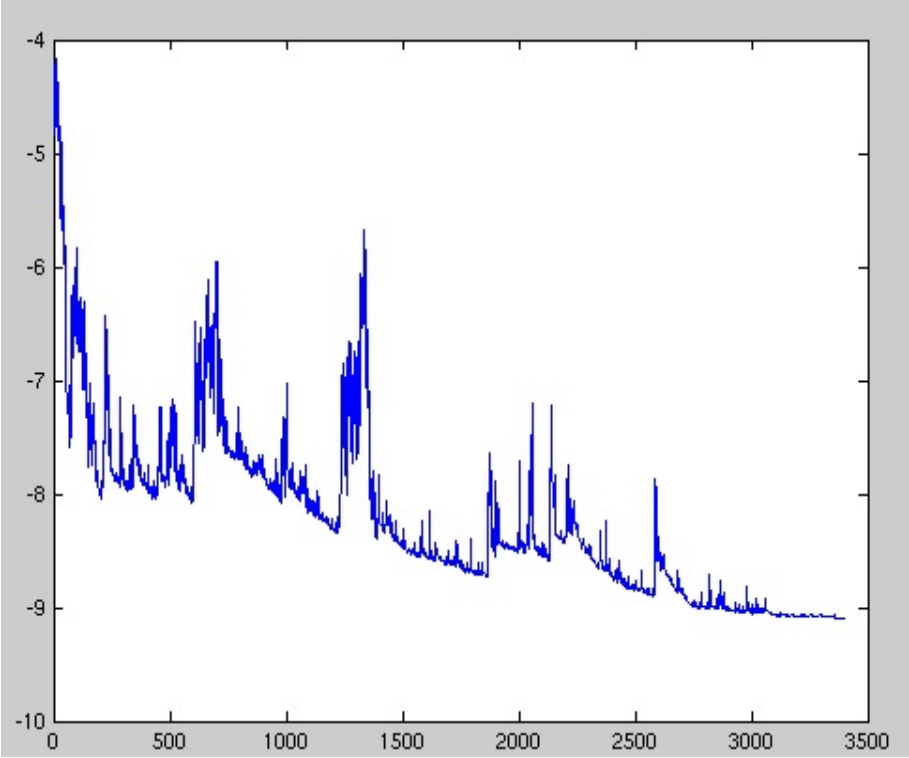
[1] lasso , Ridge

شکل ۱.۳ مشاهده می‌شود، بسیار کند است. با این وجود به دلیل صرفه جویی قابل توجهی در زمان برای هر تکرار، این الگوریتم سودمند است. همانطور که در جدول ۱.۳ مشاهده می‌شود،

جدول ۱.۳: شبه کد الگوریتم گرادیان نزولی تصادفی

الگوریتم گرادیان نزولی تصادفی	Stochastic Gradient Descent Algorithm
۱.انتخاب پارامتر اولیه مدل	1.choosen w_0
۲.برای هر نمونه از صفر تا بی نهایت	2.For t = 0,1,2,...
۳. فرمول $w_{t+1} = w_t - \eta_t \nabla F_i(w_t)$ به‌روزرسانی شود	3. $w_{t+1} = w_t - \eta_t \nabla F_i(w_t)$

الگوریتم گرادیان نزولی تصادفی در یک حلقه پارامترهای مدل یادگیری ماشین را به‌روزرسانی کرده و با توجه به کاهش سرعت در محاسبات آن‌ها را بدست می‌آورد.



شکل ۱.۳: میزان شدت نوسانات در الگوریتم گرادیان نزولی تصادفی [۱۸]

۴.۱.۳ گرادیان نزولی تصادفی نیمه انبوه

در روش گرادیان نزولی تصادفی برای هر نمونه i در دوره t که به صورت غیر تصادفی(یکنواخت) انتخاب شده است، مشاهده می‌شود که هر نمونه وابسته به کل سایز داده ورودی می‌باشد.

بنابراین در این حالت می‌توان از مدل دیگری از این الگوریتم که آن را الگوریتم گرادیان نزولی تصادفی نیمه انبوه[1] می‌نامند، استفاده کرد. در این روش به جای استفاده از کل سایز ورودی، یک مجموعه به صورت تصادفی $I_k \subset 1, ..., n$ انتخاب انجام می‌گیرد، که میزان آن برابر $b \ll n$ می‌باشد و به روزرسانی همانند فرمول (۷.۳) صورت می‌پذیرد :

$$w_{t+1} \quad = \quad w_t - \eta_t \frac{1}{b} \sum_{i=1}^{n} \nabla F_i(w_t) \quad , \quad t = 0, 1, 2, ... \tag{۷.۳}$$

در هر دو حالت گرادیان (معادلات ۸.۳ و ۹.۳) کامل تخمین زده شده همراه با داده اضافی (نویز) همراه است که می‌توان نتیجه گرفت که این تخمین نویزی بایاس نمی‌باشد.

$$\mathbb{E}[f_{i_t}(w)] = \nabla f(w) \tag{۸.۳}$$

$$\mathbb{E}[\frac{1}{b} \sum_{i=1}^{n} \nabla f_{i_t}(w)] = \nabla f(w) \tag{۹.۳}$$

گرادیان نیمه انبوه با فاکتور $\frac{1}{b}$ گرادیان را کاهش می‌دهد اما به اندازه b که نشان دهنده مجموعه انتخابی از کل داده‌های ورودی است، موجب افزایش زمان محاسبات می‌شود.

۲.۳ گرادیان نزولی تصادفی با نگرش بازگشتی

نگیان و همکاران در سال ۲۰۱۷، الگوریتمی با نگرش بازگشتی معرفی کردند. الگوریتم گرادیان نزولی تصادفی بازگشتی[2]، پیاده‌سازی یک الگوریتم جدید است که ترکیبی از برخی ویژگی‌های مهم الگوریتمی، مانند SVRG همراه با یک محاسبه بازگشتی، می‌باشد. [۵۶].
هدف از ارائه این الگوریتم بهبود در روش‌های پیشین می‌باشد.
به طور خاص، در این الگوریتم قدم‌هایی در جهت شیب تصادفی گذاشته نمی‌شود، بلکه بیشتر در صدد تجمیع استفاده از اطلاعات گرادیان‌های تصادفی محاسبه شده گذشته با اطلاعات دقیق از محاسبه گرادیان لحظه به لحظه همراه با کاهش واریانس می‌باشد.
مهمترین بخش کلیدی این الگوریتم استفاده از یک مدل به‌روزرسانی بازگشتی جهت تخمین گرادیان تصادفی می‌باشد [۴۱].

$$v_t = \nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_{t-1}) + v_{t-1} \tag{۱۰.۳}$$

همانطور که در فرمول ۱۰.۳ مشاهده می‌شود، واریانس بر مبنای محاسبه دو گرادیان بدست می‌آید. طبق فرمول ابتدا گرادیان در دوره‌های t و t-1 محاسبه شده و سپس با واریانس در

[1] Mini-Batch Stochastic Gradient Descent
[2] StochAstic Recursive grAdient algoritHm(SARAH)

مرحله t-1 جمع می‌شود. به این ترتیب الگوریتم به صورت بازگشتی محاسبات را انجام می‌دهد. سپس طبق فرمول ۱۱.۳ به‌روزرسانی پارامتر مدل صورت می‌پذیرد:

$$w_{t+1} = w_t - \eta v_t \tag{۱۱.۳}$$

در واقع می‌توان این به‌روزرسانی را نیز با به‌روزرسانی پارامتر مدل در گرادیان نزولی تصادفی همراه با کاهش واریانس نیز مقایسه کرد و مشاهده می‌شود که در اینجا گرادیان در دوره t-1 محاسبه شده اما در روش کاهش واریانس مانند معادله ۱۲.۳ در حالت صفر بررسی می‌شود.

$$v_t = \nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_0) + v_0 \tag{۱۲.۳}$$

۱.۲.۳ فرضیات مسئله

در برخی مسائل بهینه‌سازی شروطی جهت بهینه‌کردن تابع هدف وجود ندارد. این مسائل را مسئله بهیته سازی بدون شرط می‌نامند. از این رو برای خطی‌سازی همگرایی در این توابع از شروط تحدب استفاده می‌شود. همان‌طور که بیان شد با توجه به فرمول (۱.۳) در واقع می‌توان گفت مسئله مورد بحث در اینجا یک مسئله بهینه سازی نامقید است. در واقع نامقید بودن مسئله بهینه سازی یعنی، شرطی که به موجب آن مسئله بهینه شود وجود ندارد. همچنین به جهت آن که بتوانیم مسئله را به صورت خطی حل کرده و همگرایی خطی حاصل شود شروطی را به عنوان شروط پایه قرار داده و در ادامه آنها را مورد بررسی قرار می‌دهیم:

فرض شماره ۱: تابع L ـ هموار

توابع هموار[1] به توابعی گفته می‌شود که هر مقدار از مشتق از تابع در دامنه آن تابع تعریف شده باشد. بنابراین در این فرض بودن تابع را فرض اول الگوریتم قرار داده شده است: فرض اولیه مبتنی بر L-Smooth بودن است:

هر تابع $f_i : \mathbb{R}^d \to \mathbb{R}$ و $i \in [n]$ را L-Smooth می‌نامیم اگر ثابتی مانند L وجود داشته باشد که از صفر بزرگتر باشد، آنگاه خواهیم داشت:

$$\|\nabla f_i(w) - \nabla f_i(w')\| \leq L\|w - w'\|, \quad \forall w, w' \in \mathbb{R}^d \tag{۱۳.۳}$$

رابطه (۱۳.۳) برقراری شرط هموار بودن تابع f_i را نشان می‌دهد که در این رابطه دو بردار w و w' عضو مجموعه d بعدی $\mathbb{R}$ می‌باشد. در واقع اگر نُرم میزان تفاضل گرادیان در دوره i برای

[1] Smooth

دو بردار w و w' کوچکتر مساوی نرم تفاضل دو بردار در ثابت L باشد بنابراین تابع هموار بوده و در نتیجه مشتقات تابع مورد نظر در دامنه تابع تعریف شده است.
این فرضیه همچنین بر این موضوع که $P(w) := \frac{1}{n}\sum_{i=1}^{n} f_i(w)$ یک تابع هموار تحت ثابت L است نیز دلالت دارد.

فرض شماره ۲: μـ محدب قوی

تابع $P : \mathbb{R}^d \to \mathbb{R}$ را μـ محدب قوی گویند یعنی اگر ثابتی مانند $\mu > 0$ وجود داشته باشد که در آن به ازای $\forall w, w' \in \mathbb{R}^d$ داشته باشیم:

$$P(w) \geq P(w') + \nabla P(w')^T (w - w') + \frac{\mu}{2}\|w - w'\|^2 \qquad (۱۴.۳)$$

بنابراین فرض μـ محدب قوی برای رابطه (۱.۳) جهت استحکام در الگوریتم مورد بررسی اعمال می‌گردد.
در واقع با قرار دادن شرط تحدب قوی مسئله در حالت خطی مورد بررسی قرار خواهد گرفت.

فرض شماره ۳:شرط تحدب

هر تابع $f_i : \mathbb{R}^d \to \mathbb{R}$ و $i \in [n]$ با برقراری شرط $\mu > 0$ محدب قوی خواهد بود.
می‌توان مشاهده کرد با فرضیه شماره ۲ معادله (۱.۳) یک پاسخ بهینه منحصر به فرد برای پارامتر مدل یادگیری ماشین w^* (پاسخ بهینه مسئله) بدست آورد که در نتیجه آن اگر تابع هدف شرایط مناسب را داشته باشد، خواهیم داشت:

$$2\mu[P(w) - P(w^*)] \leq \|\nabla P(w)\|^2 \qquad (۱۵.۳)$$

می‌توان این موضوع را خاطر نشان کرد که برای بهینه سازی مسائل محدب قوی مانند (۱.۳) در یادگیری ماشین از یک شرط حالت مانند $\kappa \stackrel{\text{def}}{=\!=} \frac{L}{\mu}$ استفاده می‌شود. در واقع پارامتر شرط حالت بدین منظور استفاده می‌شود که اگر ماتریس پارامتر مدل بدحالت باشد ممکن است در طول محاسبات خطای کوچک را بزرگ کند بنابراین با این ضریب از رخداد این مسئله جلوگیری می‌کنیم. توجه به این نکته اهمیت بسزایی دارد که شرط سوم دلالت بر شرط دوم دارد اما عکس این قضیه صادق نمی‌باشد.

فرض شماره ۴

هر تابع $f_i : \mathbb{R}^d \to \mathbb{R}$ و $i \in [n]$ محدب است، اگر:

$$f_i(w) \geq f_i(w') + \nabla f_i(w')^T(w - w') \quad \forall i \in [n] \tag{۱۶.۳}$$

در اینجا نیز شرط سوم بر شرط چهارم دلالت دارد، اما این موضوع برای شرط دوم و چهارم صدق نمی‌کند. بنابراین در مسئله مورد نظر برای دست یافتن به هدف مسئله همیشه فرض اول مورد نیاز است، اما داشتن تنها فرض چهارم، یا فرض دوم و چهارم با یکدیگر و یا داشتن فرض سوم برای حل مسئله کافی می‌باشد.
تجزیه و تحلیل پیچیدگی در دوره‌های این مسئله با هدف محدود کردن تعداد تکرارهای خارجی τ (یا تعداد کل ارزیابی های گرادیان تصادفی) که برای تضمین شرط $\|\nabla P(w_\tau)\|^2 \leq \epsilon$ مورد نیاز است، صورت می‌پذیرد. در این حالت خواهیم گفت w_τ یک راه حل دقیق با اندازه ϵ می‌باشد. با این حال، با توجه به اینکه به طور معمول برای الگوریتم‌های گرادیان تصادفی، هدف این است که تعداد تکرارها بدست آورده شود، برای تضمین محدودیت در حد معمول مربع پیش بینی شده از یک گرادیان مورد نیاز است.

$$\mathbb{E}[\|\nabla P(w_\tau)\|^2] \leq \epsilon \tag{۱۷.۳}$$

در واقع می‌توان گفت شرط (۱۷.۳) یک شرط توقف می‌باشد، که اگر خطای مسئله از میزان ϵ بیشتر باشد، تکرار در مسئله پایان می‌یابد.

۲.۲.۳ عملکرد الگوریتم گرادیان نزولی تصادفی بازگشتی

مهم‌ترین خاصیت الگوریتم گرادیان نزولی همراه با کاهش واریانس، روش استفاده شده در محاسبه گرادیان به صورت بازگشتی، جهت کاهش واریانس در هر مرحله است. این الگوریتم شامل دو حلقه تکرار تو در تو می‌باشد. این خاصیت با بالا رفتن تکرار در حلقه خارجی این الگوریتم بوجود می‌آید. در واقع می‌توان گفت، اگرتنها حلقه داخلی در این الگوریتم اجرا و حلقه خارجی حذف شود، واریانس کاهش نمی‌یابد.اما در الگوریتم گرادیان نزولی تصادفی بازگشتی SARAH در هر تکرار در حلقه بیرونی یک گرادیان کامل محاسبه می‌شود[۴۱].
به عبارت دیگر، اگر ما به راحتی حلقه داخلی را برای بسیاری از تکرارها اجرا کنیم (بدون در نظر گرفتن اجرای حلقه های بیرونی اضافی)، واریانس مراحل در مورد الگوریتم گرادیان نزولی همراه با کاهش واریانس، کاهش نمی‌یابد و این در حالی است که در مورد الگوریتم گرادیان

نزولی همراه با کاهش واریانس به صورت بازگشتی، واریانس به صفر می‌رسد.

جدول ۲.۳: شبه کد الگوریتم گرادیان نزولی تصادفی بازگشتی [۴۱]

	Algorithm SARAH
1	Initialize :$\tilde{w}_0$
2	Iterate
3	for $s = 1, 2, ...$ do
5	$w_0 = \tilde{w}_{s-1}$
6	$v_0 = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w_0)$
7	$w_1 = w_0 - \eta v_0$
8	Iterate
8	for $t = 1, ..., m-1$ do
9	Sample i_t uniformly at random from [n]
10	$v_t = \nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_{t-1}) + v_{t-1}$
11	$w_{t+1} = w_t - \eta v_t$
12	end for
13	Set $\tilde{w}_s = w_t$ with t chosen uniformly at random from $0, 1, ..., m$
14	end for

با توجه به جدول ۲.۳ می‌توان روند اجرای الگوریتم گرادیان نزولی تصادفی بازگشتی را مشاهده کرد. در این شبه کد این نکته حائز اهمیت است که کلید اصلی طراحی این الگوریتم تخمین گرادیان تصادفی جهت به‌روزرسانی بازگشتی می‌باشد.
در ابتدا مشاهده می‌شود که مقدار اولیه پارامتر مدل مقداردهی شده است. سپس در یک حلقه خارجی پارامتر مقدار $w_0 = \tilde{w}_{s-1}$ را برای هر تکرار حلقه خارجی بدست آورده، واریانس را در لحظه صفر محاسبه کرده و پارامتر مدل را نیز با $w_1 = w_0 - \eta v_0$ به‌روزرسانی می‌کند. در حلقه داخلی به ازای هر نمونه i که در دوره t به صورت تصادفی یکنواخت انتخاب می‌شود واریانس کاهشی را به صورت بازگشتی طبق فرمول $v_t = \nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_{t-1}) + v_{t-1}$ تخمین زده و به‌روزرسانی پارامتر مدل $w_{t+1} = w_t - \eta v_t$ را انجام می‌دهد.

الگوریتم SARAH از جهاتی مشابه الگوریتم SVRG است. زیرا هر دو دارای حلقه‌های بیرونی هستند که در هر تکرار بیرونی نیاز به محاسبه یک گرادیان کامل دارند و به دنبال آن در یک مرحله به محاسبه یک گرادیان کامل با نرخ یادگیری مشخص می‌پردازند. تفاوت این دو الگوریتم در حلقه داخلی است که در الگوریتم SARAH در حلقه داخلی با استفاده از یک فرمول بازگشتی واریانس را بدست آورده و پارامتر مدل را به‌روزرسانی می‌کند. در هر حلقه داخلی دو

گرادیان تصادفی محاسبه می‌شود، از این رو کل کار در هر تکرار حلقه خارجی از نظر تعداد محاسبه گرادیان‌ها از مرتبه $O(n+m)$ است. [۴۱]

۳.۳ سیستم توزیع‌شده

برای کارایی بهتر الگوریتم مورد نظر به دلیل وجود محاسبات پیچیده و زمان‌بر در داده‌هایی با مقیاس بالا، می‌توان الگوریتم مورد نظر را در یک سیستم توزیع‌شده پیاده‌سازی کرد. در گذشته، اصطلاحات موازی، همزمان و توزیع محاسباتی عمدتا مترادف در نظر گرفته می‌شدند. محاسبات موازی و توزیعی در یک نکته بسیار مهم متفاوت هستند. محاسبات موازی شامل فرآیندهایی با استفاده از حافظه مشترک می‌باشند. در حالی که در یک سیستم توزیع‌شده، هر گره حافظه محلی خود را دارد. در واقع در سیستم توزیع‌شده می‌توان وظایف را به قسمت‌های مختلف تقسیم کرد و هر کدام را به وظایف کوچکتری تبدیل کرده که در واقع این امر باعث می‌شود، وظایف به صورت موازی با هم اجرا شده و در کنار یکدیگر عملکرد سیستم را به صورت قابل توجهی بالا ببرند. عملکرد سیستم توزیع‌شده بدین صورت است که عموما پارامترهای یک مدل را در یک سیستم پارامتر سرور به وسیله یک خوشه[۱] به‌روزرسانی می‌کند. نودهای خوشه به سرور[۲] و کارگر[۳] تقسیم می‌شوند. گره‌های کارگر پارامتر را از گره سرور دریافت کرده، به‌روزرسانی را انجام داده و در نهایت برای تجمیع داده‌ها و بدست آوردن پارامتر نهایی مدل آن‌ها را به گره سرور بازمی‌گردانند. در ادامه به شرح الگوریتم به صورت تفصیلی پرداخته می‌شود.

تقسیم‌بندی داده‌ها

تقسیم‌بندی[۴] یکی از قابلیت‌های بانک اطلاعاتی است. در واقع می‌توان گفت در مجموعه داده‌هایی که حجم آن‌ها بسیار بالا می‌باشد، استفاده از تقسیم‌بندی می‌تواند، مجموعه داده مذکور را به ساختارهای جداگانه تقسیم کند. بنابراین سرعت اجرای الگوریتم‌ها جهت پردازش داده‌ها به شدت افزایش پیدا خواهد کرد.
در موازی سازی داده در مسئله رده‌بندی، مجموعه داده D به بلوک‌های P تایی تقسیم می‌شود. بلوک‌های داده‌ها به دستگاه‌های کارگر اختصاص داده می‌شوند که با $p = 1...P$ شاخص بندی می‌شوند. بلوک داده p امین را توسط D_p نشان داده می‌شود. موازی سازی داده‌ها برای

[1] Cluster
[2] Server
[3] Worker
[4] Data Partitioning

پارامترهای مدل با توجه به رابطه ۱۸.۳ صورت می‌گیرد:

$$w^s = G(w^s - 1, \sum_{p=1}^{P} \triangle(w^s - 1, D_p)) \quad (۱۸.۳)$$

در اینجا $\triangle(.)$ به معنای به‌روزرسانی پارامترهایی است که مطابق با تقسیم‌بندی داده D_p به گره کارگر داده می‌شود. به طور خاص، گره‌های کارگر با ارسال یک درخواست پارامترهای اولیه مسئله را از سمت سرور دریافت می‌کنند، پس از آن، پارامترها را به‌روزرسانی می‌کنند، یعنی مقدار $\triangle(w^s - 1, D_p)$ را محاسبه می‌کنند. $G(.)$ تجمیع[1] به‌روزرسانی‌های پارامترهای موجود در سرور را نشان می‌دهد. به طور ویژه، سرور تمام به‌روزرسانی‌ها را از سمت گره‌های کارگر جمع می‌کند. سپس عمل تجمیع را انجام می‌دهد. با در نظر گرفتن الگوریتم گرادیان نزولی تصادفی بازگشتی به عنوان نمونه، قانون به‌روزرسانی می‌تواند مانند رابطه (۱۹.۳) باشد:

$$w^s = w^s - 1 + \sum_{p=1}^{P} \triangle(w^s - 1, D_p) \quad (۱۹.۳)$$

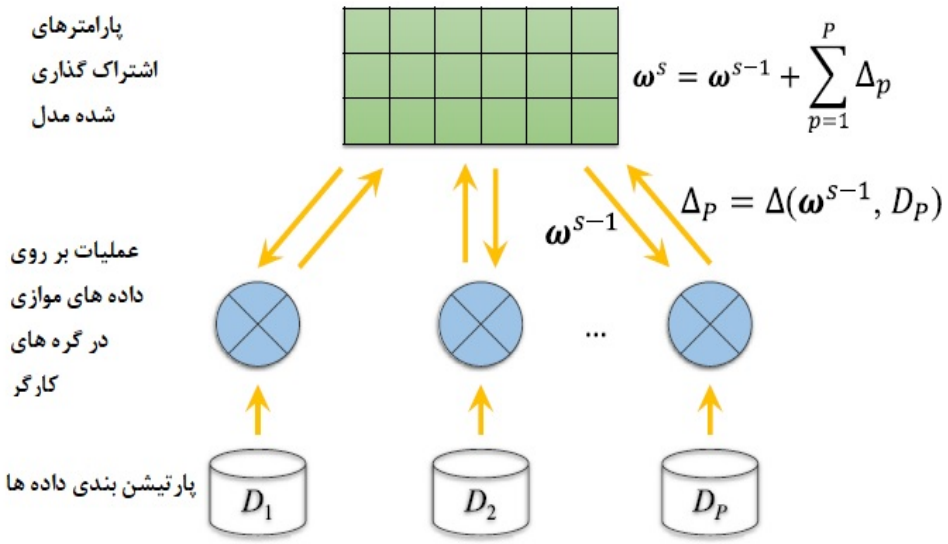
از آنجایی که $G(.)$ به عنوان یک برنامه جمع‌پذیر مورد بررسی قرار گرفته است، به طور معمول $\triangle(w^s - 1, D_p)$ به عنوان محصول نرخ یادگیری η و گرادیان با توجه به تقسیم‌بندی داده D مورد بررسی قرار می‌گیرند. گره‌های سرور و گره‌های کارگر از طریق یک توپولوژی دو طرفه با یکدیگر در تعامل هستند. پارامترهای مدل w را می‌توان در چندین سرور تقسیم و ذخیره کرد. بنابراین توسط یک حافظه دستگاه و تنها در یک سیستم پارامتر سرور محدود نمی‌شود. هر گره کارگر پارامترها را از سرور دریافت می‌کند، سپس به‌روزرسانی پارامترها را انجام می‌دهد. سرانجام، این به‌روزرسانی‌ها به سمت سرورها منتقل می‌شوند و روی آن سرورها جمع می‌شوند. سرورها می‌توانند در هنگام انجام یک کار یادگیری در مقیاس بزرگ، با گره‌های کارگر همکاری کنند تا از پردازنده سیستم در تمام دستگاه‌ها استفاده شود.

همان‌طور که در شکل ۲.۳ مشخص است، ابتدا داده‌ها به جهت موازی سازی، تقسیم‌بندی می‌شوند. در واقع این امر موجب می‌شود مجموعه داده D به بخش‌های مستقل‌تری تقسیم شود. با توجه به شکل هر قسمتی از مجموعه داده مورد استفاده جهت پردازش در الگوریتم به بخش‌های کوچکتری که آن را بخش می‌نامند از ۱ تا P تقسیم شده است.

سپس داده‌ها به طرف گره‌های کارگر فرستاده می‌شوند. در این قسمت داده‌های بخش بندی شده که هر کدام بخش کوچکی از مجموعه داده اولیه محسوب می‌شوند جهت پردازش و به‌روزرسانی پارامتر مدل یادگیری ماشین به سمت گره‌های کارگر رفته تا محاسبات انجام پذیرد.

در آخرین مرحله با توجه به شکل ۲.۳ بعد از به‌روزرسانی پارامترهای مدل یادگیری ماشین همراه با داده‌های بخش بندی شده درنهایت این پارامترها به سمت گره سرور فرستاده شده و در نهایت عمل تجمیع روی این پارامترها صورت می‌پذیرد تا به پارامتر کلی مسئله دست یافته

[1] Aggregate



شکل ۲.۳: مراحل استفاده از داده موازی سازی شده

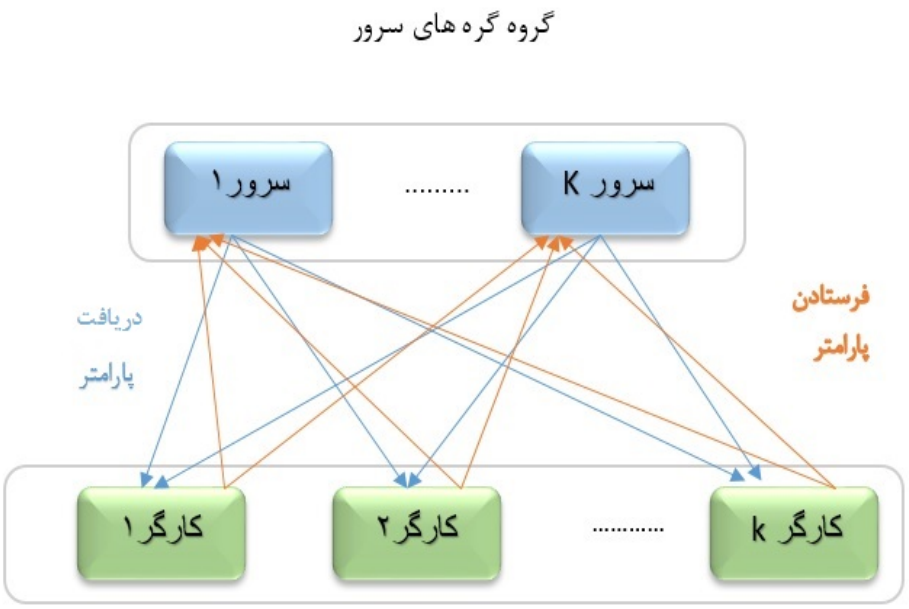
شود.

در شکل یک توپولوژی از ساز و کار گره‌های کارگر با گره سرور به تصویر کشیده شده است. در واقع می‌توان گفت تبادل اطلاعاتی به صورت متقابل بین هر دو گره قابل مشاهده است.

با توجه به شکل ۲.۳ می‌توان این‌گونه شرح داد که پارامترهای مدل یادگیری ماشین از سمت گره‌های سرور به سمت گره‌های کارگر فرستاده[1] می‌شود (این امر به درخواست گره‌های کارگر صورت می‌پذیرد) و گره‌های کارگر پس از دریافت داده‌های بخش بندی شده و همچنین دریافت پارامترهای مدل از گره‌های سرور عملیات به‌روزرسانی پارامترها را انجام می‌دهند و در نهایت پارامترهای به‌روزرسانی شده به سمت گره‌های سرور فرستاده[2] می‌شوند. در نتیجه می‌توان بیان کرد که تعاملی متقابل بین این دو گروه از گره‌ها درون یک خوشه در یک سیستم توزیع‌شده وجود دارد.

همانطور که در شکل ۳.۳ نیز قابل مشاهده می‌باشد، هم از سمت گره‌های سرور و هم از طرف گره‌های کارگر تبادل پارامتر با یکدیگر دارند. در واقع گره کارگر پارامتر مدل را از طرف گره سرور دریافت می‌کند[3] و بعد از انجام محاسبات به‌روزرسانی مورد نظر مسئله، گره کارگر اصطلاحا پارامترهای مدل را به سمت گره سرور می‌فرستد[4]، در واقع پارامترها را ارسال می‌کند.

[1] Pulling Parameters
[2] Pushing Parameters
[3] Pull
[4] Push



شکل ۳.۳: توپولوژی گره‌ها

۴.۳ فرآیند به‌روزرسانی الگوریتم

پس از شرح الگوریتم گرادیان نزولی تصادفی بازگشتی به طور کامل و همچنین نحوه عملکرد سیستم‌های توزیعی بر روی بهینه سازی گرادیان نزولی به شرح نحوه پیاده‌سازی الگوریتم توضیح داده شده بر روی سیستم توزیعی می‌پردازیم. در ادامه پارامترهای مدل و شبه کد الگوریتم را مورد بحث و بررسی قرار می‌دهیم.

۱.۴.۳ پارامترهای مدل الگوریتم پیشنهادی

با توجه به الگوریتم ارائه شده، پارامترهایی که در الگوریتم استفاده شده‌اند را باید به دقت مورد بررسی قرار داد تا با نحوه اجرای الگوریتم پیشنهادی آشنا شد، بدین منظور در این قسمت پارامترهای مدل را در قالب جدول۳.۳ شرح داده شده است:

جدول ۳.۳: شرح پارامترهای مدل پیشنهادی

پارامتر	توضیحات
w	پارامتر جهانی مدل
i_t	شاخص نمونه غیر یکنواخت در دوره t ام که به صورت تصادفی انتخاب می‌شود
∇f_{i_t}	گرادیان تصادفی
v_t	واریانس دوره t
η	نرخ یادگیری
μ	نرخ تحدب
L	ضریب همرفتی
s	تعداد تکرار حلقه خارجی
m	تعداد تکرار حلقه داخلی
n	تعداده داده‌های آموزشی
t	تعداد تکرار مورد نیاز برای الگوریتم
w^*	پاسخ بهینه پارامتر مدل
$P(w)$	تابع هدف
$R(w)$	آیتم تنظیم

۲.۴.۳ شبه کد الگوریتم پیشنهادی

با توجه به چالش‌های موجود در الگوریتم‌های بهینه سازی مانند محاسبات پیچیده و زمان‌گیر و با عنایت به کاربردهای مفید سیستم توزیعی در محاسبات روی مجموعه داده‌هایی با حجم بالا الگوریتم ارائه شده به نحوه پیاده‌سازی یک روش بهینه سازی پر کاربرد در یک سیستم توزیعی می‌پردازد. روش ارائه شده در واقع یک الگوریتم گرادیان نزولی تصادفی بازگشتی در یک سیستم توزیع‌شده[1] است.

با توجه به جدول ۵.۳ هر تکرار داخلی دو گرادیان تصادفی را ارزیابی می‌کند و از این رو کل کار در هر تکرار بیرونی از نظر تعداد ارزیابی‌های گرادیان برابر o(n+m) است. سرور این پارامترهای یادگیری شده را دریافت می‌کند و آنها را تجمیع می‌کند. پارامترهای جمع شده آخرین پارامترهای جهانی می‌باشند که برای تکرار بعدی به کارگران ارسال می‌شوند. جزئیات ارتباط بین گره‌ها و جمع شدن پارامترها در ادامه ذکر خواهد شد.

[1] Distributed Stochastic Recursive grAdient AlgoritHm(DSARAH)

جدول ۴.۳: شبه کد الگوریتم گرادیان نزولی تصادفی بازگشتی توزیع‌شده

	Algorithm DSARAH
1	Initialize :$\tilde{w_0}$ Initialize parameter in server and Pull the global parameters by all workers from the servers
2	Iterate
3	for $s = 1, 2, ...$ do update parameters in workers
5	$w_0 = \tilde{w}_{s-1}$
6	$v_0 = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w_0)$
7	$w_1 = w_0 - \eta v_0$
8	Iterate
8	for $t = 1, ..., m-1$ do
9	randomly sample $i_t \in \{0, 1, ..., n\}$
10	$v_t = \nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_{t-1}) + v_{t-1}$ variance reduced gradient
11	$w_{t+1} = w_t - \eta v_t$ update the parameters with variance reduction gradient
12	end for
13	$\tilde{w}_s = w_t$ push the newly learned parameters to the servers for aggregation

جدول ۵.۳: توضیح شبه کد الگوریتم

۱	$\tilde{w_0}$ پارامتر اولیه مدل در سرور مقدار دهی اولیه می‌شود و با یک درخواست از طرف گره‌های کارگر از گره سرور به سمت گره کارگر فرستاده می‌شود
۲	در یک حلقه خارجی در هر دوره یک گرادیان کامل محاسبه شده و واریانس مورد محاسبه قرار می‌گیرد
۳	$w_0 = \tilde{w}_{s-1}$مقدار دهی پارامتر مدل
۴	$v_0 = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w_0)$ محاسبه یک گرادیان برای واریانس
۵	$w_1 = w_0 - \eta v_0$ به‌روزرسانی مدل
۶	در یک حلقه داخلی به ازای m بار تکرار
۷	انتخاب یک شاخص غیر یکنواخت i_t به صورت تصادفی
۸	تفریق دو گرادیان در دو دوره t و t-1 و جمع آنها با واریانس در دوره t-1 به صورت بازگشتی محاسبات انجام می‌پذیرد $v_t = \nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_{t-1}) + v_{t-1}$
۹	$w_{t+1} = w_t - \eta v_t$ به‌روزرسانی پارامتر مدل
۱۰	اتمام حلقه داخلی
۱۱	$\tilde{w}_s = w_t$ فرستاددن پارامتر جدید مدل از طرف گره کارگر به گره سرور جهت تجمیع
۱۲	اتمام حلقه خارجی

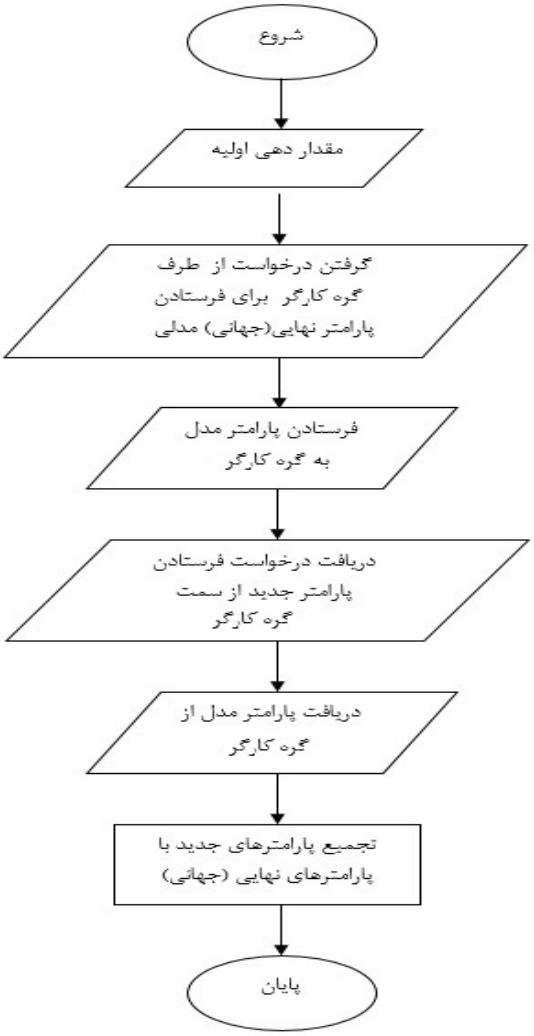
۱.۲.۴.۳ گره سرور

اینجا مقدار پارامتر سراسری یا جهانی[1] را نگه‌داری می‌کنیم. اگر درخواستی از سمت کارگر فرستاده شود این پارامتر را جهت به روز رسانی به کارگر فرستاده و بعد از آن که به‌روزرسانی

[1]Global Parameter

توسط کارگر به پایان رسید، سرور با توجه به درخواست جهت بازگرداندن پارامتر سراسری (جهانی)، مقدار جدید را دریافت کرده و به عنوان پارامتر سراسری (جهانی) جدید مدل قرار می‌دهد.

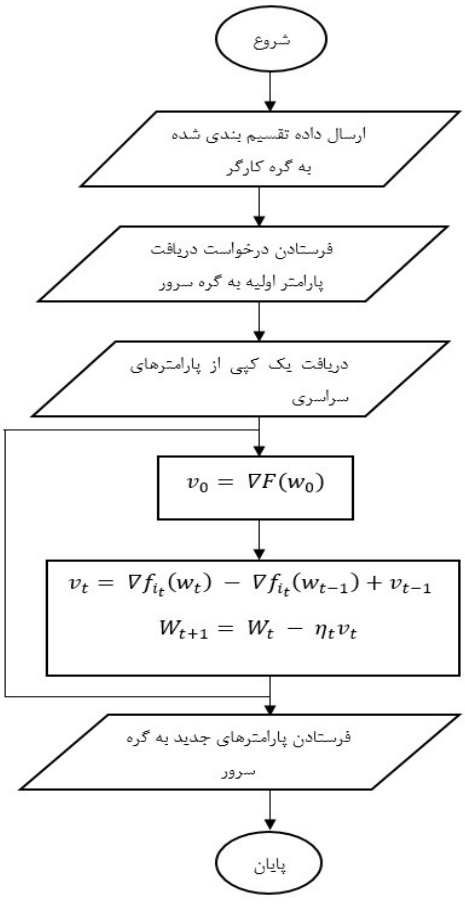
گره سرور در این سیستم به تکرارهای الگوریتم نظارت دارد. همان‌طور که در شکل ۴.۳ مشاهده می‌شود، ابتدا مقدار اولیه پارامتر مدل مسئله رده‌بندی $\tilde{w}_0$ مقدار دهی می‌شود، سپس درخواستی از سمت گره کارگر به سمت گره سرور فرستاده می‌شود که در آن پارامتر سراسری را درخواست می‌کند، سرور پارامتر را به گره کارگر می‌فرستد. بعد از انجام عملیات در گره کارگر درخواستی از سمت گره کارگر برای فرستادن پارامتر به‌روزرسانی شده به سمت گره سرور فرستاده می‌شود. در نهایت گره سرور پارامترهای دریافت شده از گره های کارگر را تجمیع کرده و برابر پارامتر سراسری یعنی $\tilde{w}^s = w_m$ قرار می‌دهد.



شکل ۴.۳: عملکرد Server

۲.۲.۴.۳ گره کارگر

در این قسمت ابتدا یک درخواست از گره کارگر به سمت سرور فرستاده می‌شود و پارامتر سراسری مدل را که با عنوان پارامتر محلی از آن یاد می‌شود، دریافت می‌کند. سپس در دو حلقه بیرونی و داخلی به‌روزرسانی را انجام می‌دهد. در حلقه خارجی پارامترهای دریافت شده را گرفته و ابتدا آن را به عنوان پارامتر اولیه مدل $w_0 = \tilde{w}_{s-1}$ مقدار دهی کرده. سپس میانگین گرادیان در ابتدای دوره را برای همه داده‌ها محاسبه کرده و آن را برابر مقدار واریانس اولیه قرار داده ($v_0 = \frac{1}{n}\sum_{i=1}^{n} \nabla f_i(w_0)$) و اولین پارامتر مدل را در اولین دوره به‌روزرسانی می‌کند ($w_1 = w_0 - \eta v_0$). سپس در حلقه داخلی به ازای هر نمونه i در دوره t واریانس هر دوره را طبق فرمول $v_t = \nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_{t-1}) + v_{t-1}$ در هر دوره بدست آورده و پارامتر مدل را در فرمول $w_{t+1} = w_t - \eta v_t$ به‌روزرسانی کرده و در نهایت درخواستی جهت فرستادن پارامتر محلی به‌روزرسانی شده جدید به عنوان پارامتر سراسری (جهانی) به سمت سرور را خواهد داشت. در نهایت عمل تجمیع این داده‌های به‌روزرسانی شده در سرور صورت می‌پذیرد و گره سرور پارامتر سراسری (جهانی)جدید را جایگزین می‌کند.



شکل ۵.۳: عملکرد Worker

۳.۴.۳ عملیات بر روی پارامترهای مدل

در این سیستم توزیع‌شده پارامترها در دو بخش تجمیع و پخش روی آن‌ها که به تفکیک در قسمت‌های بعد مورد بررسی قرار می‌گیرند، تقسیم می‌شود.

۱.۳.۴.۳ تجمیع پارامتر مدل

در سیستم توزیع‌شده به جهت بدست آوردن پارامتر سراسری در هر حلقه داخلی هر کدام از گره ها پارامتر مدل را برای نمونه iام محاسبه می‌کنند و در نتیجه همه آنها مقدار بدست آمده از به‌روزرسانی iاُمین نمونه را به سرور داده تا به جهت بدست آوردن پارامتر سراسری میانگین پارامترهای محلی بدست آمده را برای دوره t را محاسبه کند به این عمل تجمیع گفته می‌شود وقتی گره‌های کارگر پارامترهای تازه یادگیری شده خود را به سمت سرورها فرستاده می‌شوند، این پارامترها با پارامترهای سراسری روی سرورها تجمیع[1] می‌شوند. میانگین بین پارامترهای تازه آموخته شده و پارامترهای جهانی به عنوان آخرین پارامترهای سراسری شناسایی می‌شوند و منتظر خواهند بود که برای تکرار بعدی به طرف تمام گره‌های کارگر سوق داده شوند.

۲.۳.۴.۳ پخش پارامتر مدل

ارتباطات معمولاً بین گره سرور و یک گره کارگر وجود دارد. هیچ ارتباطی بین گره‌های سرور وجود ندارد. به طور خاص، پارامترها به طور یکنواخت تقسیم می‌شوند و در چندین گره‌های سرور ذخیره می‌شوند. بنابراین، اگر یک گره کارگر باید پارامترها را دریافت کند یا به‌روزرسانی ها را به طرف گره سرور روانه کند، با گره‌های سرور مربوطه ارتباط برقرار می‌کند و اصطلاحا داده‌ها پخش[2] می‌شوند. از آنجا که در این سیستم داده ها به صورت قسمت‌بندی شده به هر گره کارگر فرستاده می‌شود و هیچ پارامتر مشترکی بین گره‌های سرور مختلف وجود ندارد، گره‌های سرور نیازی به برقراری ارتباط با یکدیگر ندارند.

۵.۳ محیط پیاده‌سازی

[1] Aggregate
[2] Broadcast

در برنامه نویسی موازی، یک برنامه در یک زمان می‌تواند چند کار و وظیفه را با هم انجام دهد. تا قبل از این معمولاً برنامه‌های موازی بر روی یک کامپیوتر اجرا می‌شد، این در حالی است که مدل برنامه نویسی نگاشت_کاهش برای این به وجود آمده است که تا حد امکان کارها و وظایفی را که می‌توانند به صورت موازی با هم اجرا شوند، در کامپیوترهای مختلف به صورت موازی اجرا کند.

این اجرای موازی می‌تواند بر روی کامپیوترهای ساده مختلف که با هم ارتباط دارند انجام شوند. یعنی به جای این که از یک ابر کامپیوتر استفاده کنند، می‌توانند از مجموعه‌ای از چندین کامپیوتر تقریبا ساده و نه چندان گران جهت عملیات ذخیره سازی و محاسبات مختلف استفاده کنند. ذخیره سازی بر روی این کامپیوترها در بسیاری از مواقع توسط سیستمی به نام سیستم فایل توزیع‌شده[۱] انجام می‌شود. در واقع اطلاعات در این سیستم توزیع‌شده، بر روی کامپیوترهای مختلف قرار می‌گیرند و همچنین بر روی این کامپیوترها تکرار می‌شوند تا اگر گاهاً یک کامپیوتر خراب شد یا دسترسی به آن امکان‌پذیر نبود، داده‌ها مفقود نشود و بتوان داده‌ها را از کامپیوترهای مختلف دیگر فراخوانی کرد و از آن‌ها در محاسبات استفاده نمود. بر روی یک همچنین سیستم فایل توزیع‌شده‌ای، مدل‌های برنامه نویسی ساده‌ای مانند نگاشت_کاهش فراهم شده است تا بتوان با کمک آن‌ها، محاسبات مختلف بر روی حجم عظیمی از داده‌ها را در خوشه (مجموعه‌ای از کامپیوترها) پیاده‌سازی کرد. در سیستم‌های توزیع‌شده، گره‌ها یا همان کامپیوترهای مختلف با هم همکاری کرده و یک سیستم واحد را (از دید کاربر استفاده کننده) به وجود می‌آورند.
الگوریتم‌های توزیع‌شده برای برنامه نویسی سیستم های توزیع‌شده طراحی شده‌اند. آنها با الگوریتم‌های متمرکز تفاوت دارند زیرا از هر حالت سراسری یا یک چارچوب زمانی سراسری بی‌اطلاع هستند.

۱.۵.۳ آپاچی اسپارک

برای پردازش کلان داده در سیستم‌های توزیع‌شده، سامانه پردازش جدیدی پا به عرصه وجود نهاده است با نام اسپارک که امروزه یکی از فعال‌ترین پروژه های بنیاد آپاچی است. تا قبل از اسپارک، برای پردازش حجم عظیم داده‌ها از هدوپ [۲] به صورت معمول استفاده می‌شود. در مدل پردازشی هدوپ، داده‌ها در کل شبکه یا کلاستر توسط سیستم فایل[۳] توزیع می‌شوند و برای پردازش داده‌ها از مکانیزم نگاشت و تجمیع[۴] استفاده می‌شود. یعنی پردازش مورد نیاز

[1] Distributed File System
[2] Hadoop
[3] Hadoop Distributed File System (HDFS)
[4] MapReduce

بر روی داده‌ها مثلا آمارگیری یا یافتن یک الگوی خاص در متن به هر گره محاسباتی ارسال می‌شود. عمل نگاشت[۱] بر روی هر سیستم فایل‌های داده‌ای پردازش را آغاز می‌کند و نتایج را به شکل استاندارد درآورده و به یک گره محاسباتی دیگر برای تجمیع[۲] ارسال می‌کند. این عمل با تجمیع تمام نتایج در یک گره و نمایش آن به کاربر به پایان می‌رسد. این مدل، برای بیش از ده سال، رایج‌ترین مدل پردازشی کلان داده و مبتنی بر اکوسیستم هدوپ بوده است. مدل پردازشی نگاشت و تجمیع با وجود مزایای فراوانی که دارد از جمله مقیاس پذیری، تحمل خطا و مدل ساده پردازشی برای تولید برنامه و پردازش داده، دارای دو عیب عمده است: اول اینکه این سیستم برای کار با دیسک طراحی شده است و تمام نتایج در مراحل مختلف باید در دیسک ذخیره شوند که خود سرعت پردازش را بسیار پایین می‌آورد و دوم اینکه توابع آماده آن بسیار محدود هستند و بار اصلی تولید برنامه و پردازش داده بر عهده برنامه نویسان آشنا به مکانیزم نگاشت و تجمیع است.

در پردازش اسپارک تمرکز بر روی انجام محاسبات درون حافظه‌ای است یعنی تا حد امکان و با وجود ظرفیت حافظه داخلی دستگاه، محاسبات درون حافظه انجام می‌پذیرد. این امر باعث می‌شود سرعت پردازش داده‌ها نسبت به هدوپ معمولی در پردازش های دیسک محور تا ده برابر و در پردازش های درون حافظه ای تا صد برابر افزایش پیدا کند که خود بهبود بسیار زیادی را نشان می‌دهد و برای الگوریتم‌های تکرار شونده بسیار عالی عمل می‌کند.

۲.۵.۳ معماری اسپارک

اصلی‌ترین مولفه اسپارک برنامه راه انداز[۳] است که اجرای عملیات بر روی مجموعه داده‌های توزیع‌شده انعطاف پذیر[۴] را بین شبکه توزیع کرده و نتایج را دریافت می‌کند. نقطه قوت اصلی اسپارک استفاده از مجموعه داده‌های توزیع‌شده انعطاف پذیر یا RDD است. تمام داده‌ها در اسپارک برای پردازش باید به شکل RDD درآیند که البته به کمک توابع خود اسپارک این امر به راحتی امکان پذیر است. RDD ها فقط خواندنی هستند و با هر تراکنش جدیدی که روی یک مجموعه داده برگشت پذیر انجام می‌شود یک RDD جدید ساخته می‌شود و محاسبات با این مجموعه جدید ادامه پیدا می‌کند.

دو نوع کار می‌توان روی این مجموعه داده‌ها انجام داد. ابتدا تبدیلات[۵] هستند. عمل

[1] Map
[2] Reduce
[3] Driver
[4] Resilient Distributed Datasets(RDD)
[5] Transformation

تبدیل[1] یک RDD را به یک RDD جدید تبدیل می‌کند. مانند فیلتر کردن و انجام یک تابع سراسری روی تک تک عناصر (Map) و سپس عملیات صورت می‌پذیرد. منظور از عملیات، توابعی است که روی یک RDD اعمال می‌شود و یک مقدار را بر می‌گرداند. مثلا شمارش عناصر یا یافتن بیشینه یا کمینه عناصر می‌باشد. نکته مهم در مورد اسپارک این است که هر RDD شامل اشاره به مجموعه داده پدر خود به همراه عمل انجام گرفته برای تبدیل را داراست و برگشت پذیر بودن این مجموعه‌ها هم دقیقاً به همین روال اشاره دارد چون با داشتن مجموعه داده اولیه و مجموعه تبدیلات انجام گرفته روی آن، می‌توان به راحتی یک مجموعه را از پایه دوباره ساخت و اگر سیستم به هر دلیلی مجموعه داده فعلی خود را از دست داد، به راحتی آن را با پیمایش زنجیره تبدیلات از اولین مجموعه تا الان، می‌تواند بازیابی کند. بنابراین RDD ها برگشت پذیر هستند. نکته دیگر در باره RDD ها اینکه تا عملیاتی روی این مجموعه ها صورت نگیرد (توابع عملیاتی صدا زده نشوند)عملاً تبدیلی هم انجام نمی‌شود یعنی تبدیل‌ها زمانی انجام می‌شوند و مجموعه‌های جدید را تولید می‌کنند که یکی از توابع عملیاتی روی آنها صدا زده شوند. برنامه راه انداز از طریق مولفه مدیریت کلاستر، با رایانه ها و گره‌های محاسباتی شبکه ارتباط برقرار می‌کند. هر گره محاسباتی که به آنها گره های کارگر[2] گفته می‌شود، از دو بخش مدیریت حافظه و اجراکننده تشکیل شده است که اجراکننده، وظیفه انجام پردازش‌ها را روی RDDها برعهده دارد.

در سطوح بالا، هر برنامه اسپارک شامل یک برنامه درایور است که عملکرد اصلی کاربر را انجام داده و عملیات موازی مختلفی را روی یک خوشه اجرا می‌کند. انتزاع اصلی اسپارک یک مجموعه داده توزیع‌شده انعطاف پذیر است، که مجموعه‌ای از عناصر تقسیم شده در گره‌های خوشه می‌باشند و می‌توانند به‌صورت موازی عملیاتی را انجام دهند. این مجموعه داده‌ها با ایجاد یک پرونده در سیستم فایل هدوپ یا یک مجموعه موجود اسکالا[3] در برنامه درایور شروع شده و آن را تبدیل می‌کنند. همچنین ممکن است کاربران از اسپارک بخواهند تا RDD را در حافظه نگه دارد و به آن امکان استفاده مجدد از کارآیی موازی را بدهد. در نتیجه، RDDها می‌توانند خرابی گره‌ها را بطور خودکار بهبود دهند.
انتزاع دوم در اسپارک متغیرهای مشترک است که می‌توانند در عملیات موازی استفاده شوند. به طور پیش فرض، هنگامی که اسپارک یک تابع را به صورت موازی به عنوان مجموعه ای از کارها روی گره های مختلف اجرا می‌کند، یک نسخه از هر متغیر مورد استفاده در عملکرد را به هر کار ارسال می‌کند. بعضی اوقات، یک متغیر باید در بین کارها یا بین کارها و برنامه درایور مشترک شود.

[1] Action
[2] Worker Nodes
[3] Scala

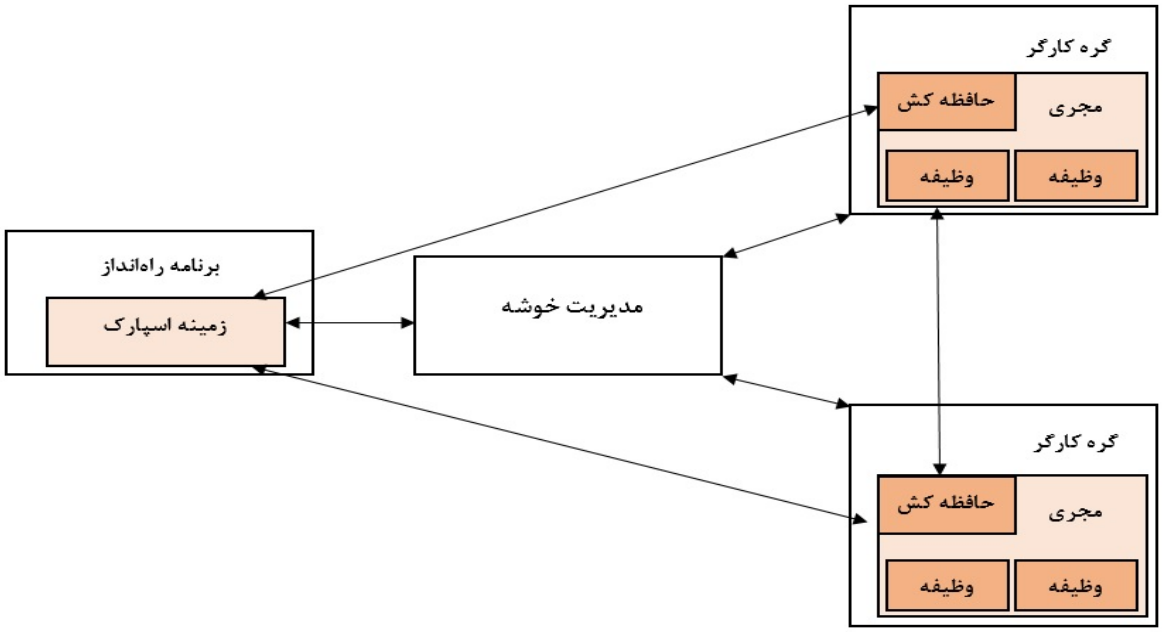
اسپارک از دو نوع متغیر مشترک پشتیبانی می‌کند: متغیرهای پخش[۱]، که می‌توانند برای ذخیره یک مقدار حافظه در همه گره‌ها، و جمع کننده‌ها[۲]، به عنوان متغیرهایی باشند که فقط به آنها افزوده میشود. با توجه به شکل ۶.۳ گره راه‌انداز کار اسپارک را به مدیریت خوشه ارسال می‌کند. ابتدا زمینه اسپارک[۳] را که با مدیریت خوشه، منابع خوشه (گره های کارگر) در آن هماهنگ شده، ایجاد کرده و در پاسخ به مدیریت خوشه گره‌های کارگر را به خوشه اختصاص داده می‌شود. زمینه اسپارک یا همان برنامه درایور اجرا کننده‌ها را روی گره های کارگر راه اندازی می‌کند. چندین اجرا کننده می‌تواند بر روی گره‌های کارگر راه‌اندازی شوند. سپس هر اجرا کننده (مجری) چندین کار را اجرا می‌کند که همزمان انجام می‌شود. مشاهدات نشان می‌دهد که هر اجرا کننده باید پیکر بندی شود تا حداکثر پنج وظیفه را همزمان انجام دهند، در غیر این صورت اختلاف بین وظایف باعث کاهش کار کلی اسپارک می‌شود. برنامه درایور اسپارک نیاز دارد که یک کپی از متغیرها به هر کدام از وظایف بفرستد. زمانی که یک وظیفه در اجرا کننده تولید می‌شود برنامه درایور اسپارک نیاز به فرستادن یک کپی از داده به آن وظیفه را دارد.

حال هزاران وظیفه را درنظر بگیرید که در سراسر خوشه در حال اجراست. در این صورت درایور اسپارک وظیفه دارد یک کپی از متغیر را به تمامی وظایف به صورت جداگانه بفرستد. این کپی به هر وظیفه‌ای که بر روی نودهای کارگر مشغول اجرا است، فرستاده می‌شود. برای به اشتراک گذاشتن متغیر در میان گره های یک خوشه به صورت کارآمد نیاز است که یک کپی به هر گره کارگر یک بار ارسال شود و سپس زمانی که کار جدید آمد و به وسیله مجریان راه‌اندازی شد، آنها از داده قابل دسترس به صورت محلی بهره ببرند. به همین جهت برای ارسال این داده‌ها اسپارک از پخش کننده استفاده می‌کند، به این صورت که درایور اسپارک یک کپی تنها از داده پخش کننده به هر گره کارگر می‌فرستد و زمانی که وظایف به وسیله مجریان تولید می‌شوند، هر یک از وظایف کپی از نود کارگر محلی برای خود برمی‌دارد. این بدان معنی است که اگر دو نود کارگر در کارهای اسپارک ایجاد شود، بعد از آن برنامه درایور باید یکبار یک داده به هر نود بفرستد و دیگر اهمیتی ندارد که چه تعداد وظیفه روی آن نودها تولید شود، داده به صورت محلی قابل دسترس است.

این کار باعث کاهش ترافیک شبکه شده و کارآیی سیستم توزیع‌شده را در اشتراک گذاری داده در میان خوشه ها بالا می‌برد. یک نکته مهم در مورد پخش‌کننده این است که تنها خواندنی می‌باشد. وظایف نمی‌توانند محتوای داده پخش‌کننده را تغییر دهند. متغیر اشتراک گذاری مهم دیگر جمع‌کننده است. این یک گونه از متغیرهای افزایشی است، در واقع می‌توان

[1] Broadcast
[2] Accumulator
[3] Spark Context

گفت وظایف در حال اجرا بر روی ماشین‌های مختلف می‌توانن مقدار خود را افزایش دهند و این اطلاعات جمع شده تنها در برنامه درایور به حالت فقط خواندنی قابل دسترسی است. در واقع جمع کننده در میان نودها عمل نوشتن و پخش کننده عمل خواندن را انجام می‌دهد. در واقع می‌توان نتیجه گیری کرد که سامانه پردازشی اسپارک با استفاده از زبان برنامه نویسی اسکالا قادر به ایجاد یک سیستم توزیع‌شده است که تنها با تعریف کردن زمینه اسپارک در قسمت اصلی[1] برنامه و با استفاده از توابع توزیعی تعریف شده در آپاچی اسپارک سیستم توزیع‌شده را راه اندازی می‌کند.



شکل ۶.۳: معماری اسپارک

۶.۳ جمع بندی

ابتدا در این فصل به بررسی مسئله گرادیان نزولی تصادفی به صورت اختصاصی و با پارامترهای مختص به الگوریتم پیشنهادی پرداخته شد. با توجه به مشکلات پیش رو در الگوریتم گرادیان نزولی تصادفی اعم از کندی در همگرایی و واریانس بالا به جهت نوسانات الگوریتم برای رسیدن به کمینه تابع هدف به بررسی الگوریتمی با نام گرادیان نزولی تصادفی بازگشتی پرداخته، که این الگوریتم یک روش جدید در راستای کاهش واریانس در الگوریتم گرادیان نزولی است که به جهت رویکرد بازگشتی که در محاسبات گرادیان‌ها در هر دوره دارد و همچنین فرضیات اولیه مورد استفاده در آن از روش گرادیان نزولی تصادفی همراه با کاهش

[1] Main Program

واریانس تحت شرایط تحدب قوی عملکرد بهتری خواهد داشت. اما به دلیل عدم پشتیبانی این الگوریتم و همچنین الگوریتم گرادیان نزولی تصادفی همراه با کاهش واریانس از سیستم‌های توزیعی، به دلیل طراحی آن‌ها که تنها بر روی یک گره پیاده‌سازی می‌شوند، در این پژوهش الگوریتمی پیشنهاد شد که روش مذکور در یک سیستم توزیعی پیاده‌سازی شود. بنابراین در این روش با استفاده از گره‌های کارگر و سرور، در یک سیستم پارامتر سرور الگوریتم گرادیان نزولی تصادفی بازگشتی پیاده‌سازی شد و توسط این سیستم پارامترهای این مدل در گره‌های کارگر به‌روزرسانی شده و در نهایت به جهت تجمیع به سرور فرستاده شده و پارامتر سراسری بدست آورده می‌شود. با توجه به رویکرد پارامتر سرور این سیستم‌ها و تقسیم‌بندی داده‌های ورودی برای فرستادن به گره‌های کارگر سرعت پردازش در مسئله رده‌بندی به مراتب بالاتر از حالت ساده الگوریتم می‌باشد. به جهت بررسی بیشتر در بخش چهارم به نتایج بدست آمده پرداخته می‌شود.

فصل ۴

پیاده‌سازی و ارزیابی روش پیشنهادی

۱.۴ مقدمه

در این بخش از پژوهش، نتایج آزمایشات و نحوه ارزیابی الگوریتم ارائه شده در فصل قبل را ارائه و مورد بررسی قرار خواهد گرفت. ابتدا به شرح مسئله مورد استفاده در ارزیابی الگوریتم پرداخته می‌شود. سپس محیط پیاده‌سازی الگوریتم پیشنهادی مورد بررسی قرار می‌گیرد. سپس نتایج بدست آمده بر روی پایگاه داده‌های مختلف را ارائه کرد و در نهایت به مقایسه الگوریتم پیشنهادی با الگوریتم‌های مرسوم پرداخت می‌شود.

۲.۴ نحوه ارزیابی الگوریتم پیشنهادی

جهت ارزیابی الگوریتم پیشنهادی گرادیان نزولی تصادفی بازگشتی در یک سیستم توزیع‌شده نیاز به یک مسئله کاربردی بسیار ضروری است. به جهت ارزیابی الگوریتم پیشنهادی آن را در یک مسئله رده‌بندی مورد بررسی قرار داده شده است. ارزیابی روش پیشنهادی از چند منظر می‌تواند مورد بررسی قرار گیرد. ابتدا می‌توان بیان کرد، روش بهینه‌سازی به جهت آن که در یک سیستم توزیع‌شده پیاده‌سازی می‌شود، انتظار می‌رود از سرعت بالایی برخوردار باشد.

همچنین برای پردازش‌های پیچیده در حجم بالای داده، محاسبات را با کارایی بهتری اجرا می‌کنند.
بنابراین می‌توان گفت این الگوریتم را در دو حالت می‌توان محک زد. حالت اول در حجم بالای داده است که سرعت بالا را تضمین کند. همچنین در حالت دوم کاربرد الگوریتم در مسائل مهم یادگیری ماشین است که بتوان کارایی الگوریتم را در علوم پایه اثبات کرد. یکی از این علوم پایه رده‌بندی [۱] داده‌ها می‌باشد که در ادامه به شرح آن می‌پردازیم.

۳.۴ رده بندی رگرسیون لجستیک

اغلب برای بیان شدت رابطه خطی بین دو متغیر کمی، از ضریب همبستگی [۲] استفاده می‌شود. همچنین برای نمایش مدل رابطه بین دو متغیر نیز از مدل رگرسیونی کمک گرفته می‌شود. در این میان یک الگو برای پیش‌بینی متغیر وابسته (Y) براساس متغیر مستقل (X) ایجاد می‌شود. ولی باید توجه داشت که در مدل ایجاد شده، هر دو متغیر مستقل و وابسته، کمی هستند. همچنین شرط پیوسته بودن این مقدارها نیز در روش رگرسیون نهفته است. ولی ممکن است بخواهیم رابطه بین یک متغیر مستقل (با مقدارهای پیوسته) را با یک متغیر وابسته با مقدارهای کیفی بسنجیم. در این حالت روش عادی رگرسیون خطی پاسخگو نخواهد بود و باید از «رگرسیون لجستیک» [۳] استفاده کرد.

یکی از روش‌های «رده‌بندی» در مبحث «آموزش با ناظر» [۴] رگرسیون لجستیک است. رگرسیون لجستیک را می‌توان توسط تابع لجستیک تعریف کرد. دامنه این تابع اعداد حقیقی هستند و برد این تابع بین صفر و یک می‌باشد.
الگوریتم پیشنهادی ارائه شده در این پژوهش با یک مسئله رده‌بندی رگرسیون لجستیک پیاده‌سازی شده است. این مسئله به صورت رابطه (۱.۴) نشان داده خواهد شد:

$$f_i(w) = log(1 + exp(-y_i x_i^T w)) + \frac{\lambda}{2}\|w\|^2 \qquad (۱.۴)$$

در رابطه بیان‌شده y_i و x_i برچسب‌های مدل می‌باشند و قسمت دوم معادله $\frac{\lambda}{2}\|w\|^2$ تابع تنظیم مسئله می‌باشد. پارامتر λ در این تابع نشان دهنده یک عدد مثبت است که میزان تنظیم مدل را معین می‌کند. هرچقدر λ کوچکتر باشد جریمه کمتری برای بزرگی نرم بردار پارامترها پرداخت می‌کنیم. مقدار مناسب برای λ از طریق آزمایش بر روی داده آزمون پیدا می‌شود.
با توجه به بیان مفروضات الگوریتم جهت خطی سازی همگرایی در الگوریتم پیشنهادی، این

[1] Clustering
[2] Correlation Coefficients
[3] Logistic Regression
[4] Supervised Machine Learning

مدل از رگرسیون لجستیک مفروضات را دارا می‌باشد.

۴.۴ مجموعه داده

در این پژوهش به جهت ارزیابی عملکرد الگوریتم پیشنهادی تحت مسئله رگرسیون لجستیک که در رابطه (۱.۴) نشان داده شده است. از چهار مجموعه داده مجزا در این ارزیابی استفاده شده است. هر یک از این چهار مجموعه داده به صورت جداگانه مورد بحث و بررسی قرار خواهد گرفت.

جدول ۱.۴: مشخصات مجموعه داده‌های مورد استفاده

مجموعه داده	تعداد ویژگی	تعداد داده آموزش	تعداد داده آزمون
covetype	۵۴	۴۰۶۷۹	۱۷۴۳۲
ijcnn1	۲۲	۹۱۷۰۱	۴۹۹۹۰
news20	۱۳۵۵۱۹۱	۱۳۹۹۷	۵۹۹۹
rcv1	۴۷۲۳۶	۶۷۷۰۳۹۹	۲۰۲۴۲

۱.۴.۴ پیش پردازش

مشخصات داده‌های[1] مورد استفاده در این ارزیابی با توجه به جدول ۱.۴ قابل مشاهده است. داده covetype دارای ۵۴ ویژگی می‌باشد که نسبت به سایر مجموعه داده‌ها، میزان پراکندگی[2] آن بیشتر است همچنین در این مجموعه، داده آزمون و داده آموزش مجزا نیست. بنابراین به صورت ۷۰ درصد و ۳۰ درصد به طور تصادفی داده آموزش و آزمون را مشخص کردیم. این مجموعه داده از مجموعه چند کلاسه به یک مجموعه دو کلاسه تبدیل شده است.

مجموعه داده دوم ijcnn1 می‌باشد. این مجموعه دارای ۲۲ ویژگی می‌باشد. این داده آموزشی در رنج مجموعه داده‌های با حجم بالا محسوب نمی‌شود اما به دلیل مقایسه در شرایط مساوی با الگوریتم پایه این مجموعه را مورد بررسی قرار داده‌ایم.

سومین مجموعه داده news20 می‌باشد. این مجموعه داده دارای تعداد ویژگی بسیار بالایی می‌باشد که نیاز به پاکسازی برای این داده وجود دارد که در آن داده‌های نویزی از بین رفته و اگر در جایی داده از دست رفته وجود داشته باشد با راه‌کارهایی مانند قرار دادن داده میانگین

[1] https://www.csie.ntu.edu.tw/ cjlin/libsvmtools/datasets/
[2] Sparsity

به جای داده از دست رفته آن را جایگزین کنیم. چنانچه داده از دست رفته در این مجموعه داده نسبت به covetype کمتر است اما روش جایگزینی داده از دست رفته با متوسط هر ستون شیوه مناسبی جهت احیاء این مجموعه داده می‌باشد.
مجموعه داده چهارم rcv1 است. این مجموعه داده دارای ۴۷۲۳۶ ویژگی می‌باشد که نسبت به دو مجموعه داده قبلی تعداد رکورد بیشتری دارد.

۲.۴.۴ مقادیر پارامترها

برای درک بهتر روند الگوریتم جدول ۲.۴ را جهت مشاهده مقادیر پارامترهای موجود در ارزیابی الگوریتم تهیه کرده‌ایم.

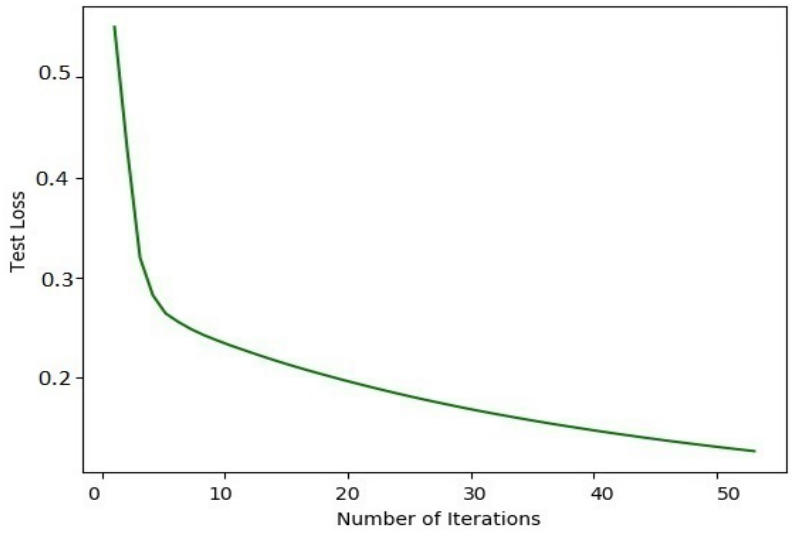
جدول ۲.۴: مقادیر پارامترها

پارامتر	مقدار پیش فرض	دیگز مقدارهای جایگزین
s	۵۰	۷۰ ـ ۳۰
m	۵۰	۷۰ ـ ۳۰
n	۶۷۷۰۳۹۹	۹۱۷۰۱ ـ ۱۳۹۹۷
L	.۰۰۰۰۰۰۰۵	۰.۰۰۰۰۱ ـ ۰.۰۰۰۰۷
t	۵۰	۷۰ ـ ۳۰
η	۰.۵	۰.۰۰۰۱ ـ ۰.۰۲

در جدول ۲.۴ مقادیر تعیین شده پارامترهای ثابت برای اجرای الگوریتم را درج کرده‌ایم. پارامتر s نشان دهنده تعداد تکرار در حلقه خارجی و پارامتر m نشان دهنده تکرار در حلقه داخلی می‌باشد که این مقدار بر حسب خطا و آزمون در الگوریتم بعد از قرار دادن چندین مقدار برای این دو پارامتر به دست آورده شده است. در نهایت میزان پارامتر η را نشان داده‌ایم که این پارامتر در حالت پیش فرض اولیه مسئله برابر ۰.۵ قرار گرفته است. لازم به ذکر است در مورد پارامتر نرخ یادگیری مقادیر ۰.۰۰۰۱ و ۰.۰۵ و ۰.۲ نیز مورد آزمون و خطا قرار گرفتند که در نهایت مقدار ۰.۵ را به عنوان پیش فرض مسئله انتخاب کردیم.

در ارزیابی اول با داده covetype الگوریتم را محک می‌زنیم. در این حالت داده با توجه به

جدول ۱.۴ دارای ۴۰۶۷۹ داده آموزش و ۱۷۴۳۲ داده تست می‌باشد. این مجموعه آموزشی نسبت مجموعه‌های دیگر داده مقادیر از دست رفته بیشتری دارد اما با تکنیک جایگزینی متوسط هر ستون داده به جای داده از دست رفته، مجموعه داده قابل استفاده می‌شود. با این تعداد داده و با نرخ یادگیری به مقدار ۰.۵ دقت الگوریتم در داده آموزشی ۹۹.۰۵ و برای داده آزمون ۹۸.۱۸۸ درصد بدست آمد. نمودار شکل ۱.۴ میزان خطای الگوریتم را به ازای گذرهای موجود در الگوریتم نشان می‌دهد.

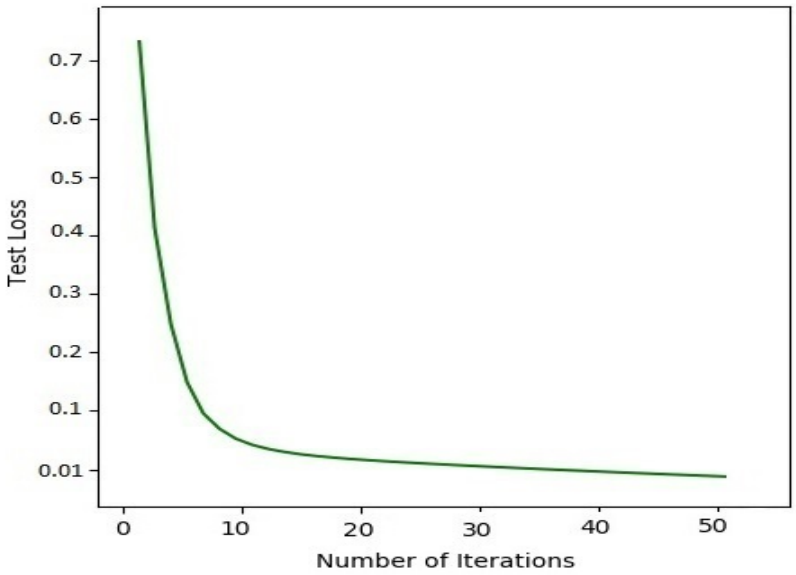


شکل ۱.۴: میزان خطا برحسب تعداد گذر در مجموعه داده covtyp

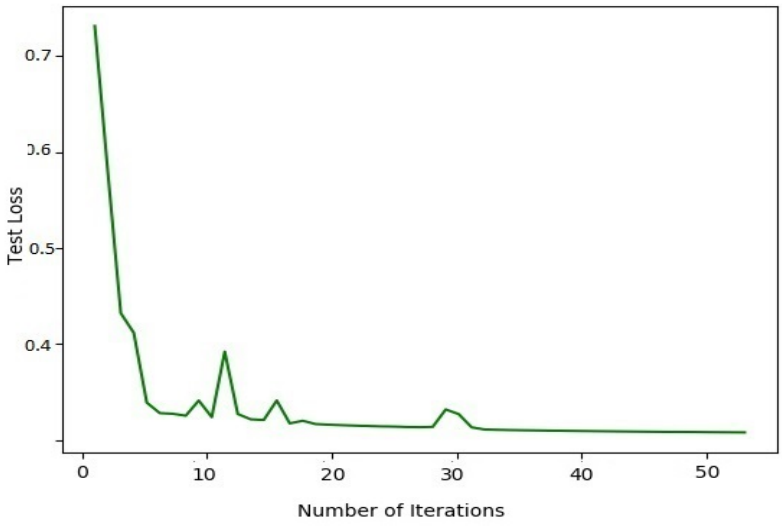
ارزیابی دوم با مجموعه داده ijcnn انجام شده است. که در شکل ۲.۴ نتیجه خطای آن بر حسب تعداد گذرهای برنامه پیاده‌سازی شده قابل مشاهده است. در این مجموعه داده که دارای ۴۹۹۹۰ داده آزمون و ۹۱۷۰۱ داده آموزش می‌باشد. دقتی برابر ۹۸.۴۵۷ درصد برای داده آموزشی و ۹۷.۲۳۵ درصد برای داده آزمون بدست آمد.

ارزیابی الگوریتم پیشنهادی با مجموعه داده سوم که news20 می‌باشد انجام پذیرفته است. این مجموعه داده شامل ۱۳۵۵۱۹۱ داده آموزشی و ۵۹۹۹ داده تست است که در شرایط مشابه با دیگر مجموعه داده‌ها دقتی برابر با ۹۶.۱۲درصد برای داده آموزشی و ۹۵.۴۵ برای داده آزمون حاصل شد که میزان خطای آن در گذرهای الگوریتم در شکل ۳.۴ این به تصویر کشیده شده است.

در نهایت آخرین مجموعه rcv1 مورد استفاده می‌باشد که دارای ۶۷۷۰۳۹۹ داده آموزش و ۲۰۲۴۲ داده آزمون بوده که دقت روش پیشنهادی برای داده آموزش برابر ۹۸.۶۳ برای داده آزمون برابر۹۷.۰۸ بدست آمد. نمودار خطا این مجموعه در شکل ۴.۴ نمایش داده شده است.

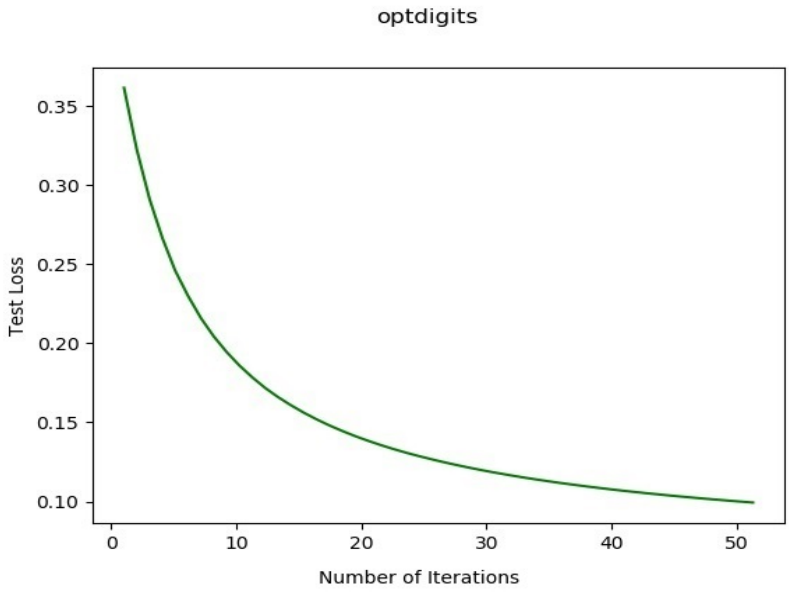


شکل ۲.۴: میزان خطا برحسب تعداد گذر در مجموعه داده ijcnn



شکل ۳.۴: میزان خطا برحسب تعداد گذر در مجموعه داده news20

همانطور که در شکل ۵.۴ نشان داده شده است، عملکرد الگوریتم پیشنهادی با تغییر تعداد نودهای کارگر در یک سطح، مورد مقایسه قرار گرفته است. بدیهی است که الگوریتم پیشنهادی به دلیل پیاده‌سازی روی یک سیستم توزیعی با توجه به افزایش تعداد گره‌های کارگر، سرعت پردازش الگوریتم به مراتب افزایش یابد. در حین تکرار، هر چه تعداد گره‌های کارگر بیشتر باشد سرعت بروزرسانی پارامترهای محلی جهت فرستاده شدن به سمت گره سرور افزایش یافته و پارامترهایی که به تازگی یادگیری شده‌اند، پس از فرستاده شدن به گره سرور تجمیع شده، در نهایت پارامتر تجمیع شده جایگزین پارامتر سراسری قبلی می‌شود. پس از



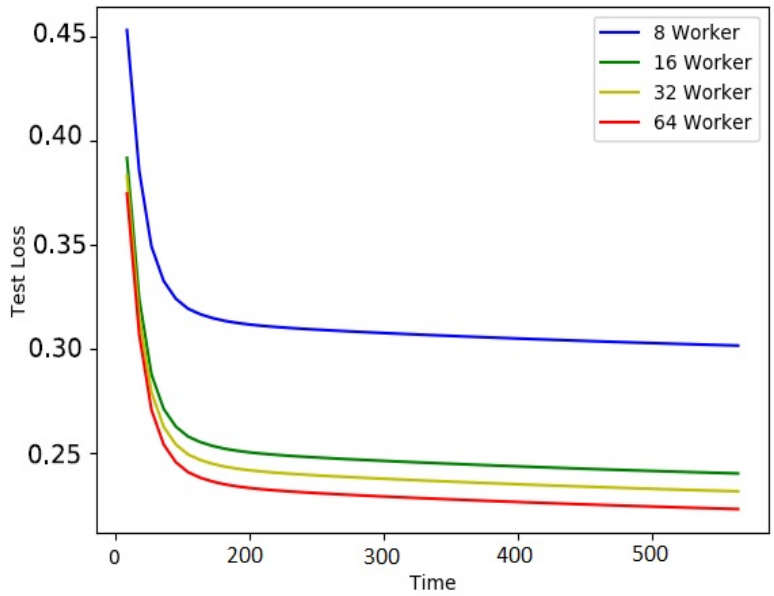
شکل ۴.۴: میزان خطا برحسب تعداد گذر در مجموعه داده rcv

آن، پارامترهای سراسری که به تازگی یادگرفته‌شده‌است، از طرف گره سرور، برای برنامه بعدی با نودهای کارگر دیگر به اشتراک گذاشته می‌شوند. بنابراین این امر باعث تسریع در همگرایی سایر گره‌های کارگر می‌شود. به همین خاطر، الگوریتم پیشنهادی هنگام تغییر تعداد نودهای کارگر، تقریباً سرعت خطی کسب می‌کند. به عنوان مثال، الگوریتم در استفاده از ۶۴ نود کارگر ، سرعت ۵۰ ثانیه را در مجموعه داده covtype به دست می‌آورد.

۵.۴ مقایسه الگوریتم پیشنهادی با الگوریتم‌های متداول

الگوریتم ارائه شده در یک سیستم توزیعی می‌باشد که طبق تعریف بیان شده این یک سیستم مبتنی بر مدل پارامتر سرور است. در واقع در این مدل به صورت یک خوشه، گره‌های کارگر و سرور با یکدیگر در ارتباط‌اند. بنابراین اگر بخواهیم مقایسه‌ای مبنی بر الگوریتم ارائه شده با الگوریتم‌های پایه مانند گرادیان نزولی، گرادیان نزولی تصادفی و گرادیان نزولی تصادفی بازگشتی داشته باشیم می‌توان در شرایط یکسان با مجموعه داده‌های یکسان این مقایسه را جهت مشخص کردن کارایی الگوریتم پیشنهادی نشان داد.

با توجه به جدول ۳.۴ می‌توان مقایسه به عمل آمده از این چهار الگوریتم را مشاهده کرد.



شکل ۵.۴: میزان کارایی الگوریتم برحسب تعداد گره‌های کارگر در مجموعه داده covtype

جدول ۳.۴: مقایسه کلی الگوریتم پیشنهادی با الگوریتم‌های مرسوم

الگوریتم	تعداد تکرار	η	دقت داده آموزش	دقت داده آزمون
GD	۵۰	۰.۰۵	٪ ۶۷.۸۷	٪۶۶.۵۴
SGD	۵۰	۰.۰۵	٪۸۶.۲۱	٪۸۵.۶۶
SARAH	۵۰	۰.۰۵	٪۹۵.۵۷	٪۹۴.۲۸
DisSARAH	۵۰	۰.۰۵	٪۹۸.۷۹	٪۹۷.۷۸

با توجه به ارزیابی انجام شده و مشاهدات ثبت شده در جدول ۳.۴، در روش گرادیان نزولی که از آن به عنوان روش مرسوم در الگوریتم‌های بهینه‌سازی یاد می‌شود، با نرخ یادگیری ۰.۰۵ و تکرار ۵۰ دقت در داده آزمون به طور متوسط برابر ۶۶.۵۴ درصد می‌باشد که این عملکرد در بهینه‌سازی عدم کارایی مناسب در حجم بالای داده را برای این الگوریتم مشخص می‌کند.
در دومین الگوریتم مورد بررسی گرادیان نزولی تصادفی دقت الگوریتم با توجه به انتخاب نرخ یادگیری مناسب نسبت به روش قبل برابر با ۸۵.۶۶۱ درصد بهبود داشته است اما باز هم به دلیل چالش بزرگ این الگوریتم یعنی نوسانات شدید در هنگام رسیدن به کمینه تابع هدف که موجب افزایش واریانس می‌شود، باعث کاهش سرعت و کندی در همگرایی این روش شود.
در سومین الگوریتم که پایه الگوریتم پیشنهادی ما را تشکیل می‌دهد، گرادیان نزولی تصادفی بازگشتی، باز هم با نرخ یادگیری ثابت و تکرار ۵۰ در حلقه داخلی و خارجی، دقت به‌دست آمده

در داده آزمون برابر با ۹۵.۸ درصد است که نسبت به دو الگوریتم قبلی بهبود قابل ملاحظه‌ای به دلیل نگرش بازگشتی به گرادیان نزولی تصادفی جهت کاهش واریانس داشته است. در اینجا ما سعی بر بهبود این الگوریتم به دلیل مشکلات انتخاب یکنواخت نمونه‌ها داشته و الگوریتم پیشنهادی را بر مبنای سیستم توزیع‌شده ارائه کردیم و در فصل گذشته آن را به تفصیل شرح دادیم.

در نهایت با توجه به مشاهدات بدست آمده از الگوریتم پیشنهادی، دقت ۹۷ درصد در حالت متوسط خواهیم داشت که می‌توان خاطر نشان کرد به جهت انتخاب غیر یکنواخت نمونه‌ها جهت بروز رسانی پارامترهای مدل یادگیری ماشین این میزان از بهبودی عملکرد حاصل می‌شود.

جدول ۳.۴ نشان دهنده متوسط پاسخ الگوریتم‌های مورد مقایسه قرار گرفته به داده‌های مورد استفاده است. در واقع می‌توان اینگونه شرح داد که در شرایطی که نرخ یادگیری η به میزان ۰.۰۵ باشد و تعداد تکرار را برای حلقه اصلی الگوریتم‌ها ۵۰ تکرار درنظر بگیریم مشاهده می‌شود که الگوریتم پیشنهادی به جهت پیاده‌سازی در سیستم توزیعی به دلیل انتخاب نمونه‌ها به صورت غیر یکنواخت عملکرد بهتری نسبت به الگوریتم‌های دیگر دارد.

همان‌طور که در مراحل شرح مسئله بیان شد این الگوریتم پیشنهادی یک پیاده‌سازی توزیع‌شده از روش بهینه‌سازی گرادیان نزولی تصادفی بازگشتی است. بنابراین به جهت مقایسه عملکرد این الگوریتم در حالت کلی می‌توان روش ارائه شده را با حالت ساده الگوریتم از نظر زمان اجرا مقایسه کرد. همان طور که در جدول ۴.۴ نشان داده شده است زمان اجرای الگوریتم پیشنهادی مبتنی بر سیستم توزیع‌شده به مراتب کمتر از زمان الگوریتم پایه در تمامی مجموعه داده‌ها می‌باشد.

جدول ۴.۴: مقایسه زمانی الگوریتم پیشنهادی و الگوریتم پایه

مجموعه داده	SARAH(time(s))	DSARAH(time(s))
covetype	۳۷۲	۱۴۸.۳
ijcnn1	۲۱۶	۲۳
news20	۲۵۴	۳۴.۸
rcv1	۲۹۴	۱۲۳

لازم به ذکر است که زمان بدست آمده برای الگوریتم پیشنهادی میانگین زمان چند مرتبه اجرای آن می‌باشد.

فصل ۵

نتیجه‌گیری و کارهای آینده

۱.۵ نتیجه گیری

همانطور که در فصل‌های گذشته به تفصیل بیان شد، در این پژوهش ابتدا به بررسی گرادیان نزولی، یکی از پر کاربردترین روش‌های بهینه سازی، پرداخته شد و می‌توان خاطر نشان کرد که این روش یکی از متداول‌ترین الگوریتم‌های بهینه سازی جهت یافتن کمینه یا بیشینه یک تابع هدف می‌باشد. همچنین بیان شد، تابع مورد نظر که الگوریتم بر روی آن پیاده‌سازی شده، تابع هزینه نام دارد. این تابع میزان خطا را برای الگوریتم پیشنهادی بر روی داده‌های آموزشی محاسبه می‌کند. جهت بالابردن کارایی الگوریتم پارامتری با نام پارامتر تنظیم معرفی شده است. در واقع با استفاده از این پارامتر به جای کاهش کل تابع هدف می‌توان این مقدار را افزایش داد و به این صورت از مشکل بیش برازش در تابع هدف جلوگیری می‌شود.

همچنین در ادامه الگوریتم بهبود یافته گرادیان نزولی با نام گرادیان نزولی تصادفی محاسبات این روش بهینه سازی به تفضیل مورد برسی قرار گرفت. مشاهده شد که این روش با وجود حذف محاسبات اضافی، به دلیل حجم بالای محاسبات با کندی در همگرایی نسبت به روش سنتی گرادیان نزولی همراه است، این امر موجب کاهش همگرایی در الگوریتم شده و دیگر اینکه موجب تشدید نوسانات در هنگام رسیدن به کمترین مقدار تابع می‌شود. بنابراین این

روش با مشکل واریانس بالا مواجه است.

از این رو به جهت کاهش واریانس حاصل از محاسبات در این روش بهبود یافته، از روش کاهش واریانس همراه الگوریتم گرادیان نزولی تصادفی استفاده شده است. در این الگوریتم نیاز به ذخیره‌سازی گرادیان‌های میانی وجود ندارد و از لحاظ نرخ همگرایی مشکلی ایجاد نمی‌شود و با کاهش همگرایی روبرو نمی‌شود. اما می‌توان مشاهده کرد که در الگوریتم گرادیان نزولی همراه با کاهش واریانس، یافتن گام مناسب جهت رسیدن به کمترین مقدار تابع مورد نظر با دشواری مواجه می‌باشد. در نتیجه الگوریتمی بازگشتی معرفی شده است که در آن نرخ یادگیری ثابت و به مراتب بزرگتر از نرخ یادگیری در روش کاهش واریانس می‌باشد.

با توجه به مفاهیم گفته شده و مواجه با حجم بالای داده‌ها جهت پردازش در الگوریتم‌های رده‌بندی، الگوریتم پیشنهادی در یک سیستم توزیع شده پیاده‌سازی شده‌است. یکی از خصوصیات مهم سیستم‌های توزیع شده که از دید کاربران مخفی است، تفاوت سیستم‌های مختلف و روش‌هایی است که از طریق آن‌ها این سیستم‌ها با هم ارتباط برقرار می‌کنند. گسترش سیستم‌های توزیع شده نسبتاً آسان است. اگر قسمت‌های خاصی از یک سیستم توزیع شده بطور موقت عملکرد صحیحی نداشته باشند، سیستم به دلیل این عملکرد مخرب از دسترس خارج نخواهد شد. البته در این سیستم‌ها سعی بر مخفی نگاه داشتن، تعویض یا تعمیر آن قسمت یا اضافه کردن بخش‌های جدید از دید کاربر شده‌است که به منظور خدمات رسانی بیشتر به کاربران و برنامه‌های کاربردی صورت می‌گیرد.

در سیستم توزیع شده اگر اطلاعاتی همزمان در چند سیستم به صورت یکسان ذخیره شود و یکی از سیستم‌ها خراب شود، اطلاعات را می‌توان از سیستم‌های دیگر بازیابی کرد و از این نظر قابلیت اطمینان افزایش می‌یابد. یکی از مزایای مهم سیستم توزیع شده سرعت بالای اجرای برنامه‌ها می‌باشد چرا که یک برنامه همزمان می‌تواند از چندین سیستم برای اجرا شدنش استفاده کند.

در واقع می‌توان این سیستم‌توزیع شده را با عنوان سیستم‌های پارامتر سرور معرفی کرد. در این مدل از سیستم‌ها، یک سرور اصلی وجود دارد، که داده‌های قسمت بندی شده را به سمت گره‌های کارگر روانه می‌کند و سپس عملیات در گره‌های کارگر صورت می‌پذیرد و در نهایت جهت تجمیع پارامترهای محلی بدست آمده و بروزرسانی شده به طرف سرور فرستاده می‌شود، این پارامتر نهایی را پارامتر سراسری می‌نامند.

بنابراین در این پژوهش جهت رده‌بندی توزیع شده، یک مسئله رده‌بندی با نام رگرسیون لجستیک با استفاده از روش بهینه سازی گرادیان نزولی تصادفی مورد بررسی قرار گرفته است.

همچنین چهار پایگاه داده با حجم بالا، جهت مقایسه زمان الگوریتم پیشنهادی با الگوریتم پایه ارائه شده که با توجه به نتایج بررسی شده در فصل چهارم این زمان کاهش چشم‌گیری داشته است.

۲.۵ کارهای آینده

به جهت استفاده از داده‌هایی با حجم بالا در الگوریتم‌های یادگیری ماشین و محاسبات پیچیده حاصل از این روش همچنین اهمیت موضوع و بهبود بیشتر در الگوریتم، در این بخش به ارائه راهکارهایی که در آینده می‌توان آن‌ها را بررسی کرد، پرداخته شده است.

۱.۲.۵ افزودن پارامتر جدید جهت کنترل گره‌های کارگر

در سیستم‌های توزیع شده ویژگی پارامتر سرور بودن، رابطه همکاری را توصیف می‌کند. مؤلفه سرور عملکردی یا خدماتی را برای یك یا بسیاری از گره‌های کارگر ارائه می‌دهد، که درخواست‌هایی برای استفاده از خدمات سرور به آن ارسال می‌شود. سرورها براساس خدماتی که ارائه می‌دهند طبقه بندی می‌شوند.

در یک سیستم توزیع شده به صورت پارامتر سرور، بعد از مرحله بخش‌بندی داده‌های ورودی،گهر یک از این بخش‌ها به گره‌های کارگر ارسال می‌شود. سپس هر یک از گره‌های کارگر با استفاده از نمونه انتخاب شده شروع به انجام محاسبات و به‌روزرسانی پارامترهای مسئله می‌کنند. در این مرحله در انجام محاسبات می‌توان مشاهده کرد که برخی از این گرههای کارگر یا زودتر و یا دیرتر محاسبات خود را به پایان می‌رسانند. این امر موجب دو دستگی در هنگام ارسال پارامترهای محلی جهت تجمیع به گره سرور می‌باشد. از طرفی از سیستم توزیع شده به جهت فواید آن برای افزایش کارایی یک سیستم و همچنین بالا بردن قابلیت اطمینان استفاده می‌شود. بنابراین این عدم هماهنگی در محاسبات توسط گره‌های کارگر موجب کاهش اطمینان و کارایی در الگوریتم پیشنهادی می‌شود.

از همین رو می‌توان پارامتری به جهت یکسان‌سازی عملکرد گره‌های کارگر تعریف کرد. در واقع این پارامتر موجب می‌شود که اگر یکی از گرههای کارگر زودتر از دیگر گره‌ها عملیات

محاسباتی خود را به اتمام رسانید صبر کند تا دیگر گره‌ها نیز محاسبات خود را انجام داده و در نهایت پارامترهای محلی را جهت تجمیع به سرور فرستاده و در آنجا پارامتر نهایی بروزرسانی شده و جایگزین پارامتر جهانی قبلی شود.

۲.۲.۵ قرار دادن فاکتور سرعت جهت بهبود نرخ یادگیری

چالش اصلی در الگوریتم گرادیان نزولی تصادفی در هنگام یافتن کمینه تابع هدف، میزان نوسانات در هنگام رسیدن به نقطه کمینه می‌باشد، که در واقع نشان دهنده افزایش واریانس در این روش است. همچنین این مشکل باعث کند شدن الگوریتم و کاهش همگرایی در این روش نسبت به گرادیان نزولی ساده می‌باشد.

بنابراین با توجه به مشکل پیش رو از روشی در کنار الگوریتم گرادیان نزولی تصادفی با عنوان کاهش واریانس استفاده شده است که چالش بوجود آمده را برطرف می‌سازد. در این روش برای یافتن نرخ یادگیری مناسب مشکلاتی بوجود می‌آید. بنابراین جهت بهبود این امر و سرعت بخشیدن به همگرایی الگوریتم می‌توان با یک فاکتور سرعت این مسئله را تا حدودی حل کرد. در واقع می‌توان نرخ یادگیری را با استفاده از یک فاکتور سرعت در هر دوره تغییر داد تا همگرایی مناسب‌تری در الگوریتم حاصل شود.

بنابراین می‌توان نرخ یادگیری الگوریتم را به صورت یک فرمول با دوبخش، یکی ثابت و دیگری فاکتور شتاب پیشنهاد کرد. قسمت ثابت میزان یادگیری پارامترها را از بهینه محلی استخراج می‌کند. در حالی که ضریب شتاب همگرایی الگوریتم را تسریع می‌بخشد. این تنظیم عامل شتاب به دلایل زیر قابل قبول است. در مرحله اول، عامل شتاب بزرگتر می‌شود و وقتی که پارامترها از حد مطلوب فاصله بگیرند، این امر موجب بازگشت همگرایی به الگوریتم می‌شود و موجب می‌شود که الگوریتم ارائه شده سریعاً متناسب با عامل نرخ یادگیری همگرا شود. عامل دوم، ضریب شتاب به اندازه‌ای بزرگ نخواهد شد که الگوریتم همگرا نشود. با توجه به اینکه عملکرد تا در الگوریتم رده بندی L-smooth باشد، یعنی $\|\nabla f_i\| < C$ می‌شود. در اینجا، C یک ثابت غیر منفی است. هنگامی که پارامترها نزدیک به بهینه سراسری هستند، عامل شتاب نزدیک به صفر می شود.

بنابراین الگوریتم پیشنهادی با حفظ شکل اصلی مسئله، در بهینه سراسری با سرعت ثابت همگرا می‌شود. در واقع این پیشنهاد جهت استفاده از فاکتور شتاب برای انجام کارهای رگرسیون خطی منظور شده است. در اینجا، یادگیری در الگوریتم به جای یک خوشه روی یک گره انجام می‌شود و همین امر موجب تسریع در زمان محاسبات الگوریتم پیشنهادی می‌شود.

نرخ یادگیری با یک عامل شتاب باعث می‌شود که الگوریتم رده‌بندی سریعتر از، نرخ یادگیری ثابت و بدون عامل شتاب همگرا شود. به طور خلاصه، عامل شتاب فواید آشکاری برای تسریع همگرایی یک الگوریتم رده‌بندی ایجاد می‌کند.

۳.۲.۵ افزودن پارامتر زمان

همانطور که در فصول گذشته بیان شد، گره‌های کارگر جهت انجام عملیات محاسباتی خود یک کپی از پارامتر سراسری را از سرور دریافت کرده و سپس عملیات بروزرسانی را آغاز می‌کنند. در این صورت با توجه به قرار دادن یک پارامتر زمان از طرف سرور، به جهت اعلان فرستادن یک کپی از پارامتر سراسری به تمام گره‌های کارگر، این گره‌ها می‌توانند بعد از اعلان این موضوع از طرف سرور به صورت همزمان عملیات محاسباتی خود را آغاز کنند. بنابراین افزودن این پارامتر نیز می‌تواند در بالا بردن سرعت عملکرد گره‌های کارگر در الگوریتم پیشنهادی تاثیر به سزایی داشته باشد.

مراجع

[1] Ravi, Sachin, and Hugo Larochelle,(2016), "Optimization as a model for few-shot learning",Published as a conference paper at ICLR,pp 227-238 .

[2] A . Coates , A .Y. Ng , H. Lee ,(2011) ," An analysis of single-layer networks in unsupervised feature, learning", In Proceedings of the fourteenth international conference on artificial intelligence and statistics ,pp 215–223

[3] G.E. Dahl , D. Yu , L. Deng , A. Acero,(2012)," Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition" **IEEE Transactions on audio, speech, and language processing** , pp 30–42

[4] R. Collobert , J. Weston,(2008)"A unified architecture for natural language processing: deep neural networks with multitask learning", Proceedings of the ACM International Conference on Machine Learning, pp. 160–167.

[5] Zaharia, Matei; Chowdhury, Mosharaf; Das, Tathagata; Dave, Ankur; Ma, Justin; McCauley, Murphy; J., Michael; Shenker, Scott,(2010) **Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing**,USENIX Association,pp 2-2.

[6] (1397) ": "

[7] Duda, Richard O., Peter E. Hart, and David G. Stork,(2012)"**Pattern classification** " , John Wiley and Sons, pp 147-158.

[8] Dimitri P. Bertsekas,(1999),"**Nonlinear Programming**" , Athena Scientific , 2nd edition, pp. 187.

[9] Cauchy, Augustin ,(1847) "General method for solving systems of simultaneous equations" **Comp. Rend. Sci**,pp 536-538.

[10] Mei, Song; Montanari, Andrea; Nguyen,(2018), "A mean field view of the landscape of two-layer neural networks" **Proceedings of the National Academy of Sciences**,pp 7665–7671.

[11] Tanenbaum, Andrew S.; Steen, Maarten van;(2007)," **Distributed systems: principles and paradigms**",Pearson Prentice Hall,pp 17-20.

[12] Lynch, Nancy A. ,(1996)"**Distributed Algorithms**",Elsevier ,pp 89-91.

[13] Andrews, Gregory R,(2000) **Foundations of Multithreaded, Parallel, and Distributed Programming**, Addison–Wesley, pp 111-124.

[14] Bentaleb, A.; Yifan, L.; Xin, J.,(2016) ," Parallel and Distributed Algorithms " **National University of Singapore**, pp 110-117.

[15] Peleg, David,(2008)," Distributed Computing: A Locality-Sensitive Approach"**SIAM Philadelphia PA**,pp 1-16.

[16] Papadimitriou, Christos H. ,(2003), "**Computational Complexity**", John Wiley and Sons, pp 260-265

[17] Ming, Yuewei and Zhao, Yawei and Wu, Chengkun and Li, Kuan and Yin, Jianping,(2018),"Distributed and asynchronous Stochastic Gradient Descent with variance reduction", **Neurocomputing: Elsevier**,pp 27-36.

[18] Ruder, Sebastian,(2016) "An overview of gradient descent optimization algorithms",Insight Centre for , **Data Analytics, NUI Galway** .

[19] Zhang, Guodong, James Martens, and Roger Grosse,(2019),"Fast Convergence of Natural Gradient Descent for Overparameterized Neural Networks" **Advances in Neural Information Processing Systems**,

[20] Jin, C., Netrapalli, P., Ge, R., Kakade, S. M., Jordan, M. I,(2019)"Stochastic Gradient Descent Escapes Saddle Points Efficiently" **ICML**.

[21] Qu, Guannan, and Na Li,(2019),"Accelerated distributed Nesterov gradient descent" **IEEE Transactions on Automatic Control**.

[22] Z. Allen-Zhu,(2017),"Recent advances in stochastic convex and non-convex optimization", **ICML Tutorial**,pp 119-131.

[23] Cramér, H,(1930),"**On the mathematical theory of risk** ",Centraltryckeriet,pp 20-37.

[24] Wald, A, "**Statistical Decision Functions**,1950, Wiley,pp 19-37.

[25] Alpaydin, Ethem ,(2010)," Introduction to Machine Learning", **MIT Press**, p 9-21.

[26] Vijayakumar, Sethu ,(2014) "The Bias–Variance Tradeoff", **University Edinburgh**,pp 620-626.

[27] Neal, Brady, (2018), "A modern take on the bias-variance tradeoff in neural networks" , **arXiv preprint** .

[28] Vandenberghe, Lieven , (2019)"Fast Gradient Methods", **Lecture notes for EE236C at UCLA**,pp 300-309.

[29] Liu, Sijia,(2018) "Zeroth-order stochastic variance reduction for nonconvex optimization", **Advances in Neural Information Processing Systems** ,pp 3727-3737. .

[30] Papini, M., Binaghi, D., Canonaco, G., Pirotta, M., Restelli,(2018)," Stochastic variance-reduced policy gradient", **arXiv preprint**.

[31] E.P. Xing , Q. Ho , W. Dai , J.K. Kim , J. Wei , S. Lee , X. Zheng , P. Xie , A. Kumar , Y. Yu ,(2015)," Petuum: a new platform for distributed machine learning on big data", **Big Data** ,pp 49-67.

[32] Cen, S., Zhang, H., Chi, Y., Chen, W., Liu, T. Y.,(2019)" Convergence of Distributed Stochastic Variance Reduced Methods without Sampling Extra Data", **arXiv preprint**.

[33] G. Coulouris, J. Dollimore, T. Kindberg and G. Blair,(2012), **Distributed Systems: Concepts and Design** , 5th Edition, Pearson-Addison Wesley Pub,pp 37-96.

[34] Howard, Alexander and Lee, Tim and Mahar, Sara and Intrevado, Paul and Woodbridge, Diane,(2018),"**Distributed Data Analytics Framework for Smart Transportation**",IEEE,pp 1374-1380

[35] Zhang, Ruiliang and Zheng, Shuai and Kwok, James T,(2016)"Asynchronous distributed semi-stochastic gradient optimization", **Association for the Advancement of Artificial Intelligence**, pp 2323–2329.

[36] Zinkevich, Martin and Weimer, Markus and Li, Lihong and Smola, Alex J,(2010),"**Advances in neural information processing systems :Parallelized stochastic gradient descent**", pp 2595-2603.

[37] Xing, Eric P and Ho, Qirong and Dai, Wei and Kim, Jin Kyu and Wei, Jinliang and Lee, Seunghak and Zheng, Xun and Xie, Pengtao and Kumar, Abhimanu and Yu, Yaoliang,(2015)"Petuum: A new platform for distributed machine learning on big data", **IEEE Transactions on Big Data**,pp 49-67.

[38] ZHANG, Bingjing and PENG, Bo and QIU, Judy,(2017),"Parallelizing Big Data Machine Learning Applications with Model Rotation", **New Frontiers in High Performance Computing and Big Data**,pp 199.

[39] Zhang, Wei and Gupta, Suyog and Lian, Xiangru and Liu, Ji,(2015),"Staleness-aware async-sgd for distributed deep learning", **Google Patents:arXiv preprint** .

[40] Chen, Xue-Wen and Lin, Xiaotong,(2014),"Big data deep learning: challenges and perspectives", **IEEE access**,pp 514-525.

[41] Nguyen, Lam M and Liu, Jie and Scheinberg, Katya and Tak, Martin,(2017)"SARAH: A novel method for machine learning problems using stochastic recursive gradient" **In Proceedings of the 34th International Conference on Machine Learning-Volume 70** ,pp 2613-2621.

[42] Feng, Minwei and Xiang, Bing and Zhou, Bowen,(2016),"**Distributed deep learning for question answering**",pp 2413-2416.

[43] Ghosh, Sukumar ,(2007),"**Distributed Systems – An Algorithmic Approach**", Chapman and Hall/CRC,pp 1-56.

[44] Lian, Xiangru and Huang, Yijun and Li, Yuncheng and Liu, Ji,(2015),"**Advances in Neural Information Processing Systems:Asynchronous parallel stochastic gradient for nonconvex optimization**",pp 2737-2745.

[45] Krivulin, Nikolai K,(1995),"**Proceedings of the Conference on Control and Information :Unbiased gradient estimation in queueing networks with parameter-dependent routing**",pp 351.

[46] Banks, Jerry,(1998),"**Handbook of simulation: principles, methodology, advances, applications, and practice**",pp 185-254.

[47] Foster, Ian and Kesselman, Carl and Tuecke, Steven,(2001),"The anatomy of the grid: Enabling scalable virtual organizations", **The International Journal of High Performance Computing Applications**,pp 200-222.

[48] Sanchez, Clifton W,"Resynchronization of a synchronous serial interface",(1998) **Google Patents**,pp 684-748.

[49] Dean, Jeffrey and Corrado, Greg and Monga, Rajat and Chen, Kai and Devin, Matthieu and Mao, Mark and Senior, Andrew and Tucker, Paul and Yang, Ke and Le, Quoc V and others,(2012),"**Advances in neural information processing systems:Large scale distributed deep networks** ",pp 1223-1231.

[50] Allen-Zhu, Z Katyusha,(2016),"The first direct acceleration of stochastic gradient methods" **The Journal of Machine Learning Research**, pp 148-158.

[51] Li, Mu and Zhang, Tong and Chen, Yuqiang and Smola, Alexander J,(2014),"**Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining: Efficient mini-batch training for stochastic optimization**",pp 661-670.

[52] Cotter, Andrew and Shamir, Ohad and Srebro, Nati and Sridharan, Karthik,"**Advances in neural information processing systems:Better mini-batch algorithms via accelerated gradient methods** ",2011, pp 1647-1655.

[53] Baldi, Pierre,(1995),"Gradient descent learning algorithm overview: A general dynamical systems perspective", **IEEE Transactions on neural networks**,pp 182-195.

[54] Klein, Stefan and Pluim, Josien PW and Staring, Marius and Viergever, Max A,(2009),"Adaptive stochastic gradient descent optimisation for image registration", **International journal of computer vision**, pp 227.

[55] Schmidt, Mark and Le Roux, Nicolas and Bach, Francis,(2017),"Minimizing finite sums with the stochastic average gradient", **Mathematical Programming**,pp 83-112

[56] Xiao, Lin and Zhang, Tong,(2014),"A proximal stochastic gradient method with progressive variance reduction", **SIAM Journal on Optimization**,pp 2057-2075.

[57] C. Darken, J. Chang, and J. Moody.(1992), "Learning rate schedules for faster stochastic gradient search" Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop,pp 1-11 .

واژه‌نامه انگلیسی به فارسی

رده‌بندی ..... Classification
بهینه‌سازی ..... Optimization
کلان داده ..... Big Data
ابزار آپاچی اسپارک ..... Apache Spark
نگاشت_کاهش ..... Map-Reduce
گرادیان نزولی ..... Gradient Descent
گرادیان صعودی ..... Gradient Ascent
گرادیان نزولی تصادفی ..... Stochastic Gradient Descent
سیستم توزیع شده ..... Distributed Systems
سیستم یکتا ..... Single Systems
منسجم ..... Coherent
میان افزار ..... Middle ware
قابلیت دسترسی ..... Accessible
شفافیت ..... Transparency
بازبودن ..... Open
مقیاس پذیری ..... Scable
نحو ..... Syntax
معنایی ..... Semantic
اندازه ..... Size
جغرافیایی ..... Geographically
راهبری ..... Administratively
پردازش داده ..... Data Processing
موازی ..... Parallel
نمودارحافظه مشترک ..... Nodes
کمینه ..... Minimum

بیشینه ..... Maximum
تابع هزینه ..... Cost Function
تابع هدر ..... Lost Function
نرخ یادگیری ..... Learning Rate
گام ..... Steps
همگرایی ..... Convergence
مقاوم کردن ..... Annealing
خطای غیر محدب ..... Non-Convex Error
کمینه محلی ..... Minimum Local
نقطه زینی ..... Saddle Point
نواسانات ..... Flunctuation
پروتکل ارتباطی آسنکرون ..... Asynchronous Serial Interface Protocol
رگرسیون لجستیک ..... Logistic Regression
واگرا ..... Divergence
پیچیدگی ..... Complexity
واریانس، ناسازگاری ..... Variance
بایاس، مورب ..... Bias
تعادل ..... Trade off
گرادیان تصادفی همرا با کاهش واریانس ..... Stochastic Variance Reduced Gradient
آستانه خطا ..... Fault Tolerance
گرادیان نزولی تصادفی بازگشتی ..... Stochastic Recursive Gradient Algorithm
گرادیان نزولی تصادفی بارشی ..... Down Pour Stochastic Gradient Descent
پورت سریال سنکرون ..... Synchronous Serial Port
مؤلفه ..... Lipschitz Continuous
تحدب قوی ..... Strong Convexity
استمرار یکنواخت ..... Uniform Continuous
ثابت لیپشیتز ..... Lipschitz Constant
پارامتر تنظیم ..... Regularization Item
همرفت ..... Smooth
گرادیان نزولی تصادفی نیمه انبوه ..... Mini-Batch Stochastic Gradient Descent
خوشه ..... Cluster
گره سرور ..... Server Node
گره کارگر ..... Worker Node
قسمت‌بندی داده ..... Data Partitioning

تجمیع کننده ..... Aggregate
فرستادن پارامتر ..... Pushing Parameters
دریافت پارامتر ..... Pulling Parameters
الگوریتم گرادیان تصادفی بازگشتی توزیع شده . Distributed Stochastic Recursive Gradient Algorithm
پارامتر سراسری ..... Global Parameters
پخش کننده ..... Broadcast
فایل سیستم توزیع شده ..... Distributed File Systems
مجموعه‌ای از ابزارهای نرم‌افزاری که می‌توان در منبع آن‌ها تغییر ایجاد کرد ..... Hadoop
فایل سیستم توزیع شده در نرم افزار هدوپ ..... Hadoop Distributed File Systems
راه‌انداز سیستم ..... Driver
مجموعه داده توزیعی انعطاف پذیر ..... Resilient Distributed Dataset
عملیات ..... Action
تبدیلات ..... Transformation
یک زبان برنامه نویسی تحت جاوا ..... Scala
جمع کننده ..... Accumulator
زمینه اسپارک ..... Spark Context
قسمت اصلی یک برنامه کامپیوتری ..... Main Program
ضریب همبستگی ..... Correlation Coefficients
یادگیری ماشین با ناظر ..... Supervised Machin Learning

Abstract

Today, for data processing in classification algorithms, we use huge datasets. Having a huge amount of calculations for big data classification, it is important to use a high-speed system for this kind of algorithms.

Also, to optimization objective function of classification algorithms, use various optimizations methods. Stochastic Gradient Descent (SGD) is one of the most popular algorithms to perform optimization and by far the most common way to optimize classification problems. This optimization method is an iterative method for optimizing an objective function with suitable smoothness properties. It can be regarded as a stochastic approximation of the gradient descent optimization,It is also one of the most important utilized algorithms to optimize parameters to reduce classification error in discrimination-based methods. However, due to the extra computation and the challenge of choosing the appropriate learning rate as well as the large fluctuations when reaching the minimum of the objective function, it can cause a high variance resulting in slow convergence. Therefore, to increase the convergence rate and reduce the variance in the algorithm, the Stochastic Recursive Gradient Algorithm (SARAH) was proposed as a novel approach to the finite-sum minimization problems.In this method, by using a recursive update method, the variance of each iteration is calculated and the parameters of the classification model are updated.However, it cannot support the distributed systems trivially due to the intrinsic design, which causes it to be able to perform in just one computational node. Therefore, when dealing with big data, using this optimization method, reduces the speed of classification algorithm.

In this study, to speed up performance of big data classification , a method is proposed that is called Distributed Classification with Stochastic Gradient Descent. In this method, a logistic regression classification problem is implemented using a Stochastic Gradient Descent optimization algorithm in a distributed system. In the algorithm implemented in a server parameter system, the parameters of the classification problem are updated in a cluster by the server and worker nodes, such that the worker nodes pull a copy of the global parameter using a request sent to the server, and parameters. Eventually the updated parameters are pushed to the server.The server node receives and aggregate these

parameters and stores them as a global parameter.

The proposed algorithm is implemented in distributed spark environment. Spark is a distributed general-purpose cluster-computing framework. This technology can be used for a variety of big data applications specially where the speed of operation is of particular importance. The results obtained from comparing distributed method with centralized method indicates that for all four training data sets, with the increase of worker nodes per run, the algorithm execution rate has almost doubled compare to the centralized method, keeping the convergence rate.

Key Words: Big Data, Classification,logistic regression, Gradient, Gradient Descent, Variance Reduced, Distributed Systems, Apache Spark

Faculty Of Computer Engineering

MSc Thesis in Artificial Intelligence Engineering

Distributed Classification with Stochastic Gradient Descent

By: Behnaz Sadat Mirhadi Tafreshi

Supervisor:

Dr. Hoda Mashayekhi

Advisor:

Dr. Mohsen Biglari

January 2019