
بسم الله الرحمن الرحيم

دانشگاه صنعتی اصفهان
دانشکده مهندسی برق و کامپیوتر

بهبود کارایی الگوریتم یادگیری فدرال برای داده‌های غیرمستقل و غیریکنواخت با در نظر گرفتن میزان شباهت بین شبکه‌های عصبی در دستگاه‌های نهایی

پایان‌نامه کارشناسی ارشد مهندسی کامپیوتر - هوش مصنوعی و رباتیکز

علی بزرگ‌زاد

استادراهنما

دکتر امیر خورسندی

۱۴۰۳

فهرست مطالب

عنوان **صفحه**

فصل سوم : بررسی اختصاصی پیشینه روش‌های حل مشکل ناهمگنی آماری

فصل چهارم : تعویض مدل‌های شبکه عصبی بین کاربران

چکیده

در این چکیده ...

کلمات کلیدی: یادگیری فدرال، یادگیری عمیق، ... .

فصل اول

مقدمه

۱-۱ شناخت موضوع

در سال‌های اخیر، پیشرفت‌های سریع فناوری و دسترسی آسان به اینترنت باعث شده‌اند که بسیاری از دستگاه‌های اطراف ما به اینترنت متصل شوند. این پدیده که به اینترنت اشیا[1] معروف است، شامل انواع دستگاه‌ها از جمله دستگاه‌های پوشیدنی[2]، خودروهای خودران، خانه‌های هوشمند[3] و به ویژه تلفن‌های هوشمند[4] می‌شود. این دستگاه‌ها به طور چشمگیری زندگی روزمره انسان‌ها را دگرگون کرده‌اند. استفاده از این سیستم‌ها همگی باعث تولید حجم قابل توجهی داده در طول روز می‌شوند که شرکت‌های بزرگ فناوری از این داده‌ها بهره برده و با استفاده از آن‌ها اقدام به انواع سرویس‌دهی به کاربران خود می‌نمایند.

با پیشرفت علم هوش مصنوعی و استفاده گسترده از روش‌های یادگیری ماشین، امکان بهره‌برداری بهینه از حجم عظیم داده‌های تولید شده فراهم شده است. این داده‌ها می‌توانند برای اجرای الگوریتم‌های مختلف به منظور دستیابی به اهداف متنوع به کار گرفته شوند. روش‌های متعددی برای مدیریت و اجرای این الگوریتم‌های یادگیری وجود دارد که در ادامه به توضیح هر یک پرداخته خواهد شد.

---

[1] Internet of Things
[2] Wearable Devices
[3] Smart Homes
[4] Smart Phones

۱-۱-۱ یادگیری متمرکز

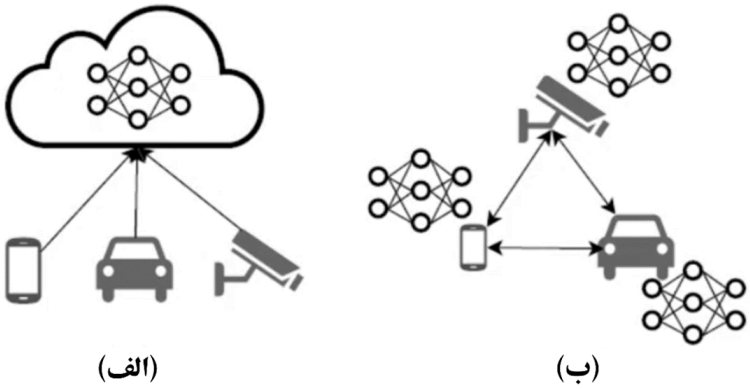
روش یادگیری متمرکز[1] که در بسیاری از سیستم‌های امروزی به کار می‌رود، به این صورت عمل می‌کند که تمامی گره‌ها[2] اطلاعات خود را به صورت کامل به سرویس‌دهنده ابری[3] ارسال می‌کنند. سرویس‌دهنده ابری با دسترسی به تمامی داده‌ها، الگوریتم‌های مورد نظر را اجرا می‌کند [۱]. این روش در شکل ۱-۱ (الف) به تصویر کشیده شده است.

۱-۱-۲ یادگیری غیر متمرکز

در روش یادگیری غیر متمرکز[4] ، هر گره به صورت مستقل الگوریتم‌های مورد نظر را اجرا می‌کند. پس از چند مرحله اجرای کد، اطلاعات به‌روز شده را با گره‌های همسایه به اشتراک می‌گذارد. این فرآیند تا زمانی ادامه می‌یابد که تمامی گره‌ها به یک مقدار مشخص همگرا شوند [۲]. این روش در شکل ۱-۱ (ب) نشان داده شده است.

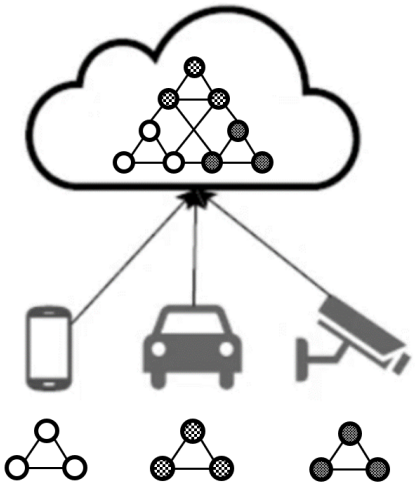
۱-۱-۳ یادگیری توزیع شده

در روش یادگیری توزیع‌شده[5] ، یک هسته مرکزی مسئولیت مدیریت کل سیستم و تمامی داده‌ها را بر عهده دارد. با این حال، به دلیل نیاز به توان پردازشی بالا، این هسته بار پردازشی را بین گره‌های موجود تقسیم می‌کند. در این رویکرد، فرض بر این است که تمامی گره‌ها دارای توان پردازشی یکسانی هستند و داده‌ها به طور مساوی بین گره‌ها توزیع می‌شوند. این روش در شکل ۱-۲ نشان داده شده است.



شکل ۱-۱: (الف) یادگیری متمرکز، (ب) یادگیری غیرمتمرکز [۲].

[1] Centralized Learning
[2] Nodes
[3] Cloud Server
[4] Decentralized Learning
[5] Distributed Learning



شکل ۱-۲: یادگیری توزیع شده.

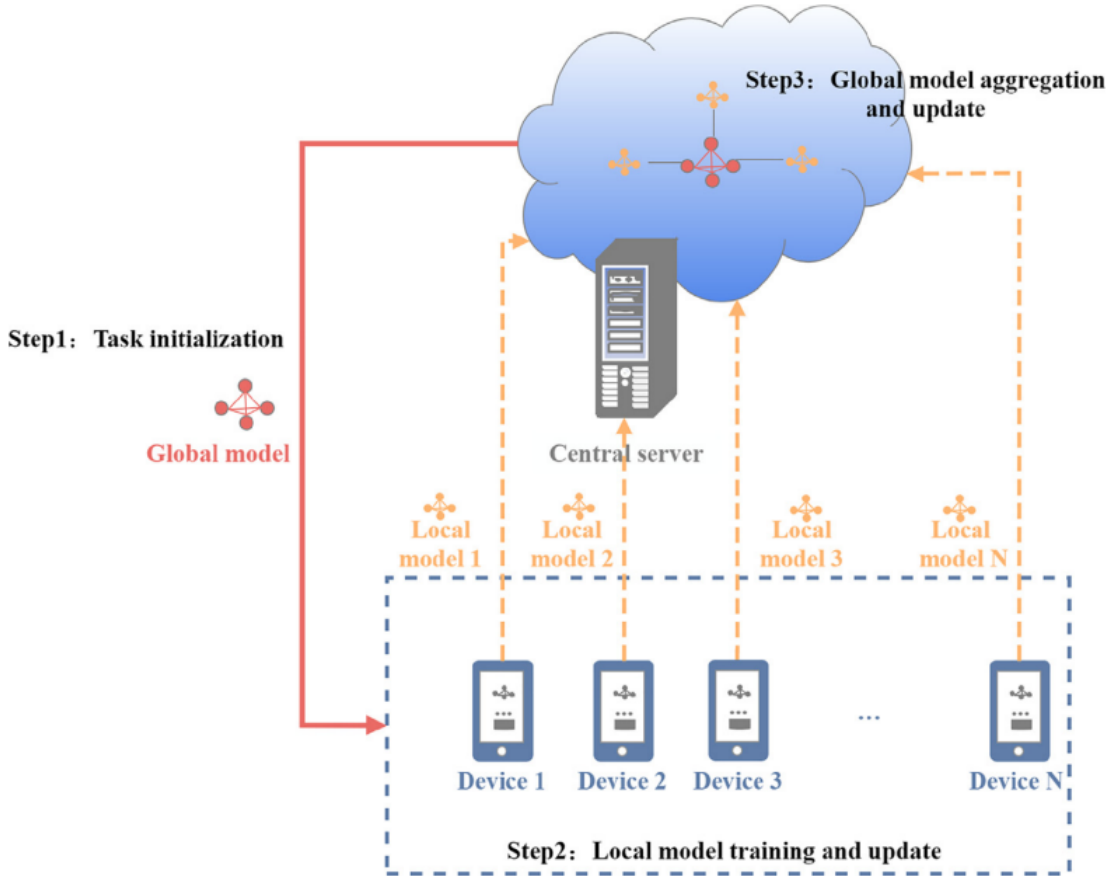
۱-۲ یادگیری فدرال

سیستم‌های متمرکز تا پیش از این بیشتر نیازها را برطرف می‌کردند، اما در دنیای امروزی و با افزایش تعداد دستگاه‌های متصل، چالش‌های جدیدی مطرح شده است. هزینه‌های بالای مرتبط با انتقال حجم زیاد داده‌ها از یک جهت، و افزایش نگرانی‌ها درباره امنیت اطلاعات حساس و شخصی از جهت دیگر، محققان را به سمت استفاده از الگوریتم‌های غیرمتمرکز و توزیع‌شده در حوزه یادگیری ماشین سوق داده است. یکی از جدیدترین زیرمجموعه‌های مهم و پرکاربرد روش‌های یادگیری توزیع‌شده، یادگیری فدرال است که بسیار مورد توجه قرار گرفته است.

در روش یادگیری فدرال، برخلاف رویکردهای متمرکز یادگیری ماشین، تجزیه و تحلیل داده‌ها به دستگاه‌های لبه[1] یا سرویس‌گیرنده‌ها[2] منتقل می‌شود. این روش، به عنوان یک جایگزین مطلوب برای مدل‌سازی داده‌ها در محیط‌هایی با تعداد زیادی سرویس‌گیرنده معرفی شده است. در این چارچوب، به جای انتقال داده‌های اصلی، پارامترهای مدل‌های محلی در هر مرحله از فرآیند آموزش به سمت سرور منتقل می‌شوند، که این امر توانایی بهبود امنیت و کاهش هزینه‌های ارتباطی را فراهم می‌کند. در شکل ۱-۳ این معماری به نمایش گذاشته شده است.

سرور در حقیقت نقش رهبری را ایفا می‌کند و با توجه به نوع داده‌ها، یک مدل شبکه عصبی[3] ایجاد کرده و آن را به سمت کاربران ارسال می‌کند. در ادامه کاربران با توجه به داده‌های خود شبکه را آموزش می‌دهند و بعد از چند بار تکرار به صورت محلی، وزن‌های به‌روزرسانی شده را به سمت سرور بر می‌گردانند. همان‌طور که در شکل ۱-۳ مشاهده می‌شود، داده‌ها همگی در سمت کاربران قرار گرفته‌اند و به سمت سرور ارسال نمی‌شوند.

[1] Edge Devices
[2] Clients
[3] Neural Network



شکل ۱-۳: یادگیری فدرال [۳].

عدم اجبار در به اشتراک گذاشتن اطلاعات گره‌ها در یادگیری فدرال، کمک شایانی به حفظ حریم شخصی کاربران می‌کند [۴].

۱-۳ تاریخچه یادگیری فدرال

در اوایل فصل بهار سال ۲۰۱۷، محققان گوگل (Google) برای اولین بار موضوع یادگیری فدرال را در یک مطلب کوتاه در وبلاگ هوش مصنوعی خود معرفی کردند. این مطلب با عنوان "یادگیری فدرال: یادگیری ماشین اشتراکی، بدون نیاز به آموزش متمرکز داده‌ها" منتشر شد [۵]. در این نوشته، به طور مختصر از Google Keyboard یا به اختصار Gboard صحبت شد که با بهره‌گیری از یادگیری فدرال، قابلیت پیش‌بینی و پیشنهاد لغت بعدی به کاربر را دارد. با استفاده از یادگیری فدرال، دیگر نیازی به ارسال داده‌های کاربران به سرور نبود و مدل به صورت محلی به‌روزرسانی می‌شد.

این روش، با بهره‌گیری از اطلاعات بسیار زیاد ذخیره شده در دستگاه‌ها، بدون نیاز به ارسال داده‌های حساس به سرور، به حفظ حریم شخصی کاربران کمک کرده و خدمات بهتری را ارائه می‌دهد. در شکل ۱-۴، نحوه

استفاده از یادگیری فدرال در این برنامه به نمایش درآمده است.

۱-۴ کاربرد یادگیری فدرال

ارتباطات نرم‌افزاری و سخت‌افزاری به معنای توانایی تبادل داده‌ها و هماهنگی عملکرد بین اجزای مختلف یک سیستم است، به طوری که این اجزا بتوانند به صورت یکپارچه و هماهنگ با یکدیگر کار کنند. فناوری مبتنی بر صنعت ۱۴/۰[1]، این ارتباطات را در انواع سیستم‌ها به طور گسترده‌ای گسترش داده است. این هماهنگی بین نرم‌افزار و سخت‌افزار، به یک پدیده مهم در محیط‌های هوشمند و خودکار تبدیل شده است.

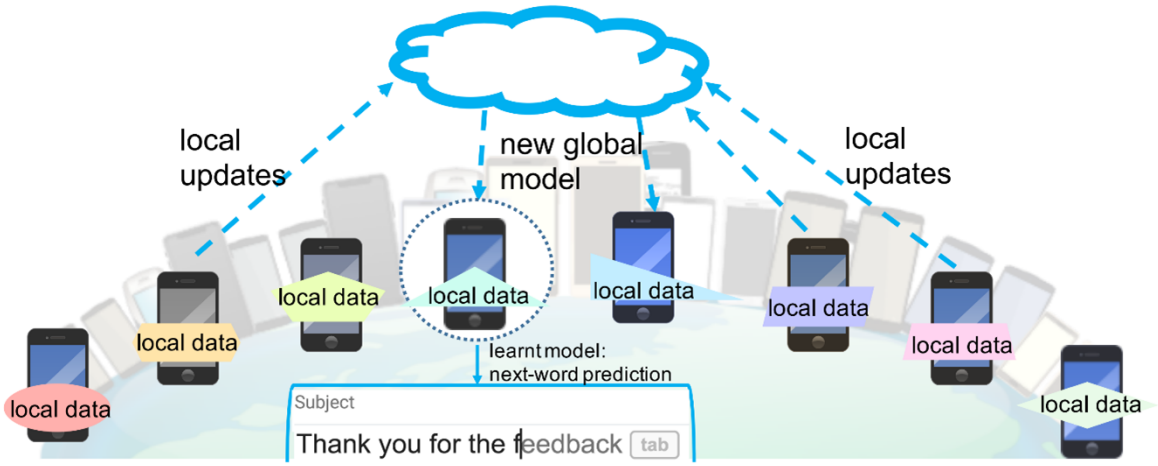
سامانه‌های متمرکز قبلی که تنها مسئول جمع‌آوری، پایش و کنترل شرایط به صورت محلی بودند، اکنون جای خود را به دستگاه‌های هوشمندی داده‌اند که قابلیت پردازش و برنامه‌ریزی داده‌ها را در سطح سیار و سیستمی دارند. علاوه بر این، گسترش ارتباطات مبتنی بر اینترنت، امکان انتقال و تبادل داده‌ها بین سیستم‌های مختلف را فراهم کرده است. این تحولات منجر به کاهش نیاز به تصمیم‌گیری متمرکز و توسعه سیستم‌های کنترل و پایش پیشرفته شده است. این ویژگی‌ها، همراه با حجم روزافزون داده‌ها، یادگیری فدرال را به یکی از بهترین روش‌ها برای توسعه سیستم‌های هوشمند تبدیل کرده است [۷]. در ادامه، سه نمونه از کاربردهای یادگیری فدرال شرح داده خواهد شد.

۱-۴-۱ یادگیری فدرال در شهر هوشمند

در یک شهر هوشمند[2]، اطلاعات جمع‌آوری شده از حسگرهای مختلف مانند داده‌های ترافیک، مصرف انرژی، پسماند شهری و رویدادهای امنیتی، ارزش بالایی دارند و به عنوان منبعی کلیدی برای بهبود عملکرد شهر هوشمند و ارتقای کیفیت زندگی شهروندان محسوب می‌شوند. اما در کنار این مزایا، حفظ حریم شخصی و امنیت اطلاعات شهروندان نیز از اهمیت بالایی برخوردار است. یادگیری فدرال به عنوان یک رویکرد نوین که مبتنی بر حفظ حریم شخصی است، در اینجا به کار گرفته می‌شود.

در یک شهر هوشمند، سازمان‌های مختلف هر کدام اطلاعات خاص خود را دارند، اما این اطلاعات به طور متقابل بر یکدیگر تأثیر می‌گذارند و می‌توانند در مدیریت بهینه شهر نقش مهمی ایفا کنند. یادگیری فدرال با حفظ حریم شخصی کاربران، این امکان را فراهم می‌کند که سازمان‌ها بدون نیاز به اشتراک‌گذاری داده‌های حساس خود با یکدیگر، از داده‌های موجود بهره‌برداری کنند و مدل‌های هوش مصنوعی و الگوریتم‌های بهبود عملکرد شهر هوشمند را توسعه دهند. به عنوان مثال، با استفاده از یادگیری فدرال می‌توان بهبود مدیریت ترافیک، بهینه‌سازی

[1] Industry 4.0
[2] Smart City



شکل ۱ - ۴: استفاده از یادگیری فدرال برای پیش‌بینی کلمه بعدی در Gboard [۶].

مصرف انرژی، کاهش آلودگی هوا و افزایش امنیت شهری را تحقق بخشید، در حالی که حریم شخصی شهروندان به بهترین نحو ممکن حفظ می‌شود.

۱ - ۴ - ۲ یادگیری فدرال در بیمارستان

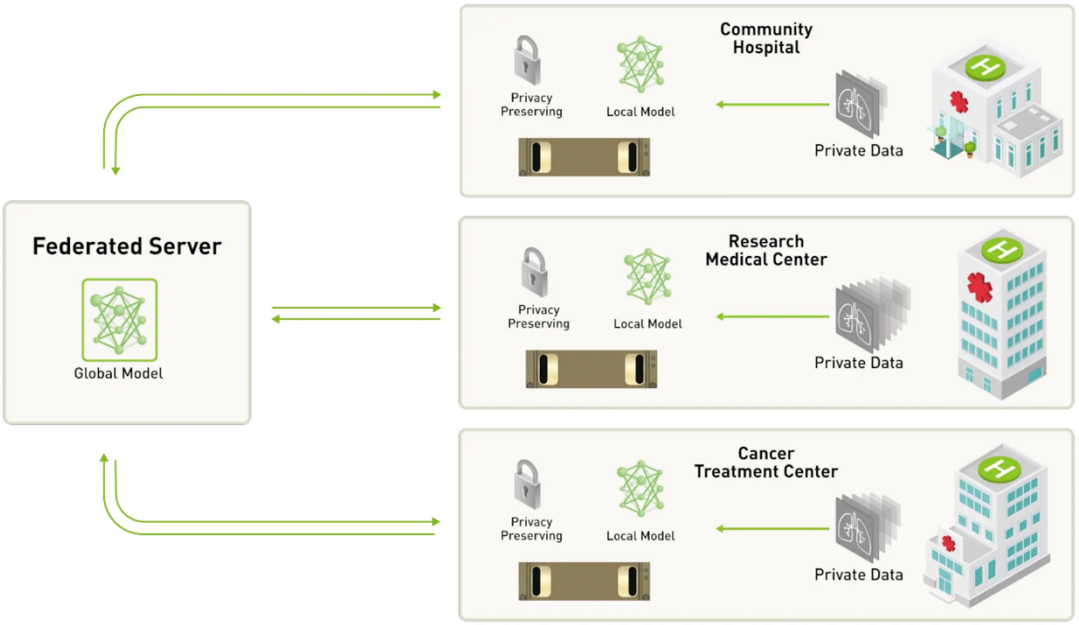
در یک بیمارستان، اطلاعات پزشکی به شدت حساس و مهم هستند و باید به صورت محرمانه نگهداری شوند. با این حال، بهره‌برداری از این داده‌ها برای ارتقاء خدمات بهداشتی و درمانی بسیار ارزشمند است. در این شرایط، یادگیری فدرال می‌تواند نقش مهمی ایفا کند. با استفاده از روش‌های یادگیری فدرال، بیمارستان‌ها می‌توانند از داده‌های پزشکی بیماران خود برای توسعه مدل‌هایی استفاده کنند که به بهبود خدمات، ارتقاء روش‌های تشخیص و درمان بیماری‌ها و افزایش بهره‌وری پزشکان کمک می‌کنند، بدون اینکه نیاز باشد این داده‌ها به طور مستقیم به یک مرکز جمع‌آوری اطلاعات ارسال شوند.

برای مثال، با بهره‌گیری از یادگیری فدرال، مدل‌های هوش مصنوعی می‌توانند روی داده‌های محلی بیماران در هر بیمارستان آموزش داده شوند تا بیماری‌های مختلف را شناسایی و تشخیص دهند و اطلاعات مورد نیاز برای درمان‌های مؤثرتر را فراهم کنند، در حالی که اطلاعات حساس بیماران به طور کامل محافظت می‌شود. در شکل ۱ - ۵ یک نمونه استفاده از یادگیری فدرال در بیمارستان‌ها به نمایش در آمده است.

۱ - ۴ - ۳ یادگیری فدرال در فروشگاه برنامه‌های کاربردی تلفن‌همراه

یک فروشگاه برنامه‌های کاربردی[1] تلفن‌همراه را در نظر بگیرید که به کاربران امکان دریافت و نصب برنامه‌های مختلف را می‌دهد. این فروشگاه می‌خواهد با استفاده از داده‌های کاربران خود، الگوریتمی توسعه دهد که بتواند به طور دقیق‌تری برنامه‌های مورد علاقه کاربران را پیشنهاد دهد. اگر این فروشگاه از روش‌های متمرکز استفاده

[1] App Store



شکل ۱ - ۵: یادگیری فدرال در بیمارستان [۸].

کند، باید داده‌های حساس و شخصی کاربران را جمع‌آوری و تحلیل کند، که این موضوع می‌تواند نگرانی‌های جدی در مورد حریم خصوصی کاربران ایجاد کند و غیر عملی باشد.

با استفاده از یادگیری فدرال، این فروشگاه می‌تواند الگوریتم خود را بر روی داده‌های محلی هر کاربر اجرا کند. به این ترتیب، هیچ داده حساسی به یک مرکز جمع‌آوری داده‌ها ارسال نمی‌شود و حریم خصوصی کاربران حفظ می‌شود. به عنوان مثال، اگر یک کاربر به برنامه‌های موسیقی علاقه‌مند باشد، الگوریتم محلی در تلفن هوشمند او می‌تواند این الگو را شناسایی کند و پیشنهادات مربوط به برنامه‌های موسیقی را ارائه دهد، بدون این‌که نیاز به ارسال داده‌های شخصی و حساس او به سرور شرکت باشد.

۱ - ۵ دید کلی از روند موضوع و بیان هدف پژوهش

تکمیل این بخش پس از رسیدن به ساختار کلی پایان‌نامه (چون ممکنه در ادامه تغییر کنه)

چند جلمه کلیدی:

به دلیل پراکندگی همگرایی به کندی صورت می‌گیرد

روش جابجایی وزن‌ها بین کاربران نهایی در طول فرایند

چرا جابجایی تصادفی، جابجایی هوشمند بر اساس میزان شباهت

۱-۶ مروی بر روند ارائه مطالب پایان‌نامه

تست

فصل دوم

مفاهیم پایه در یادگیری فدرال و نگاه کلی به پیشینه پژوهش چالش‌ها

۲-۱ مقدمه

توزیع داده‌ها بین کاربران در یادگیری فدرال ممکن است با چالش‌ها و مشکلات گوناگونی روبرو شود. یکی از مشکلات اساسی، اختلافات و ناسازگاری‌هایی است که ممکن است در فرآیند آموزش میان کاربران یا دستگاه‌های مختلف پدید آید. اگر این چالش‌ها پیش از آغاز فرآیند مدل‌سازی به درستی شناسایی نشده و راه‌حل‌های مناسبی برای آن‌ها اتخاذ نشود، مدل نهایی احتمالاً با مشکلاتی همچون کاهش دقت و عملکرد روبرو خواهد شد. این مسئله یکی از بزرگترین موانع در مسیر یادگیری فدرال است و نیازمند دقت و استفاده از روش‌های خلاقانه برای حل آن است.

در این فصل، ابتدا به بیان ریاضی یادگیری فدرال پرداخته می‌شود که برای درک آن نیاز به آشنایی پایه با مفاهیم ریاضی در یادگیری ماشین و یادگیری عمیق است. سپس چالش‌های موجود در یادگیری فدرال بررسی شده و دیدگاه‌های مختلف مقالات علمی در مورد هر یک از این چالش‌ها به صورت کلی مرور می‌شود. در نهایت، به رویکردهای اصلی و اساسی برای حل این چالش‌ها اشاره خواهد شد.

۲-۲ ریاضیات پایه در یادگیری فدرال

برای تشریح ریاضیات پایه در یادگیری فدرال، ابتدا لازم است تا مفاهیم اساسی یادگیری ماشین و یادگیری عمیق را بررسی کنیم و رابطه‌های اصلی مرتبط با آن‌ها را بیان کنیم. پس از این مقدمه، با مرتبط کردن این اصول به یادگیری فدرال، می‌توانیم به طور دقیق ریاضیات اولیه در یادگیری فدرال را توضیح دهیم و نشان دهیم که چگونه این مفاهیم در این حوزه خاص به کار گرفته می‌شوند.

۲-۲-۱ مفاهیم پایه در یادگیری ماشین و یادگیری عمیق

یادگیری ماشین شاخه‌ای از هوش مصنوعی است که به سیستم‌ها اجازه می‌دهد بدون نیاز به برنامه‌نویسی صریح، از داده‌ها بیاموزند و پیش‌بینی کنند. در یادگیری ماشین، الگوریتم‌ها با استفاده از داده‌های ورودی، مدل‌هایی می‌سازند که می‌توانند الگوها و روابط پیچیده را در داده‌ها تشخیص دهند. این فرآیند به کامپیوترها امکان می‌دهد تا با تجربه و مشاهده، بهبود پیدا کنند و وظایفی مانند تشخیص تصویر، پردازش زبان طبیعی و پیش‌بینی بازار را انجام دهند.

در حالی که یادگیری عمیق یک زیرمجموعه از یادگیری ماشین است که از شبکه‌های عصبی مصنوعی برای مدل‌سازی و یادگیری از داده‌ها استفاده می‌کند. این روش‌ها از لایه‌های متعدد برای استخراج ویژگی‌ها و یادگیری الگوها در داده‌های پیچیده بهره می‌برند. شبکه‌های عصبی عمیق، که شامل چندین لایه مخفی هستند، قادر به یادگیری ویژگی‌های سطح بالا از داده‌های ورودی می‌باشند. این لایه‌ها به ترتیب اطلاعات را پردازش کرده و به یکدیگر منتقل می‌کنند تا خروجی نهایی تولید شود.

یادگیری عمیق برای تنظیم وزن‌های شبکه عصبی از الگوریتم‌های بهینه‌سازی بهره می‌برد. یکی از این الگوریتم‌ها، گرادیان نزولی[1] است که با تعیین شیب تابع هزینه[2]، وزن‌ها را به‌طور مکرر به‌روزرسانی می‌کند تا به کمترین مقدار ممکن برای این تابع برسد. الگوریتم انتشار به عقب[3] یکی از مهم‌ترین روش‌ها در این زمینه است که از گرادیان نزولی برای بهینه‌سازی وزن‌ها استفاده می‌کند. در این فرآیند، ابتدا خطای خروجی شبکه محاسبه می‌شود و سپس این خطا به‌صورت معکوس از لایه خروجی به سمت لایه‌های ورودی منتقل می‌شود تا وزن‌ها تنظیم شوند و شبکه به دقت مطلوب دست یابد.

۲-۲-۲ فرمول‌های پایه در یادگیری عمیق

- تابع هزینه و انتشار به عقب

[1] Gradient Descent
[2] Loss Function
[3] Backpropagation

تابع هزینه یا تابع خطا معیاری است که اختلاف بین خروجی پیش‌بینی شده و مقدار واقعی را اندازه‌گیری می‌کند. یکی از توابع هزینه رایج، میانگین مربعات خطا[1] (MSE) است:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(y_i - \hat{y}_i)^2 \qquad (۱-۲)$$

که در آن y_i مقدار واقعی، $\hat{y}_i$ مقدار پیش‌بینی شده و m تعداد نمونه‌ها است. الگوریتم انتشار به عقب از این تابع هزینه استفاده می‌کند تا وزن‌ها را به‌روزرسانی کند. این فرآیند شامل محاسبه گرادیان‌ها و به‌روزرسانی وزن‌ها در جهت کاهش خطا است.

- بهینه‌سازی با گرادیان نزولی

بهینه‌سازی با گرادیان نزولی یکی از رایج‌ترین روش‌ها برای به‌روزرسانی وزن‌های شبکه عصبی است. رابطه به‌روزرسانی وزن‌ها به صورت زیر است:

$$\theta_j \leftarrow \theta_j - \alpha\frac{\partial J(\theta)}{\partial \theta_j} \qquad (۲-۲)$$

که در آن θ_j وزن، α نرخ یادگیری و $\frac{\partial J(\theta)}{\partial \theta_j}$ مشتق جزئی تابع هزینه نسبت به وزن θ_j است. این فرآیند تکرار می‌شود تا تابع هزینه به حداقل مقدار خود برسد.

۲-۲-۳ ارتباط مفاهیم یادگیری عمیق با یادگیری فدرال

یادگیری فدرال از مفاهیم پایه‌ای یادگیری عمیق و شبکه‌های عصبی بهره می‌برد، اما با ساختاری توزیع‌شده که در آن داده‌ها بین چندین دستگاه تقسیم شده‌اند. در یادگیری فدرال، مدل‌های یادگیری عمیق به‌صورت محلی بر روی دستگاه‌های کاربران آموزش داده می‌شوند و تنها به‌روزرسانی‌های مدل به سرور مرکزی ارسال می‌شود. این روش، علاوه بر حفظ حریم خصوصی داده‌ها، امکان استفاده از داده‌های گسترده و متنوع را فراهم می‌کند. الگوریتم‌های بهینه‌سازی مانند گرادیان نزولی و انتشار به عقب به‌طور محلی اجرا می‌شوند و به‌روزرسانی‌ها به‌صورت تجمیعی برای بهبود مدل کلی استفاده می‌شوند، که یادگیری فدرال را به یک رویکرد قدرتمند برای مدل‌سازی در محیط‌های توزیع‌شده تبدیل می‌کند.

۲-۲-۴ بیان ریاضی یادگیری فدرال

برای بررسی مباحث ریاضی پایه در یادگیری فدرال، ابتدا باید مسئله بهینه‌سازی مرکزی که در این زمینه مطرح می‌شود، به‌طور دقیق تعریف گردد. در یادگیری فدرال، هدف اصلی یافتن مجموعه‌ای از پارامترهای مدل است که عملکرد کلی مدل را بر روی داده‌های توزیع‌شده بین تعداد زیادی دستگاه بهینه کند. هر دستگاه دارای داده‌های

[1] Mean Squared Error

محلی است و یک تابع هزینه محلی بر اساس این داده‌ها برای آن دستگاه تعریف می‌شود. مسئله بهینه‌سازی کلی در یادگیری فدرال به دنبال کمینه کردن مجموع وزنی این توابع هزینه محلی است تا یک مدل جامع و یکپارچه حاصل شود.

یک طرح به‌روزرسانی همزمان در نظر گرفته می‌شود که به صورت دوره‌های ارتباطی انجام می‌شود. در این سیستم، یک مجموعه ثابت از K مشتری وجود دارد که هر کدام دارای یک مجموعه داده محلی ثابت هستند. در ابتدای هر دوره، یک زیرمجموعه تصادفی شامل C مشتری انتخاب می‌شود و سرور وضعیت فعلی پارامترهای مدل جهانی را به هر یک از این مشتری‌ها ارسال می‌کند. هر مشتری انتخاب شده سپس بر اساس وضعیت جهانی و مجموعه داده محلی خود محاسبات محلی را انجام می‌دهد و یک به‌روزرسانی به سرور ارسال می‌کند. سپس سرور این به‌روزرسانی‌ها را برروی وضعیت جهانی خود اعمال می‌کند و این فرآیند تکرار می‌شود [۹].

در حالی که تمرکز بر اهداف شبکه عصبی غیرمحدب[1] است، الگوریتم مورد بررسی برای هر هدف جمع‌ـ متناهی[2] به صورت زیر قابل اعمال است.

$$\min_{w\in\mathbb{R}^d} f(w) \quad \text{where} \quad f(w) \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} f_i(w) \tag{۳-۲}$$

برای یک مسئله یادگیری ماشین، معمولاً $f_i(w) = \ell(x_i, y_i; w)$ در نظر گرفته می‌شود، به این معنی که این تابع نشان‌دهنده‌ی خطای پیش‌بینی بر روی نمونه (x_i, y_i) با استفاده از پارامترهای مدل w است. فرض می‌کنیم که داده‌ها بین K مشتری تقسیم شده‌اند، که در آن $\mathcal{P}_k$ مجموعه‌ای از نقاط داده مربوط به مشتری k است و $n_k = |\mathcal{P}_k|$ تعداد این نقاط داده را نشان می‌دهد. بنابراین، با توجه به این مورد می‌توان رابطه (۳-۲) را به صورت زیر بازنویسی نمود:

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \quad \text{where} \quad F_k(w) = \frac{1}{n_k}\sum_{i\in\mathcal{P}_k} f_i(w) \tag{۴-۲}$$

اگر مجموعه $\mathcal{P}_k$ با توزیع یکنواخت[3] تصادفی از مثال‌های آموزشی بین مشتری‌ها تشکیل شده باشد، در آن صورت $\mathbb{E}_{\mathcal{P}_k}[F_k(w)] = f(w)$ خواهد بود، که در اینجا امید ریاضی بر روی مجموعه مثال‌های اختصاص داده شده به یک مشتری ثابت گرفته می‌شود. این همان فرض استقلال و توزیع یکنواخت داده‌ها[4] (IID) است که عموماً توسط الگوریتم‌های بهینه‌سازی توزیع‌شده استفاده می‌شود، در اینجا حالتی که فرض مذکور برقرار نیست (یعنی F_k می‌تواند تقریباً به هر میزانی از f فاصله داشته باشد) به عنوان حالت غیرمستقل و غیریکنواخت (non-IID) شناخته می‌شود [۹].

[1] non-Convex
[2] Finite-Sum
[3] Uniform Distribution
[4] Independent and Identically Distributed

۲-۳ چالش‌های موجود در یادگیری فدرال و نگاه کلی مقالات به آن‌ها

با وجود مزایای فراوان در مقایسه با روش‌های سنتی یادگیری ماشین، یادگیری فدرال به دلیل ساختار شبکه‌ای خود با چالش‌های متعددی مواجه است. در ادامه به بررسی چالش‌های اصلی یادگیری فدرال و دیدگاه کلی مقالات در مورد آن‌ها خواهیم پرداخت.

۲-۳-۱ تبادل داده

تبادل داده بین سرور و کاربران به دلیل مشکلات پهنای باند و ارتباطات شبکه‌ای اصولا کار پر هزینه‌ای می‌باشد. یکی از دلایل اصلی پرهزینه بودن این ارتباطات، حجم بالای داده‌هایی است که باید بین دستگاه‌های کاربری و سرور منتقل شوند. معمولاً مشکلات ارتباطی به انتقال‌های بسیار زیاد به‌روزرسانی‌های مدل بین گره‌های محاسباتی نسبت داده می‌شود. با افزایش تعداد پارامترها در مدل‌های پیشرفته، اندازه این مدل‌ها نیز به طور متناسب بزرگ می‌شود [۱۰].

از سوی دیگر، تعداد زیادی از دستگاه‌های کاربران نهایی در فرآیند آموزش مدل‌ها مشارکت دارند که این امر می‌تواند هزینه‌های ارتباطی را به طور قابل توجهی افزایش دهد. همچنین، به دلیل مشکلات ارتباطی، در بسیاری از مواقع همه دستگاه‌ها در هر چرخه از فرآیند آموزش شرکت نمی‌کنند که این مسئله نیز باعث افزایش هزینه‌ها و پیچیدگی‌های مرتبط با انتقال داده‌ها می‌شود.

استفاده از فشرده‌سازی داده‌ها می‌تواند هزینه‌های ارتباطی را به میزان قابل توجهی کاهش دهد. برای مدیریت هزینه‌های بالای ارتباطات در فرآیند یادگیری فدرال، روش‌هایی مورد بررسی قرار گرفته‌اند که بر فشرده‌سازی داده‌های ارسالی از دستگاه‌های نهایی به سرور مرکزی تمرکز دارند. این تکنیک‌ها با کاهش حجم اطلاعات ارسالی، به کاهش هزینه‌های ارتباطی کمک می‌کنند [۱۱].

روشی به نام PCFL[1] وجود دارد که از نظر ارتباطی بسیار کارآمد است و شامل سه عنصر اصلی می‌باشد. این عناصر شامل فشرده‌سازی دوطرفه، فشرده‌سازی مکانی وزن‌ها و یک پروتکل پیشرفته برای حفظ حریم خصوصی داده‌ها هستند. فشرده‌سازی دوطرفه، داده‌ها را در دو مرحله، هم قبل از ارسال از دستگاه‌های نهایی به سرور و هم هنگام ارسال نتایج به‌روزرسانی‌شده از سرور به دستگاه‌ها، فشرده می‌کند تا حجم داده‌های انتقالی کاهش یابد. فشرده‌سازی مکانی وزن‌ها نیز با فشرده کردن وزن‌های مدل، حجم انتقال را کاهش داده و کارایی ارتباطات را بهبود می‌بخشد. پروتکل حفظ حریم خصوصی داده‌ها نیز امنیت اطلاعات کاربران را در طول فرآیند یادگیری فدرال تضمین می‌کند. این سه عنصر با همکاری هم، موجب کاهش هزینه‌های ارتباطی و بهبود کارایی در روش

[1] Privacy Communication efficient Federated Learning

PCFL می‌شوند [۱۲].

۲-۳-۲ ناهمگنی‌های سیستمی[1]

در دنیای یادگیری فدرال، دستگاه‌ها از نظر حافظه، توان محاسباتی و ارتباطات بسیار با یکدیگر متفاوت هستند. این تفاوت‌ها ممکن است از اختلافاتی مانند تفاوت در پردازنده، نوع حافظه، نوع اتصال شبکه و نیاز به انرژی ناشی شود. محدودیت‌های موجود در شبکه و سیستمی می‌توانند باعث ایجاد وضعیت‌هایی شوند که برخی از دستگاه‌ها در یک زمان معین در دسترس نباشند. برای مثال، اگر تعداد زیادی دستگاه همزمان درخواست ارسال داشته باشند، ممکن است برخی از آن‌ها به دلیل پهنای باند محدود یا محدودیت‌های سخت‌افزاری، قادر به ارسال درخواست نشوند. همچنین، ممکن است یک دستگاه فعال، به دلیل مشکلاتی مانند اختلالات در شبکه یا مصرف اضافی انرژی، از فرآیند یادگیری خارج شود.

این تفاوت‌های سیستمی، یکی از چالش‌های یادگیری فدرال محسوب می‌شوند و می‌توانند باعث افزایش تأخیر و ایجاد اشکالات در سیستم شوند. بنابراین، برای رفع این مشکلات، روش‌های یادگیری فدرال باید توانایی پیش‌بینی دقیق تعداد دستگاه‌هایی که در هر فرآیند شرکت می‌کنند را داشته باشند. همچنین، باید بتوانند در برابر دستگاه‌هایی که در حین عملیات دچار مشکل شده و از دسترس خارج می‌شوند، مقاومتی مناسب داشته باشند [۶].

برای مقابله با ناهمگنی سیستمی، روشی تحت عنوان تعادل در به‌روزرسانی مدل مطرح شده است. در این روش، وزن‌دهی به نمونه‌ها بر اساس میزان نیاز به آموزش در هر دستگاه صورت می‌گیرد. این کار باعث می‌شود که دستگاه‌های با حجم داده کمتر، وزن بیشتری در به‌روزرسانی مدل داشته باشند [۱۳]. در رویکرد دیگری به نام یادگیری فعال، دستگاه‌هایی که داده‌های خود را به سرور ارسال می‌کنند، فعالیت خود را به نحوی تنظیم می‌کنند که مدل از داده‌های مهم‌تر و کمتر دیده شده بیشتر یاد می‌گیرد. این روش می‌تواند به تعادل در آموزش مدل کمک کند و از ناهمگنی سیستمی جلوگیری کند [۱۱].

۲-۳-۳ ناهمگنی‌های آماری[2]

روش‌های مختلفی برای تولید و جمع‌آوری داده‌ها بین دستگاه‌ها وجود دارد. این داده‌ها معمولاً به صورت مستقل از هم تولید نمی‌شوند و بین آن‌ها ارتباطات و پیوندهایی وجود دارد. چنین الگویی از تولید داده با فرضیات استقلال و توزیع یکنواخت داده‌ها (IID) در مسائل بهینه‌سازی در تضاد است، که منجر به پیچیدگی‌هایی در فرآیند مدل‌سازی، تحلیل نظری و ارزیابی عملکرد می‌شود. بنابراین، با وجود هدف نهایی که یادگیری یک مدل

[1] Systems Heterogeneity
[2] Statistical Heterogeneity

جامع و یکپارچه است، روش‌های جایگزین مانند یادگیری چندوظیفه‌ای[1] و فرایادگیری[2] به عنوان راه‌حل‌های ممکن مطرح شده‌اند [۶].

یک روش برای حل مشکل ناهمگنی آماری در یادگیری فدرال استفاده از رویکرد ترکیبی یا ترکیب روش‌های یادگیری محلی است. در این رویکرد، به جای استفاده از یک الگوریتم یادگیری مشترک برای تمام دستگاه‌ها، از چندین الگوریتم یادگیری محلی با تنوع مدل‌ها و تنظیمات مختلف استفاده می‌شود. سپس، اطلاعات مدل‌های محلی روی سرور یا گره مرکزی جمع‌آوری می‌شود و با استفاده از ترکیب این اطلاعات، یک مدل یادگیری مشترک به‌روزرسانی خواهد شد [۱۳].

۲-۳-۴ حریم شخصی

اگرچه حفظ حریم شخصی یک مزیت مهم در یادگیری فدرال به شمار می‌رود، اما در صورت عدم کنترل مناسب می‌تواند به یک چالش تبدیل شود. یکی از چالش‌های اساسی در این زمینه، نگهداری حریم خصوصی است که به دلیل قرار گرفتن داده‌های حساس و شخصی در اختیار بخش‌های مختلف شبکه، اهمیت بیشتری پیدا می‌کند. در این روش، دستگاه‌های محلی داده‌های کاربران را جمع‌آوری و به سرور ارسال می‌کنند تا مدل‌های یادگیری مشترک به‌روزرسانی شوند. این ارتباطات می‌توانند شامل اطلاعات حساسی باشند که امکان شناسایی افراد یا فرآیندهای حیاتی آن‌ها را فراهم می‌کنند.

یکی از مشکلات کلیدی اینجاست که حتی با استفاده از روش‌های رمزنگاری و امنیتی، ممکن است اطلاعات خاصی همچنان به سرور ارسال شوند که می‌تواند حریم خصوصی را نقض کند. به‌ویژه، اگر داده‌های حساس مانند اطلاعات هویتی به صورت رمزگذاری نشده انتقال یابند، امنیت حریم خصوصی کاربران به خطر می‌افتد.

روش حفظ حریم خصوصی تفاضلی[3] با افزودن نویز به نتایج محاسبات یا به داده‌های ورودی، اطمینان حاصل می‌کند که حضور یا عدم حضور یک نمونه داده خاص در مجموعه داده‌ها، تأثیر قابل توجهی بر خروجی محاسبات نداشته باشد. این روش به ویژه برای حفظ حریم خصوصی در یادگیری فدرال مفید است زیرا از افشای اطلاعات حساس از طریق پارامترهای مدل جلوگیری می‌کند [۱۴].

رویکرد رمزنگاری هم‌شکل[4] امکان محاسبه روی داده‌های رمزنگاری شده را بدون نیاز به رمزگشایی آن‌ها فراهم می‌کند. این تکنیک به ویژه در یادگیری فدرال برای حفظ حریم خصوصی داده‌ها در حین انجام محاسبات مفید است زیرا نیاز به تغییر ماهیت داده نبوده و چون جابجایی در یادگیری فدرال بسیار زیاد رخ می‌دهد، این روش بسیار کارا خواهد بود [۱۵].

---

[1] Multi-Tasking
[2] Meta Learning
[3] Differential Privacy
[4] Homomorphic Encryption

۲-۴ رویکردهای کلی و پایه‌ای در حل چالش‌ها

روش‌های بهینه‌سازی توزیع‌شده معمولاً برای حل مسائل بهینه‌سازی در سیستم‌هایی با شبکه‌های محاسباتی بزرگ و توزیع‌شده استفاده می‌شوند. این روش‌ها بر مبنای تقسیم مسئله بهینه‌سازی به زیرمسائل کوچک‌تر و حل آن‌ها در گره‌های مختلف شبکه استوارند. در این روش‌ها، اغلب فرض می‌شود که داده‌ها به صورت همگن و یکپارچه در سراسر شبکه توزیع شده‌اند و گره‌ها می‌توانند به راحتی با یکدیگر ارتباط برقرار کنند.

این فرضیات در یادگیری فدرال به ندرت برقرار است، زیرا در یادگیری فدرال داده‌ها به صورت محلی و ناهمگن در دستگاه‌های مختلف قرار دارند و ارتباطات بین دستگاه‌ها ممکن است محدود و نامنظم باشد. بنابراین روش‌ها و رویکردهای لازم جهت حل این چالش‌ها متفاوت از مسائل بهینه‌سازی توزیع شده هستند. در این مرحله، تلاش می‌شود دو رویکرد پایه‌ای برای مسائل یادگیری فدرال معرفی شود.

۲-۴-۱ به‌روزرسانی محلی و میانگین‌گیری در سرور

یکی از روش‌های اصلی و پرکاربرد در یادگیری فدرال روش میانگین‌گیری فدرال[1] (FedAvg) است که توسط محققان گوگل در سال ۲۰۱۷ معرفی شد [۹]. این الگوریتم به منظور بهینه‌سازی مدل‌های یادگیری ماشین در یک محیط توزیع‌شده طراحی شده است. در این روش داده‌ها به صورت محلی در دستگاه‌های کاربران باقی می‌مانند و تنها به‌روزرسانی‌های مدل به اشتراک گذاشته می‌شوند. رویکرد اصلی FedAvg بر مبنای ترکیب به‌روزرسانی‌های محلی از دستگاه‌های مختلف به یک مدل جهانی استوار است.

یکی از مزایای اصلی FedAvg این است که به طور موثری با چالش ناهمگنی داده‌ها مقابله می‌کند. در یادگیری فدرال، داده‌های موجود در دستگاه‌های مختلف ممکن است توزیع‌های متفاوتی داشته باشند. این ناهمگنی می‌تواند به دلیل تفاوت در رفتار کاربران یا حتی محیط‌های مختلف جمع‌آوری داده باشد. میانگین‌گیری وزنی در FedAvg به مدل کمک می‌کند تا به‌روزرسانی‌های مختلف را به گونه‌ای ترکیب کند که این ناهمگنی‌ها را در نظر بگیرد. به عبارت دیگر، اگر یک دستگاه داده‌های بیشتری داشته باشد، تأثیر بیشتری بر مدل نهایی خواهد داشت. این رویکرد باعث می‌شود که مدل فدرال به تعادل بهتری در یادگیری از داده‌های ناهمگن برسد و کارایی بالاتری داشته باشد. این ویژگی به ویژه در کاربردهایی مانند فروشگاه برنامه‌های کاربردی که کاربران متنوع و داده‌های متفاوتی دارند، بسیار سودمند است و می‌تواند به بهبود عملکرد مدل در شرایط واقعی کمک شایانی کند.

علاوه بر این، FedAvg به کاهش نیاز به ارتباطات مکرر بین دستگاه‌ها و سرور مرکزی کمک می‌کند. در

[1] Federated Averaging

بسیاری از روش‌های بهینه‌سازی توزیع‌شده، نیاز است که دستگاه‌ها به طور مکرر با سرور مرکزی ارتباط برقرار کنند تا به‌روزرسانی‌های خود را ارسال کنند. اما در FedAvg دستگاه‌ها می‌توانند چندین مرحله از بهینه‌سازی را به صورت محلی انجام دهند و سپس تنها به‌روزرسانی نهایی را ارسال کنند. این کاهش در نیاز به ارتباطات نه تنها باعث کاهش پهنای باند مورد نیاز می‌شود، بلکه به حفظ حریم خصوصی کاربران نیز کمک می‌کند، زیرا داده‌ها هرگز از دستگاه‌های محلی خارج نمی‌شوند. بررسی‌ها نشان داده‌اند که متناسب با اندازه داده‌ها پس از رسیدن به تعداد معینی از گره‌ها، اضافه کردن گره‌های بیشتر تأثیری در کاهش هزینه‌های ارتباطی نخواهد داشت. در چنین شرایطی، تمرکز بر افزایش توان محاسباتی محلی یا تعداد مراحل آموزش محلی می‌تواند موجب تسریع فرایند آموزش شود [۹].

موفقیت‌های اخیر در کاربردهای یادگیری عمیق تقریباً به‌طور انحصاری به استفاده از انواع الگوریتم نزول گرادیان تصادفی[1] (SGD) برای بهینه‌سازی متکی بوده‌اند. در واقع، بسیاری از پیشرفت‌ها به تنظیم مدل و بهینه‌سازی تابع خطا با روش‌های ساده گرادیان مربوط می‌شود. بنابراین، منطقی است که الگوریتم‌های بهینه‌سازی فدرال با شروع از SGD طراحی و توسعه یابند [۹].

الگوریتم SGD می‌تواند به سادگی در بهینه‌سازی فدرال استفاده شود، به این صورت که در هر دور ارتباط، گرادیان‌ها بر اساس داده‌های یک مشتری تصادفی انتخاب شده، محاسبه شوند. این رویکرد از نظر محاسباتی کارآمد است، اما نیازمند تعداد بسیار زیادی از دورهای آموزش برای تولید مدل‌های خوب است. برای مثال حتی با استفاده از رویکرد پیشرفته‌ای مانند نرمال‌سازی دسته‌ای[2]، برای آموزش مجموعه‌داده‌ای تنها با ۶۰,۰۰۰ داده ورودی و با دسته‌های کوچکی به اندازه ۶۰، به ۵۰,۰۰۰ دور آموزش جهت رسیدن به مدل مطلوب نیاز می‌باشد [۱۶].

در تنظیمات فدرال، مشارکت تعداد زیادی از مشتریان هزینه چندانی در زمان واقعی ندارد زیرا همه کاربران می‌توانند به صورت همزمان به آموزش مدل محلی بپردازند. بنابراین، برای خط مبنا از SGD همزمان با دسته‌های بزرگ استفاده می‌شود. برای اعمال این رویکرد در تنظیمات فدرال، در هر دور یک زیرمجموعه‌ای از مشتریان با ضریب کنترلی C انتخاب می‌شوند و گرادیان خطا روی تمام داده‌های نگهداری شده توسط این مشتریان محاسبه می‌گردد. بنابراین، C اندازه دسته کلی را کنترل می‌کند، به‌طوری که $C = 1$ معادل با نزول گرادیان یک دسته کامل است. این الگوریتم خط مبنا FederatedSGD یا FedSGD نامیده می‌شود [۹].

یک پیاده‌سازی معمول از FedSGD با $C = 1$ و نرخ یادگیری ثابت η به این صورت است که هر گره k، گرادیان $g_k = \nabla F_k(w_t)$ که میانگین گرادیان روی داده‌های محلی در مدل فعلی w_t است را محاسبه می‌کند و

[1] Stochastic Gradient Descent
[2] Batch Normalization

سرور مرکزی این گرادیان‌ها را جمع‌آوری کرده و به‌روزرسانی $w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$ را انجام می‌دهد، در حالی که $\sum_{k=1}^{K} \frac{n_k}{n} g_k = \nabla f(w_t)$ خواهد بود. یک به‌روزرسانی معادل به این صورت است که برای هر گره عبارت $\forall k, w_{t+1}^k \leftarrow w_t - \eta g_k$ محاسبه و سپس $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$ انجام شود.

در نتیجه، هر گره به صورت محلی یک گام گرادیان نزولی را روی مدل فعلی با استفاده از داده‌های محلی خود انجام داده و سپس سرور میانگین وزنی مدل‌های به‌دست‌آمده را محاسبه می‌کند. با نوشتن الگوریتم به این صورت، امکان تکرار به‌روزرسانی محلی $w^k \leftarrow w^k - \eta \nabla F_k(w^k)$، چندین بار پیش از مرحله میانگین‌گیری فراهم شده و باعث افزایش محاسبات در هر گره خواهد شد. الگوریتم FederatedAveraging (FedAvg) به این صورت به وجود آمد [۹]. جهت درک بهتر این ساختار می‌توانید شکل ۱-۳ را مشاهده کرده و در گام سوم شکل میانگین وزنی مدل‌ها را در نظر بگیرید.

در این روش میزان محاسبات توسط سه پارامتر کلیدی کنترل می‌شود: C، زیرمجموعه‌ای از تعداد گره‌هایی که در هر مرحله محاسبات انجام می‌دهند؛ E، تعداد مراحل آموزشی که هر گره در هر دور روی مجموعه داده محلی خود انجام می‌دهد؛ و B، اندازه دسته محلی که برای به‌روزرسانی‌های هر گره استفاده می‌شود. در اینجا $B = \infty$ انتخاب می‌شود تا نشان دهد که کل مجموعه داده محلی به عنوان یک دسته واحد در نظر گرفته می‌شود. بنابراین، به عنوان یک نمونه از این الگوریتم گسترده شده جدید، انتخاب $B = \infty$ و $E = 1$ باعث می‌شود که این روش دقیقاً مانند FedSGD عمل کند. همچنین برای یک گره با n_k نمونه محلی، تعداد به‌روزرسانی‌های محلی در هر دور با $u_k = E\frac{n_k}{B}$ نمایش داده می‌شود [۹]. شبه کد کامل این روش در الگوریتم ۲-۱ ارائه شده است. علاوه بر این، تمامی نمادهای مورد استفاده در این الگوریتم در جدول ۲-۱ توضیح داده شده است. هدف از این جدول، فراهم کردن درکی جامع از نحوه عملکرد و پیاده‌سازی الگوریتم می‌باشد.

۲-۴-۲ بهینه‌سازی FedProx

روش FedProx به بررسی چالش‌های یادگیری فدرال در بسترهای ناهمگن می‌پردازد. این روش با ایجاد تغییرات جزئی در روش موجود FedAvg، به بهبود پایداری و دقت در شبکه‌های ناهمگن کمک می‌کند. این تغییرات شامل اضافه کردن یک عبارت نزدیک مبدا[1] به تابع هدف است که به صورت اصولی به سرور کمک می‌کند تا ناهمگنی را مدیریت کند [۱۷].

رابطه هدف FedProx به صورت زیر تعریف می‌شود:

$$\min_w f(w) = \min_w \sum_{k=1}^{K} \frac{n_k}{n} \left(F_k(w) + \frac{\mu}{2} \|w_t - w_t^k\|^2 \right) \qquad (۵-۲)$$

[1] Proximal Term

الگوریتم ۲-۱: میانگین‌گیری فدرال (FedAvg) [۹]

initialize w_0;
for *each round* $t = 1, 2, \dots, T$ **do**
  $m \leftarrow \max(C \cdot K, 1)$;
  $U_t \leftarrow$ (random set of m clients);
  for *each client* $k \in U_t$ ***in parallel*** **do**
    $w_{t+1}^k \leftarrow \texttt{ClientUpdate}(k, w_t)$;
  end
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$;
end
Function $\texttt{ClientUpdate}(k, w)$: *// Run on client* k
  $\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B);
  for *each local epoch* i *from* 1 *to* E **do**
    for *batch* $b \in \mathcal{B}$ **do**
      $w \leftarrow w - \eta \nabla \ell(w; b)$;
    end
  end
  return w to server;
end

جدول ۲-۱: نمادهای الگوریتم FedAvg

متغیر	توضیحات
w	وزن‌های شبکه‌عصبی
T	تعداد گام‌ها
K	تعداد مشتریان
C	ضریب کنترلی برای زیرمجموعه‌ای از مشتریان
m	زیر مجموعه‌ای از مشتریان
U_t	مجموعه‌ای نامرتب از m در گام t
n	تعداد داده‌های آموزشی
$\mathcal{P}_k$	مجموعه داده‌های متعلق به مشتری k
B	اندازه دسته محلی
E	تعداد مراحل آموزش محلی
η	نرخ یادگیری

در رابطه (۲-۵) بخش $\frac{\mu}{2}\|w_t - w_t^k\|^2$، همان عبارت نزدیک مبدا است که به تابع هدف اضافه شده است. همچنین μ، یک پارامتر تنظیم برای این عبارت به حساب می‌آید و در نهایت w_t^k وزن‌های مدل محلی دستگاه k در تکرار t است.

حال با توجه به رابطه (۲-۵)، به‌روزرسانی وزن‌ها به شکل زیر تغییر پیدا خواهد کرد و بخش $\mu(w_t - w_t^k)$، گرادیان عبارت نزدیک مبدا است.

$$w_{t+1} = w_t - \eta(\nabla F_k(w_t) + \mu(w_t - w_t^k)) \qquad (۲-۶)$$

بنابراین، به‌روزرسانی‌های محلی در هر گام با به‌روزرسانی سراسری مرحله قبل مرتبط هستند. عبارت نزدیک مبدا به عنوان یک مکانیزم منظم‌کننده[1] عمل می‌کند که تفاوت‌های بین وزن‌های جهانی w و وزن‌های محلی w_t^k

[1] Regularization

را کاهش می‌دهد. این ترم به کاهش تاثیرات منفی ناهمگنی سیستم‌ها و داده‌ها کمک می‌کند و باعث پایداری بیشتر در فرآیند همگرایی می‌شود [۱۷].

فصل سوم

بررسی اختصاصی پیشینه روش‌های حل مشکل ناهمگنی آماری

۳-۱ مقدمه

همان‌طور که در فصل گذشته اشاره شد، یکی از مهم‌ترین مشکلات در حوزه یادگیری فدرال، مسئله داده‌های غیرمستقل و غیریکنواخت (non-IID) است که منجر به بروز چالش‌ها و ناهمگنی‌های آماری می‌شود. این مشکل باعث می‌شود که مدل‌های یادگیری نتوانند به خوبی از داده‌های توزیع‌شده استفاده کنند و کارایی مطلوبی داشته باشند. به دلیل اهمیت بالای این موضوع، محققان بسیاری تلاش‌های گسترده‌ای برای حل این مشکل انجام داده‌اند.

مبحث اصلی این پایان‌نامه نیز به طور دقیق به همین مسئله اشاره دارد و به دنبال یافتن راه‌حلی مؤثر برای مقابله با داده‌های non-IID است. در ادامه، به صورت خلاصه به بررسی راه‌حل‌هایی که تاکنون برای حل این مشکل مطرح شده‌اند، خواهیم پرداخت تا تصویر جامعی از تلاش‌های انجام شده در این زمینه ارائه دهیم. همچنین باید توجه داشت که هر یک از این راه‌حل‌ها نقاط قوت و ضعف خاص خود را دارند و بسته به شرایط و نوع داده‌ها، می‌توانند نتایج متفاوتی را به همراه داشته باشند. بررسی دقیق این راه‌حل‌ها و ارزیابی کارایی آن‌ها می‌تواند به بهبود سیستم‌های یادگیری فدرال و غلبه بر مشکلات مرتبط با داده‌های غیرمستقل و غیر یکنواخت کمک شایانی کند.

۳-۲ نگرش برپایه داده

۳-۲-۱ اشتراک‌گذاری داده

مشکل اصلی الگوریتم FedAvg در مواجهه با داده‌های غیرمستقل و غیریکنواخت، تفاوت وزن‌های اولیه در شروع فرآیند آموزش است. این تفاوت‌ها می‌توانند باعث شوند که مدل‌های محلی در هر گره به طور قابل توجهی متفاوت از یکدیگر باشند، که در نتیجه منجر به مشکلات همگرایی و کاهش کارایی مدل نهایی می‌شود.

برای رفع این مشکل، روشی پیشنهاد شده است که در آن ابتدا سرور مرکزی مقدار کمی از داده‌ها را به صورت محلی آموزش می‌دهد. در این مرحله، سرور مرکزی با استفاده از این داده‌ها، یک مدل اولیه را آموزش داده و وزن‌های اولیه آن را تنظیم می‌کند. سپس، این وزن‌های اولیه به همراه داده‌های آموزش دیده شده به تمامی کاربران ارسال می‌شود. این اقدام باعث می‌شود که تمام کاربران در ابتدای فرآیند آموزش با مجموعه‌ای از داده‌های مشترک و وزن‌های اولیه مشابه روبه‌رو شوند.

نقطه قوت این روش در این است که به دلیل انجام این عملیات تنها در آغاز فرآیند آموزش، هزینه زیادی به شبکه تحمیل نمی‌شود. در واقع، انتقال داده‌ها و وزن‌ها فقط در ابتدا انجام شده و پس از آن کاربران به صورت مستقل به آموزش مدل‌های محلی خود ادامه می‌دهند. این اقدام منجر به کاهش اختلافات ناشی از ناهمگنی داده‌ها شده و فرآیند همگرایی مدل نهایی سریع‌تر و با دقت بیشتری انجام می‌شود [۱۸].

در شکل ۳-۱، نحوه اجرای این روش و مراحل مختلف آن به تصویر کشیده شده است. این تصویر نشان می‌دهد که چگونه سرور مرکزی ابتدا داده‌های کمی را آموزش می‌دهد، وزن‌های اولیه را تنظیم می‌کند و سپس این وزن‌ها و داده‌ها را به کاربران ارسال می‌کند تا فرآیند آموزش محلی با یک نقطه شروع مشترک برای همه کاربران آغاز شود.

یکی دیگر از روش‌های مطرح شده در زمینه یادگیری فدرال به این صورت است که کاربران بتوانند نتایج آموزش تعدادی داده اشتراکی را با یکدیگر به اشتراک بگذارند و از نتایج دیگر کاربران بر روی این داده‌های اشتراکی مطلع شوند. در این روش، کاربران نتایج به‌دست آمده از آموزش داده‌های مشترک را با هم مبادله می‌کنند، که این کار منجر به بهبود عملکرد مدل‌های محلی و در نهایت مدل سراسری می‌شود [۱۹].

براساس بررسی‌های انجام شده، برای مثال در مجموعه داده‌ای تنها با ۶۰,۰۰۰ داده ورودی، اگر حدود ۵ درصد از داده‌ها به صورت اشتراکی در اختیار کاربران قرار گیرد، دقت مدل تا حدود ۳۰ درصد افزایش خواهد یافت. این افزایش دقت به دلیل همگرایی بهتر مدل‌ها و کاهش تفاوت‌های آماری بین داده‌های محلی است. به عبارتی دیگر، این روش کمک می‌کند که مدل‌ها با یکدیگر هماهنگ‌تر شوند و نتایج دقیق‌تری ارائه دهند.

با این حال، باید توجه داشت که اشتراک‌گذاری داده‌ها بین کاربران می‌تواند مسائل حریم شخصی را به

همراه داشته باشد. به عبارت دیگر، هنگامی که داده‌های اشتراکی بین کاربران مبادله می‌شود، احتمال نقض حریم شخصی کاربران افزایش می‌یابد. بنابراین، هنگام پیاده‌سازی این روش، ضروری است که اقدامات لازم برای حفظ حریم شخصی کاربران به طور جدی مد نظر قرار گیرد. این اقدامات می‌تواند شامل استفاده از تکنیک‌های رمزنگاری، ناشناس‌سازی داده‌ها، یا روش‌های دیگر برای محافظت از اطلاعات حساس کاربران باشد [۳].

در نهایت، روش به اشتراک‌گذاری داده‌ها بین کاربران، اگرچه می‌تواند به بهبود دقت و کارایی مدل‌ها کمک کند، اما نیازمند دقت و توجه ویژه‌ای به مسائل حریم شخصی است. پژوهشگران و توسعه‌دهندگان باید با در نظر گرفتن این چالش‌ها، راهکارهایی را برای حفظ امنیت و حریم شخصی کاربران در هنگام اجرای این روش‌ها ارائه دهند.

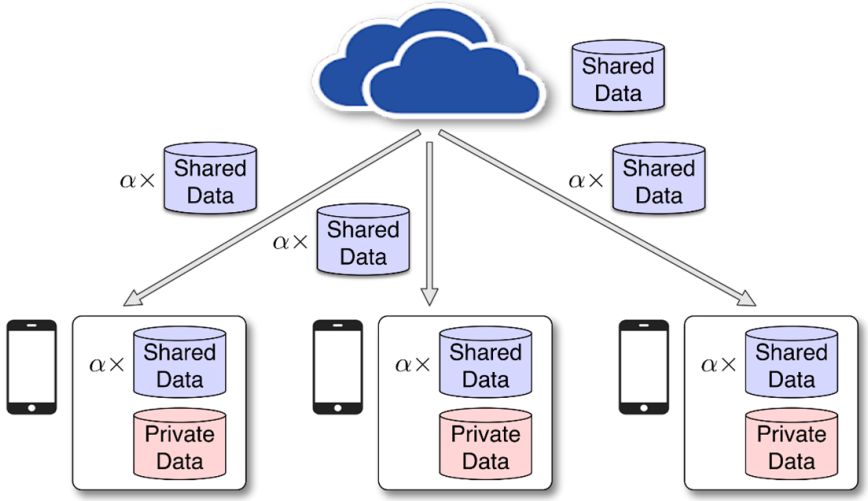
۳-۲-۲ بهبود داده[1]

ابتدا، کاربران تعدادی از داده‌های خود را به سمت سرور ارسال می‌کنند. سرور، با استفاده از داده‌های دریافتی، یک مدل شبکه مولد رقابتی[2] ایجاد می‌کند و این مدل را برای تمامی کاربران ارسال می‌نماید. کاربران با استفاده از این شبکه مولد رقابتی و با توجه به داده‌های خود، تعدادی داده جدید تولید کرده و در مراحل بعدی آموزش از این داده‌ها نیز استفاده می‌کنند. به این ترتیب، شبکه مولد رقابتی به کاربران کمک می‌کند تا داده‌های بیشتری برای آموزش مدل‌های خود در اختیار داشته باشند و از این داده‌ها برای بهبود عملکرد مدل‌های خود استفاده کنند. در شکل ۳-۲ نحوه عملکرد این روش به تصویر کشیده شده است.

این روش، به دلیل استفاده از تکنیک‌های رمزگذاری[3] و رمزگشایی[4] داده‌ها، نسبت به روش‌های اشتراک‌گذاری داده‌ها از نظر حفظ حریم شخصی کاربران بهتر عمل می‌کند. به این معنی که، به جای ارسال داده‌های خام کاربران به سرور یا دیگر کاربران، از داده‌های تولید شده توسط شبکه مولد رقابتی استفاده می‌شود که احتمال نقض حریم شخصی را کاهش می‌دهد.

استفاده از تکنیک‌های رمزگذاری و رمزگشایی داده‌ها در این روش، باعث می‌شود که داده‌های حساس کاربران در طول فرآیند آموزش، به صورت امن باقی بمانند. به عبارت دیگر، حتی اگر داده‌ها در طول انتقال یا در سرور مورد دسترسی غیرمجاز قرار گیرند، به دلیل رمزگذاری، اطلاعات واقعی کاربران فاش نخواهد شد. این ویژگی، امنیت و حریم شخصی کاربران را به طور قابل توجهی افزایش می‌دهد و از اطلاعات حساس آنان در برابر تهدیدات محافظت می‌کند [۲۰].

---
[1] Data Enhancement
[2] Generative Adversarial Network (GAN)
[3] Encoding
[4] Decoding



شکل ۳-۱: نمایش نحوه به اشتراک گذاری داده [۱۸].

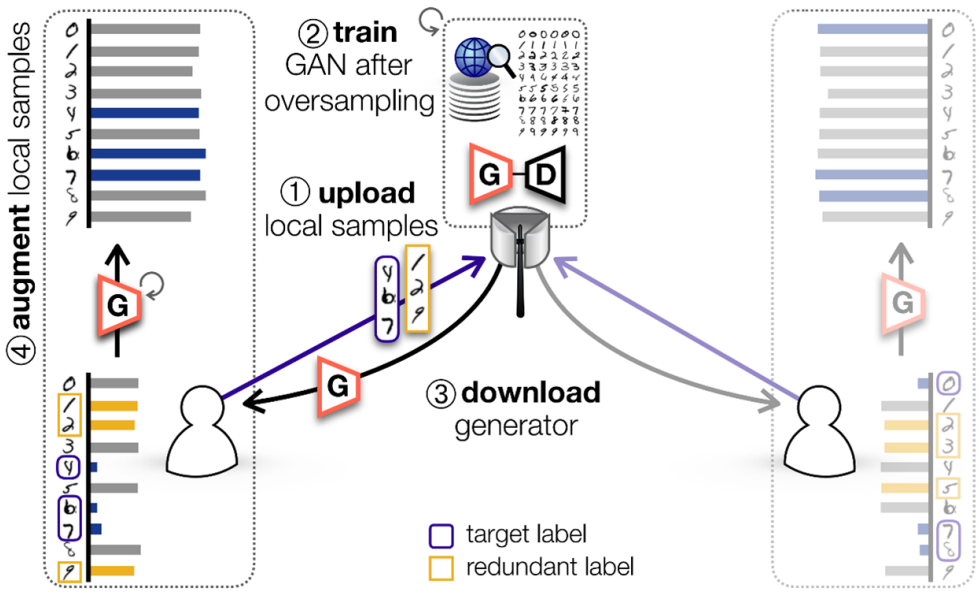
بنابراین، روش‌های بهبود داده شده که مبتنی بر رمزگذاری و رمزگشایی داده‌ها هستند، نه تنها به بهبود عملکرد مدل‌های یادگیری کمک می‌کنند، بلکه از حریم شخصی کاربران نیز حفاظت می‌نمایند. این ترکیب از امنیت و کارایی، این روش‌ها را به گزینه‌های مناسبی برای استفاده در سیستم‌های یادگیری فدرال تبدیل کرده است.

۳-۲-۳ انتخاب داده

در هنگام انتخاب کاربران برای فرآیند آموزش، می‌توان از الگوریتم‌هایی که بر پایه کیفیت داده‌ها عمل می‌کنند، استفاده نمود. به عبارت دیگر، می‌توان از الگوریتم حریصانه کوله‌پشتی برای اولویت‌بندی کاربران بهره برد، به نحوی که کاربران با داده‌های غنی و گسترده‌تر، اولویت بالاتری جهت انتخاب داشته باشند. این رویکرد به بهبود کیفیت آموزش کمک می‌کند، زیرا داده‌های با کیفیت بالاتر تاثیر مثبتی بر نتایج نهایی مدل خواهند داشت [۲۱].

علاوه بر این، می‌توان از روش‌های یادگیری عمیق برای تخمین زمان اجرای مدل در سمت کاربران استفاده کرد. این روش‌ها می‌توانند زمان مورد نیاز برای اجرای مدل را پیش‌بینی کنند و بر اساس این پیش‌بینی، از بین ویژگی‌های مختلف جهت آموزش، تنها آن‌هایی را انتخاب نمایند که تاثیر بیشتری بر خروجی خواهند داشت. به این ترتیب، با بهینه‌سازی انتخاب ویژگی‌ها، می‌توان زمان و منابع محاسباتی را به شکل موثرتری مدیریت کرد.

یکی از نکات کلیدی در استفاده از این روش‌های انتخاب داده این است که هیچ کدام از آن‌ها تغییری بر روی داده‌ها و کاربران ایجاد نمی‌کنند. به عبارت دیگر، این روش‌ها به گونه‌ای طراحی شده‌اند که داده‌های موجود و وضعیت کاربران بدون تغییر باقی می‌مانند، اما فرآیند انتخاب و استفاده از داده‌ها بهینه‌تر و کارآمدتر می‌شود. این ویژگی، استفاده از این راه‌حل‌ها را در برنامه‌های مختلف بسیار کاربردی و موثر می‌سازد [۲۲].



شکل ۳-۲: استفاده از شبکه مولد رقابتی جهت تولید داده [۲۰].

در نتیجه، استفاده از الگوریتم‌های مبتنی بر کیفیت داده‌ها و روش‌های یادگیری عمیق برای تخمین زمان اجرا، می‌تواند به طور قابل توجهی فرآیند آموزش در سیستم‌های یادگیری فدرال را بهبود بخشد. این روش‌ها نه تنها کیفیت داده‌های مورد استفاده را افزایش می‌دهند، بلکه با بهینه‌سازی منابع محاسباتی و زمان اجرا، کارایی سیستم را نیز بهبود می‌بخشند. این ترکیب از بهینه‌سازی داده‌ها و مدیریت منابع، به ویژه در محیط‌های با منابع محدود، اهمیت ویژه‌ای دارد و می‌تواند به نتایج بهتری در آموزش مدل‌ها منجر شود.

۳-۳ نگرش برپایه مدل

۳-۳-۱ تجمیع و به‌روزرسانی مدل[1]

هنگام اجرای الگوریتم در مراحل میانی، می‌توان با استفاده از ساختار شبکه‌های عصبی عمیق موجود، تفاوت گره‌های شبکه بین کاربران مختلف را بررسی نمود. این بررسی به ما امکان می‌دهد تا ساختار مدل اصلی را بر اساس تفاوت‌ها و ویژگی‌های مختلف کاربران، بهبود بخشیم و در نتیجه مدل کارآمدتری ایجاد کنیم. این فرآیند می‌تواند به بهینه‌سازی عملکرد مدل و افزایش دقت آن در مراحل بعدی کمک کند [۲۳].

روش دیگری برای بهبود عملکرد یادگیری فدرال این است که هم در سمت سرور و هم در سمت کاربران چندین مدل شبکه عصبی قرار داده شود. این شبکه‌ها به صورت جداگانه آموزش داده شده و به‌روزرسانی می‌شوند. پس از چند مرحله آموزش، می‌توان با استفاده از الگوریتم‌های تطابق بهترین، شبکه‌ها را با یکدیگر ترکیب کرد. این رویکرد به بهبود عملکرد کلی مدل کمک می‌کند و باعث می‌شود تا مدل نهایی از ویژگی‌ها و مزایای چندین

[1] Model Update and Aggregation

شبکه عصبی بهره‌مند شود [۲۴]. در شکل ۳-۳ نحوه عملکرد این روش به تصویر کشیده شده است.

همچنین، مکانیزم یادگیری فدرال نیمه‌ـناهمزمان[1] نیز یکی دیگر از روش‌های موثر در این حوزه است. در این روش، مدل‌های کاربران به ترتیبی که به سرور می‌رسند به‌روزرسانی می‌شوند. این رویکرد به خوبی با کاربران کند[2] که ممکن است در گردش‌های مختلف به سرور بپیوندند، سازگار است. با به‌روزرسانی و ترکیب مدل‌ها در مراحل مختلف، این مکانیزم به خوبی می‌تواند توازن را برای داده‌های ناهمگن برقرار کند و عملکرد مدل را بهینه سازد [۲۵، ۳].

در نهایت، با استفاده از این رویکردها و الگوریتم‌ها می‌توان به طور موثرتری با چالش‌های موجود در یادگیری فدرال مقابله کرد و مدل‌هایی با دقت و کارایی بالاتر ایجاد نمود. این روش‌ها نه تنها به بهبود ساختار مدل‌ها کمک می‌کنند، بلکه باعث می‌شوند تا فرآیند آموزش بهینه‌تر و سازگارتر با تنوع و ناهمگنی داده‌ها انجام شود.

۳-۳-۲ بهینه‌سازی تطبیقی[3]

الگوریتم پیش‌بینی میزان‌کار به گونه‌ای طراحی شده است که به صورت خودکار اطلاعات جامعی از سابقه آموزش هر کاربر را جمع‌آوری می‌کند. این اطلاعات شامل عملکرد کاربر در مراحل قبلی آموزش است. سپس بر اساس این سوابق، میزان پیچیدگی الگوریتم برای مرحله بعدی آموزش تعیین می‌شود تا برای کاربر مربوطه مناسب باشد . این رویکرد به بهینه‌سازی فرآیند آموزش کمک می‌کند و موجب می‌شود تا الگوریتم‌ها به شکل موثرتری با توانایی‌های هر کاربر هماهنگ شوند [۲۶].

یکی از روش‌های اولیه در بهینه‌سازی تطبیقی، استفاده از روش کاهش نرخ یادگیری است. در این روش، نرخ یادگیری برای هر کاربر به طور جداگانه و بر اساس عملکرد گذشته وی تعیین می‌شود. این به معنای آن است که کاربران با عملکرد بهتر ممکن است نرخ یادگیری بالاتری داشته باشند، در حالی که برای کاربرانی که با مشکلاتی مواجه بوده‌اند، این نرخ کاهش می‌یابد تا فرآیند یادگیری بهبود یابد [۲۷].

در طول سال‌های اخیر، بهینه‌سازی تطبیقی نشان داده است که می‌تواند تاثیر قابل‌توجهی بر بهبود عملکرد الگوریتم‌ها داشته باشد. به همین دلیل، محققان به سمت توسعه روش‌هایی رفته‌اند که امکان تغییر و تطبیق پارامترهای الگوریتم را در طول زمان فراهم کنند. این رویکرد باعث می‌شود تا هر کاربر بتواند در مراحل مختلف آموزش، پارامترهای مربوط به الگوریتم را متناسب با نیازها و شرایط خود تنظیم کند. این انعطاف‌پذیری به الگوریتم‌ها کمک می‌کند تا با گذشت زمان کارایی بیشتری داشته باشند و به طور خاص‌تر با شرایط و نیازهای کاربران سازگار شوند.

---

[1] Semi-Asynchronous
[2] Stragglers
[3] Adaptive Optimization

به طور کلی، استفاده از الگوریتم‌های پیش‌بینی و بهینه‌سازی تطبیقی می‌تواند به شکل چشم‌گیری کیفیت آموزش و کارایی سیستم‌های یادگیری را بهبود بخشد. این روش‌ها با فراهم کردن امکان تنظیم پارامترهای آموزشی بر اساس سوابق و عملکرد کاربران، موجب می‌شوند تا فرآیند یادگیری به شکل دقیق‌تر و موثرتری انجام شود. در نتیجه، کاربران می‌توانند از تجربیات گذشته خود بهره ببرند و با شرایط بهتر و مناسبتری به یادگیری ادامه دهند.

۳-۳-۳ بهینه‌سازی منظم

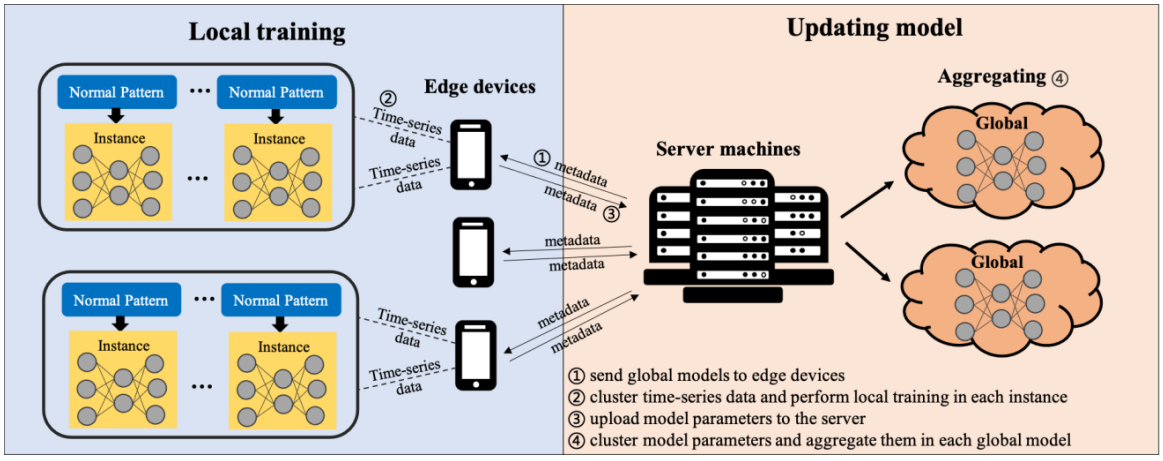
از مهم‌ترین و پرکاربردترین روش‌های موجود جهت کنترل داده‌های غیرمستقل و غیریکنواخت، رویکردهای بهینه‌سازی منظم هستند. این رویکردها با هدف بهبود فرآیند یادگیری و کاهش نوسانات ناشی از تفاوت در توزیع داده‌ها به کار گرفته می‌شوند. به عنوان مثال، یکی از روش‌های متداول در این زمینه، در نظر گرفتن نزدیک‌ترین همسایه است که طی آن تابع بهینه‌سازی محلی برای هر کاربر به‌روزرسانی می‌شود تا از نوسانات زیاد جلوگیری کند و هماهنگی بیشتری بین داده‌های مختلف کاربران ایجاد شود [۱۷].

یکی دیگر از روش‌های معروف در این زمینه، مکانیزم استاد–دانشجو[1] است. در این روش، یک مدل به عنوان استاد و مدل‌های دیگر به عنوان دانشجو عمل می‌کنند. گرادیان‌ها برای هر کاربر توسط یک جمله اضافه شده به نام جمله منظم‌سازی[2] تنظیم می‌شود. این جمله منظم‌سازی به منظور کاهش خطاها و بهبود دقت مدل‌ها افزوده می‌شود و از بیش‌برازش[3] جلوگیری می‌کند [۲۸].

رویکردهای بهینه‌سازی منظم، همان‌طور که در حوزه‌های مختلف یادگیری ماشین و یادگیری عمیق توانسته‌اند کارایی خود را به اثبات برسانند، در یادگیری فدرال نیز عملکرد بسیار خوبی دارند. این رویکردها با تنظیم مدل‌ها به گونه‌ای که نوسانات ناشی از داده‌های مختلف را کاهش دهند، به بهبود عملکرد کلی سیستم کمک می‌کنند. همچنین، با جلوگیری از بیش‌برازش، مدل‌ها را به سمت تعمیم بهتر هدایت می‌کنند، که این امر در محیط‌هایی با داده‌های غیرمستقل و غیریکنواخت بسیار حیاتی است.

در مجموع، استفاده از روش‌های بهینه‌سازی منظم در یادگیری فدرال نه تنها باعث بهبود دقت مدل‌ها می‌شود، بلکه موجب می‌گردد تا فرآیند یادگیری با پایداری و کارایی بیشتری انجام شود. این رویکردها به دلیل توانایی‌شان در کنترل نوسانات و کاهش خطاها، از ابزارهای اساسی در یادگیری فدرال به شمار می‌آیند و به توسعه مدل‌های دقیق و قابل اعتماد کمک می‌کنند.

---
[1] Teacher-Student
[2] Regularization Term
[3] Overfitting



شکل ۳-۳: چارچوب یک سیستم یادگیری فدرال چندمحلی و چندمرکزی برای کشف ناهنجاری‌ها [۲۴].

۳-۴ نگرش برپایه چهارچوب

۳-۴-۱ خوشه‌بندی مشابهت[1]

خوشه‌بندی یکی از روش‌های بسیار پرکاربرد و مهم در حوزه یادگیری ماشین است که ایده‌های آن می‌توانند در یادگیری فدرال نیز مورد استفاده قرار گیرند. در این روش، هنگامی که کاربران مدل‌های خود را آموزش داده و به سرور ارسال می‌کنند، سرور بر اساس مدل‌های دریافتی شباهت‌های آن‌ها را بررسی کرده و کاربرانی که مدل‌های مشابه دارند را در یک خوشه قرار می‌دهد. این فرایند به سرور امکان می‌دهد تا در مراحل بعدی، مدل یکسانی را برای اعضای هر خوشه ارسال کند. این رویکرد باعث می‌شود که مدل‌های آموزش‌دیده شده توسط کاربران با داده‌های مشابه، به طور همزمان و هماهنگ بهبود یابند و از همگرایی بهتری برخوردار شوند [۲۹].

به طور معمول، پس از چندین دوره آموزشی، فرآیند خوشه‌بندی مجدداً تکرار می‌شود تا از به‌روزرسانی‌های جدید و تغییرات احتمالی در داده‌ها و مدل‌ها بهره‌برداری شود. در شکل ۳-۴، حالت کلی خوشه‌بندی شباهت در سیستم‌های فدرال به تصویر کشیده شده است.

با وجود تمام مزایایی که روش خوشه‌بندی مشابهت به همراه دارد، یکی از مهم‌ترین مشکلات آن هزینه بالای ارتباطات است. در این روش، نیاز است که ساختار خوشه‌بندی در مراحل مختلف ارسال و دریافت شود، که این فرایند می‌تواند هزینه زیادی را بر شبکه اعمال کند. به خصوص در محیط‌هایی با تعداد زیاد کاربران و داده‌های بزرگ، این هزینه‌ها به طور قابل توجهی افزایش می‌یابد و می‌تواند عملکرد کلی سیستم را تحت تأثیر قرار دهد.

بنابراین، در حالی که خوشه‌بندی شباهت می‌تواند کارایی و دقت یادگیری فدرال را بهبود بخشد، باید به دقت هزینه‌های ارتباطی آن نیز مورد ارزیابی قرار گیرد و در صورت امکان، بهینه‌سازی‌های لازم انجام شود تا این هزینه‌ها کاهش یابند. به کارگیری روش‌های بهینه‌سازی ارتباطات و فشرده‌سازی داده‌ها می‌تواند در این زمینه

[1] Similarity Clustering

مفید باشد و به حفظ تعادل بین کارایی و هزینه‌ها کمک کند.

۳-۴-۲ دانش تقطیر[1]

به‌طور کلی، در روش‌های دانش تقطیر، هدف اصلی ساده‌سازی مدل‌های پیچیده و ارائه مدل‌هایی ساده اما کارآمد است. یکی از الگوریتم‌های مهم در این زمینه DS-FL[2] است. این الگوریتم با استفاده از مجموعه داده‌های بدون برچسب، به جای ارسال پارامترهای مدل، تنها خروجی مدل محلی را به اشتراک می‌گذارد. این روش به‌خصوص برای داده‌های غیرمستقل و غیریکنواخت بسیار مؤثر عمل می‌کند و نتایج مطلوبی به همراه دارد.

یکی از مهم‌ترین مزایای استفاده از دانش تقطیر، کاهش چشمگیر سربار شبکه است. به دلیل اینکه در این روش به جای ارسال پارامترهای مدل‌های محلی، فقط خروجی نهایی مدل‌ها ارسال می‌شود، حجم داده‌های ارسالی به طور قابل توجهی کاهش می‌یابد. این کاهش حجم داده‌ها نه تنها هزینه‌های ارتباطی را پایین می‌آورد بلکه سرعت پردازش و به‌روزرسانی مدل‌ها را نیز افزایش می‌دهد. به این ترتیب، بهره‌وری سیستم بهبود یافته و توان محاسباتی به نحو بهتری مدیریت می‌شود.

در روش DS-FL، ابتدا هر کاربر محلی با استفاده از داده‌های خود، مدلی را آموزش می‌دهد. سپس به جای ارسال پارامترهای مدل به سرور مرکزی، تنها خروجی مدل روی داده‌های بدون برچسب به اشتراک گذاشته می‌شود. سرور مرکزی با تجمیع این خروجی‌ها، یک مدل جهانی به‌روز شده را ایجاد می‌کند و آن را برای کاربران ارسال می‌کند. این فرایند تکرار می‌شود تا مدل جهانی به بهینه‌ترین حالت ممکن برسد [۳۰].

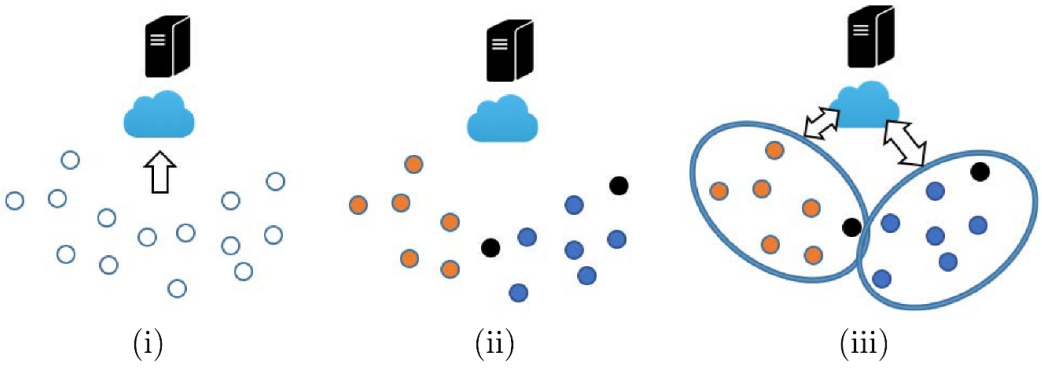
به کارگیری دانش تقطیر در یادگیری فدرال نه تنها به بهبود کارایی شبکه کمک می‌کند، بلکه امنیت و حریم خصوصی داده‌ها را نیز افزایش می‌دهد. چون خروجی مدل‌ها اغلب اطلاعات حساس کمتری نسبت به پارامترهای مدل در خود دارند، احتمال افشای اطلاعات شخصی کاربران کاهش می‌یابد. این ویژگی به‌خصوص در محیط‌هایی که حفظ حریم خصوصی کاربران اولویت بالایی دارد، از اهمیت ویژه‌ای برخوردار است.

به‌طور خلاصه، روش‌های دانش تقطیر مانند DS-FL با هدف ساده‌سازی مدل‌های پیچیده و کاهش هزینه‌های ارتباطی، به بهبود کارایی و امنیت در سیستم‌های یادگیری فدرال کمک می‌کنند. این روش‌ها با ارسال خروجی‌های مدل به جای پارامترها، سربار شبکه را کاهش داده و به تطبیق بهتر مدل‌ها با داده‌های غیرمستقل و غیریکنواخت کمک می‌کنند.

---

[1] Knowledge Distillation

[2] Distillation-base Semi-supervised Federated Learning



شکل ۳-۴: روش خوشه‌بندی مشابهت [۲۹].

۳-۴-۳ لایه‌های شخصی‌سازی[1]

روش لایه‌های شخصی‌سازی شده به این شکل عمل می‌کنند که در ابتدا کاربران بر اساس معیارهایی مانند کارایی آموزش و سرعت اجرا به گروه‌های مختلفی تقسیم می‌شوند. سپس، این کاربران بر اساس معیارهای تعیین شده به صورت لایه‌ای مرتب می‌شوند. به این ترتیب، سرور هنگامی که مدل را به‌روزرسانی می‌کند و قصد دارد آن را در مرحله بعد به سمت کاربران ارسال نماید، سعی می‌کند کاربرانی را که در یک لایه مشترک حضور دارند انتخاب کند. این انتخاب به سرور امکان می‌دهد تا گردش به‌روزرسانی‌ها را با سرعت و کارایی هماهنگ‌تری به پایان برساند و عملکرد بهتری از سیستم بگیرد [۳۱].

یکی از نکات کلیدی در اجرای این روش، تعیین میزان حد و آستانه‌ای است که بر اساس آن، کاربران به لایه‌های مختلف تقسیم می‌شوند. این تقسیم‌بندی باید به گونه‌ای باشد که خروجی مدل بهینه باشد و کارایی سیستم حفظ شود. تعیین این حد و آستانه‌ها می‌تواند چالش‌برانگیز باشد و نیاز به سعی و خطا دارد تا بهترین ترکیب ممکن به دست آید.

به‌طور کلی، روش لایه‌های شخصی‌سازی شده با تقسیم‌بندی کاربران و مرتب‌سازی آن‌ها در لایه‌های مختلف، امکان بهبود هماهنگی و کارایی در گردش به‌روزرسانی‌ها را فراهم می‌کند. این رویکرد نه تنها باعث می‌شود که کاربران با سرعت مشابه در یک لایه قرار گیرند، بلکه به سرور کمک می‌کند تا با کاهش ناهماهنگی‌ها، به‌روزرسانی مدل‌ها را با کارایی بیشتری انجام دهد. انتخاب صحیح معیارهای تقسیم‌بندی و آستانه‌ها در این روش، از اهمیت بالایی برخوردار است و نیازمند تحلیل و ارزیابی دقیق است تا بهترین نتایج ممکن به دست آید.

[1] Personalization Layers

۳-۵ نگرش برپایه الگوریتم

۳-۵-۱ فرایادگیری

مدل ابتدایی فرایادگیری پیاده‌شده بر بستر یادگیری فدرال، در واقع از همان الگوریتم FedAvg بهره می‌برد و با ترکیب آن با روش فرایادگیری، تلاش دارد تا فرایند آموزش را بهینه‌سازی کرده و پارامترهای مناسب‌تری را به دست آورد [۳۲]. الگوریتم اولیه‌ـ دوگانه[1] (FedPD)، یکی از الگوریتم‌های کارا با استفاده از فرایادگیری است که حتی برای توابع غیرمحدب نیز مقاوم بوده و علاوه بر دستیابی به همگرایی مناسب، از نظر کاهش ارتباطات نیز بسیار کارآمد عمل می‌کند [۳۳].

روش‌های فرایادگیری به دلیل توانایی‌شان در هماهنگی سریع با داده‌های جدید و تغییر پارامترهای مربوطه، مورد توجه قرار گرفته‌اند. این روش‌ها می‌توانند به سرعت با شرایط جدید سازگار شوند و پارامترهای مدل را بهبود بخشند. با این حال، یکی از چالش‌های اصلی این روش‌ها مربوط به کاربران کند است. این کاربران ممکن است به دلیل محدودیت‌های سخت‌افزاری یا مشکلات ارتباطی، نتوانند به‌روزرسانی‌های سریع و هماهنگ را انجام دهند و این موضوع می‌تواند باعث اختلال در عملکرد مدل شود.

به طور کلی، مدل‌های فرایادگیری در بستر یادگیری فدرال با ترکیب روش‌های مختلف و بهره‌گیری از الگوریتم‌های بهینه‌سازی مانند FedAvg و FedPD، سعی دارند تا با بهبود فرآیندهای آموزش و کاهش هزینه‌های ارتباطی، به نتایج بهتری دست یابند. این روش‌ها با وجود چالش‌هایی که ممکن است در پیاده‌سازی و هماهنگی با کاربران کند داشته باشند، به دلیل قابلیت‌هایشان در بهینه‌سازی و هماهنگی سریع با داده‌های جدید، پتانسیل بالایی برای بهبود عملکرد سیستم‌های یادگیری فدرال دارند.

۳-۵-۲ یادگیری چندوظیفه‌ای

یادگیری چندوظیفه‌ای به این معناست که هر یک از کاربران شرکت‌کننده در فرآیند یادگیری فدرال، به دنبال یادگیری وظایف مختلفی هستند و تلاش می‌شود که در این مسیر، حریم شخصی کاربران به طور قابل‌توجهی حفظ شود. در یادگیری فدرال چندوظیفه‌ای، کاربران بر اساس داده‌های محلی خود، مدل را آموزش می‌دهند و نتایج آن را به سمت سرور مرکزی ارسال می‌کنند. سپس سرور، با تحلیل پارامترهای ارسال شده، روابط معناداری میان این مدل‌ها پیدا کرده و مدل به‌روز شده را دوباره به سمت کاربران بازمی‌گرداند [۳۴].

به عبارت دیگر، در این روش، هر کاربر ابتدا مدل را با استفاده از داده‌های محلی خود آموزش می‌دهد. این فرآیند موجب می‌شود که داده‌های شخصی کاربران از دستگاه‌های آنان خارج نشود و فقط نتایج به دست آمده از مدل‌های محلی به سرور ارسال شود. سرور مرکزی با جمع‌آوری این نتایج، به دنبال یافتن الگوها و روابطی است

[1] Primal-Dual

که بتواند مدل کلی را بهبود بخشد. این مدل بهبود یافته سپس به کاربران ارسال می‌شود تا مجدداً با داده‌های محلی آنان آموزش داده شود.

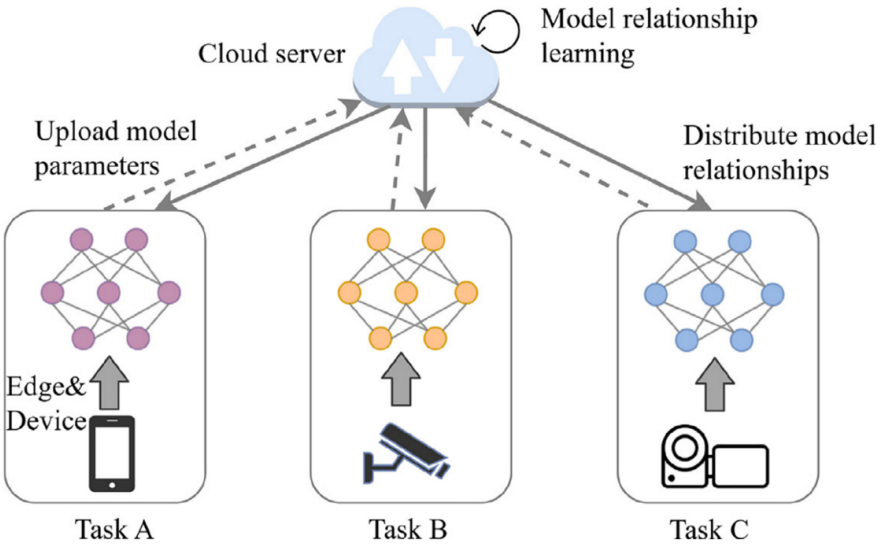
در شکل ۳-۵، نمای کلی از نحوه عملکرد یادگیری چندوظیفه‌ای در سیستم‌های فدرال به نمایش گذاشته شده است. این شکل به خوبی نشان می‌دهد که چگونه هر کاربر با استفاده از داده‌های محلی خود مدل را آموزش داده و نتایج را به سرور ارسال می‌کند و سرور با تحلیل این نتایج، مدل بهبود یافته را به کاربران بازمی‌گرداند.

به طور کلی، یادگیری چندوظیفه‌ای فدرال، به دلیل توانایی‌اش در تطبیق با داده‌های متنوع و محافظت از حریم خصوصی کاربران، یک رویکرد بسیار مؤثر و کارآمد در زمینه یادگیری فدرال محسوب می‌شود.

۳-۵-۳ یادگیری مادام‌العمر[1]

رویکرد اصلی یادگیری مادام‌العمر به این صورت است که تلاش می‌کند در هر مرحله از الگوریتم، کاربرانی که برای اجرا انتخاب می‌شوند را به خاطر بسپارد. همان‌طور که پیش‌تر مطرح شد، در یادگیری فدرال ممکن است در هر مرحله تعداد کمی از کاربران انتخاب شوند. این مسئله باعث می‌شود که وزن‌ها و مدل‌هایی که برای کاربران جدید ارسال می‌شوند، لزوماً کارایی لازم را نداشته باشند. الگوریتم یادگیری مادام‌العمر تلاش دارد تا کاربران را به‌خاطر بسپارد و مدل‌های متناسب با هر کدام را ایجاد و به سمت آن‌ها ارسال کند [۳۵].

این رویکرد به این صورت عمل می‌کند که در هر مرحله از یادگیری، سوابق کاربران انتخاب شده را ذخیره می‌کند و از این سوابق برای بهبود و تطبیق مدل‌های آینده استفاده می‌کند. به این ترتیب، زمانی که کاربر جدیدی



شکل ۳-۵: یادگیری فدرال چندوظیفه‌ای [۳].

[1] Life-Long Learning

وارد فرآیند یادگیری می‌شود، الگوریتم می‌تواند از اطلاعات ذخیره شده قبلی استفاده کند و مدل بهتری را برای او ارسال کند. این روش باعث می‌شود که مدل‌ها به مرور زمان بهینه‌تر شده و عملکرد بهتری داشته باشند.

یکی از نکات مهم در یادگیری مادام‌العمر، حفظ و به‌خاطر سپاری کاربران است. در زمینه یادگیری فدرال، این کار به دلیل تعداد بسیار زیاد کاربران ممکن است چالش‌برانگیز باشد. یادگیری فدرال به‌طور معمول با تعداد زیادی از کاربران سروکار دارد و حفظ سوابق همه این کاربران به‌طور همزمان می‌تواند منابع زیادی را مصرف کند و پیچیدگی‌های فنی زیادی را به همراه داشته باشد.

در نتیجه، یادگیری مادام‌العمر با ذخیره و استفاده از اطلاعات کاربران در طول زمان، می‌تواند به طور مؤثری به مدیریت چالش‌های مربوط به داده‌های غیرمستقل و غیریکنواخت در یادگیری فدرال کمک کند و همچنین به حفظ و بهبود کارایی مدل‌های یادگیری فدرال کمک نماید.

فصل چهارم

تعویض مدل‌های شبکه عصبی بین کاربران

۴-۱ مقدمه

در فصل پیشین، روش‌های متعددی برای حل مشکل داده‌های non-IID مورد بررسی قرار گرفتند. در این فصل، رویکرد جامعی برای مقابله با این چالش، یعنی تبادل مدل‌های شبکه عصبی میان کاربران نهایی، بررسی می‌شود که محور اصلی این پایان‌نامه را تشکیل می‌دهد. به منظور درک بهتر، ابتدا یک مثال از داده‌های non-IID مطرح خواهد شد.

فرض کنید هدف، آموزش مدلی برای تشخیص اشیا مانند علائم ترافیکی و علائم فروشگاهی است. اگر وسایل نقلیه به ترتیب در بزرگراه و مرکز شهر حرکت کنند، داده‌های ویدیویی آن‌ها توزیع‌های متفاوتی از این علائم خواهند داشت. به این معنا که داده‌های آموزشی جمع‌آوری شده از بزرگراه ممکن است کمتر شامل علائم فروشگاهی باشند، در حالی که داده‌های جمع‌آوری شده از مرکز شهر حاوی تعداد بیشتری از هر دو نوع علائم خواهند بود. این تفاوت در توزیع داده‌ها در دستگاه‌های نهایی می‌تواند باعث ایجاد مشکل انحراف وزن‌ها شود.

برای حل این مسئله، عملیات تعویض مدل‌های شبکه عصبی بین کاربران نهایی پیشنهاد می‌شود. این رویکرد، مدل‌ها را بین دستگاه‌های نهایی جابجا می‌کند تا تنوع داده‌ها در دستگاه‌های مختلف کاهش یابد. این عملیات بدون نیاز به هزینه‌های محاسباتی و ارتباطی اضافی، به بهبود عملکرد مدل در مواجهه با داده‌های non-IID کمک

می‌کند.

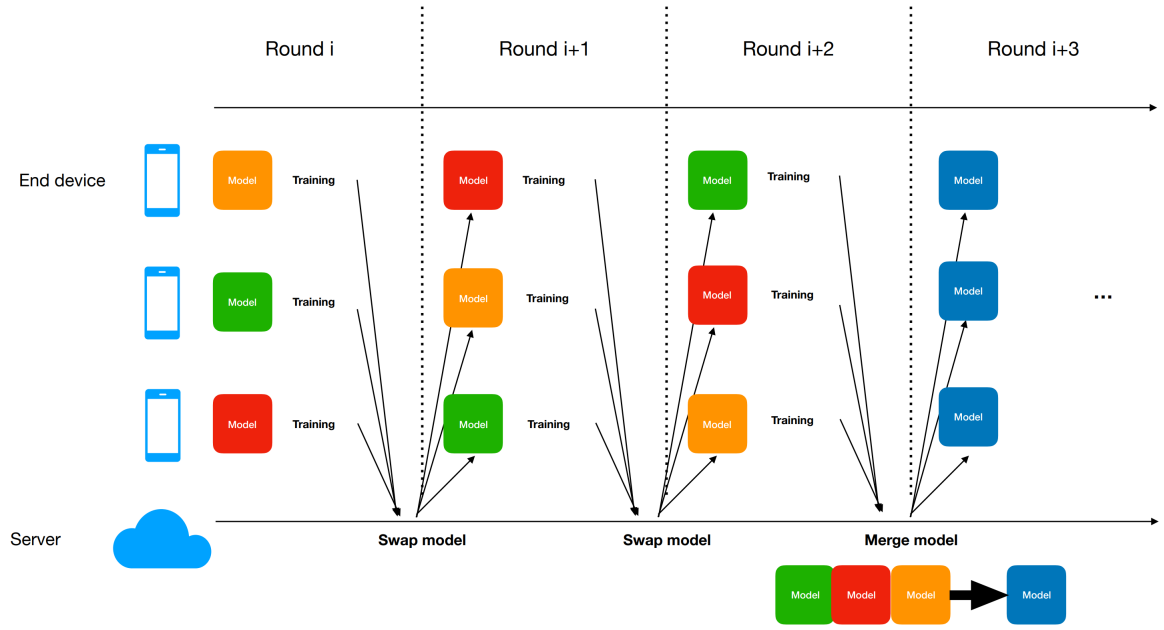
در این فصل ابتدا ...

۴-۲ روش تعویض فدرال[1]

در این روش، یک عملیات جدید به نام تعویض فدرال یا FedSwap پیشنهاد شده است که جایگزین برخی از دوره‌های FedAvg در سرور می‌شود. این عملیات با هدف بهبود فرآیند یادگیری فدرال و کاهش تاثیرات منفی داده‌های non-IID طراحی شده است. اصل اساسی FedSwap این است که به جای اجرای FedAvg در هر تکرار، به دستگاه‌های نهایی اجازه می‌دهد تا مدل‌های محلی خود را در سرور با یکدیگر تبادل کنند [۳۶].

در روش اصلی یادگیری فدرال، در هر تکرار یک مدل جهانی جدید از ترکیب مدل‌های هر دستگاه نهایی به دست می‌آید. همان‌طور که در شکل ۴-۱ نشان داده شده است، به جای اجرای FedAvg در هر تکرار، سرور می‌تواند به عنوان یک گزینه دیگر، مدل‌ها را بین دستگاه‌های نهایی تعویض کند. در این روش، به جای اینکه در هر تکرار فرآیند FedAvg انجام شود، دستگاه‌های نهایی اجازه دارند مدل‌های محلی خود را در سرور با یکدیگر مبادله کنند.

برای حفظ عدالت در این فرآیند، از یک استراتژی چرخشی استفاده می‌شود. در این استراتژی، به طور منظم و به ترتیب، دو دستگاه نهایی به یکدیگر اجازه می‌دهند که مدل‌های خود را تبادل کنند. این کار باعث می‌شود



شکل ۴-۱: روش تعویض فدرال [۳۶].

[1] Federated Swapping

که همه دستگاه‌های نهایی به طور مساوی در فرآیند تبادل مدل‌ها شرکت کنند و هیچ دستگاهی از مزایای این تبادل محروم نماند.

علاوه بر این، انتظار می‌رود که این عملیات تعویض مدل بین دستگاه‌های نهایی، به هر مدل دید گسترده‌تری از کل مجموعه داده‌ها بدهد. به عبارت دیگر، هر مدل محلی با تبادل مدل با دیگر دستگاه‌ها، می‌تواند اطلاعات بیشتری از داده‌های مختلف دریافت کند. این امر به کاهش انحراف وزن‌ها کمک می‌کند، زیرا مدل‌ها با داده‌های متنوع‌تری آموزش می‌بینند و به تدریج به یک مدل جامع‌تر و دقیق‌تر نزدیک می‌شوند.

به طور خلاصه، عملیات FedSwap با تبادل مدل‌های محلی بین دستگاه‌های نهایی، نه تنها به بهبود دقت و عملکرد مدل‌ها کمک می‌کند، بلکه مشکلات ناشی از داده‌های non-IID را نیز کاهش می‌دهد. این روش به عنوان یک رویکرد موثر در یادگیری فدرال می‌تواند باعث بهبود قابل توجهی در نتایج نهایی شود [۳۶].

جزئیات عملیات FedSwap در الگوریتم ۴-۱ ارائه شده است. همچنین در جدول ۴-۱ نمادهای مختص این الگوریتم به نمایش در آمده است. در این الگوریتم، w_t^k به عنوان وزن مدل در دستگاه نهایی k پس از گام t تنظیم می‌شود. در ابتدا، دستگاه‌های نهایی چندین به‌روزرسانی محلی انجام می‌دهند تا مدل‌های خود را بهبود بخشند. پس از هر h_1 به‌روزرسانی محلی، سرور وارد عمل شده و عملیات FedSwap را اجرا می‌کند. در این مرحله، مدل‌های محلی بین دستگاه‌های نهایی تبادل می‌شوند تا هر دستگاه بتواند از مدل‌های متنوع‌تری برای آموزش استفاده کند.

این تبادل مدل‌ها به کاهش تنوع داده‌ها بین دستگاه‌های مختلف کمک می‌کند و باعث می‌شود که مدل‌ها با داده‌های مختلفی آموزش ببینند. پس از انجام h_2 عملیات FedSwap، سرور وارد عمل شده و عملیات FedAvg را اجرا می‌کند. در این مرحله، سرور مدل‌های محلی را تجمیع می‌کند تا یک مدل مشترک ایجاد شود که از داده‌های تمام دستگاه‌ها بهره‌مند است.

برای تعیین مقادیر h_1 و h_2، ابتدا آزمایش‌های مختلفی انجام شده و بر اساس نتایج به دست آمده، بهینه‌ترین مقادیر انتخاب شده‌اند. در این آزمایش‌ها، چند نکته مهم مشاهده شده است. ابتدا، مقدار h_1 به عملکرد به‌روزرسانی مدل محلی در دستگاه‌های نهایی وابسته است. از آنجایی که وظیفه یادگیری معمولاً یک وظیفه عمومی مثل طبقه‌بندی است، مقدار h_1 بر اساس مقدار گام تعریف‌شده در روش میانگین‌گیری فدرال سنتی تنظیم می‌شود [۳۶].

علاوه بر این، مقدار h_2 نقش مهمی در توازن بین سربار ارتباطی و همگرایی مدل ایفا می‌کند. با افزایش مقدار h_2، تعداد دفعات تعویض فدرال بین دستگاه‌های نهایی بیشتر خواهد شد. این امر می‌تواند با کاهش تعداد دفعات ادغام فدرال، پهنای باند ارتباطی بیشتری را صرفه‌جویی کند. با این حال، این امر ممکن است باعث

الگوریتم ۴-۱: تعویض فدرال (FedSwap) [۳۶]

Initialize all clients model with weight w_0;
for $t = 1, 2, \ldots, T$ **do**
  for *each client* $k = 1, 2, \ldots, K$ ***in parallel*** **do**
    $w_t^k = w_{t-1}^k - \eta \nabla F(w_{t-1}^k)$;
  end
  if $t|h_1 = 0$ *and* $t|h_1 h_2 \neq 0$ **then**
    for *each client* $k = 1, 2, \ldots, K$ **do**
      $w_t^k \leftarrow \texttt{FedSwap}(k, \{w_t^k\}_{k \in K})$;
    end
  end
  if $t|h_1 h_2 = 0$ **then**
    $w_t \leftarrow \texttt{FedAvg}(\{w_t^k\}_{k \in K})$;
    for *each client* $k = 1, 2, \ldots, K$ ***in parallel*** **do**
      $w_t^k \leftarrow w_t$;
    end
  end
end
Function *FedSwap*$(k, \{w_t^k\}_{k \in K})$**:**
  r represent a random client in K;
  $w_t \leftarrow w_t^r$;
  $w_t^r \leftarrow w_t^k$;
  return w_t;
end
Function *FedAvg*$(\{w_t^k\}_{k \in K})$**:**
  $w_t \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_t^k$;
  return w_t;
end

جدول ۴-۱: نمادهای مختص الگوریتم FedSwap

متغیر	توضیحات
h_1	تعداد گام‌ها بین هر FedSwap
h_2	تعداد FedSwap بین هر FedAvg

افزایش احتمال انحراف وزن‌ها و کاهش دقت همگرایی مدل جهانی شود. به عبارت دیگر، هرچه مقدار h_2 بزرگتر باشد، تعداد دفعاتی که مدل‌ها بین دستگاه‌های نهایی تعویض می‌شوند بیشتر است و این ممکن است به بهبود عملکرد مدل‌ها در مواجهه با داده‌های non-IID کمک کند، اما ریسک انحراف وزن‌ها نیز بیشتر خواهد شد.

از سوی دیگر، اگر مقدار h_2 کوچکتر باشد، فراوانی تعویض فدرال بین دستگاه‌های نهایی کاهش می‌یابد. این امر منجر به افزایش سربارهای ارتباطی می‌شود زیرا نیاز به ادغام مکرر مدل جهانی خواهد بود. بنابراین، مقدار h_2 باید به گونه‌ای تنظیم شود که توازن مناسبی بین کاهش سربار ارتباطی و حفظ دقت مدل ایجاد کند.

در مجموع، روش FedSwap با تعویض مدل‌های محلی بین دستگاه‌های نهایی بدون نیاز به هزینه‌های محاسباتی و ارتباطی اضافی، می‌تواند به بهبود عملکرد مدل‌ها در مواجهه با داده‌های non-IID کمک کند

.[۳۶]

۳-۴ نحوه جابجایی مدل‌ها

۱-۳-۴ جابجایی تصادفی

در الگوریتم FedSwap، مدل‌ها بین دستگاه‌های نهایی جابجا می‌شوند. انتخاب دستگاه‌های نهایی جهت جابجایی به صورت کاملاً تصادفی انجام می‌گیرد، به این معنا که هنگامی که دو دستگاه می‌خواهند مدل‌های خود را جابجا کنند، فرآیند انتخاب این دو دستگاه به صورت کاملاً تصادفی انجام می‌شود. این تصادفی بودن انتخاب دستگاه‌ها باعث می‌شود که هیچ الگوی ثابتی در جابجایی مدل‌ها وجود نداشته باشد و هر بار ترکیب جدیدی از دستگاه‌ها در فرآیند تبادل مدل شرکت کنند.

یکی از ویژگی‌های مهم الگوریتم FedSwap این است که تمام دستگاه‌های نهایی به صورت مساوی و عادلانه در این فرآیند جابجایی شرکت می‌کنند. به عبارت دیگر، همه دستگاه‌ها بدون استثنا در فرآیند جابجایی قرار می‌گیرند، اما انتخاب دستگاه‌ها برای جابجایی به صورت تصادفی صورت می‌پذیرد. این روش باعث می‌شود که تمامی دستگاه‌ها فرصت مساوی برای تبادل مدل‌ها و بهبود دقت و عملکرد خود داشته باشند.

با استفاده از این رویکرد، الگوریتم FedSwap قادر است به بهبود عملکرد مدل‌های محلی کمک کند، زیرا تبادل تصادفی مدل‌ها بین دستگاه‌ها باعث می‌شود که هر دستگاه به داده‌ها و اطلاعات بیشتری دسترسی پیدا کند. این امر به کاهش انحراف وزن‌ها و بهبود همگرایی مدل جهانی کمک می‌کند.

۲-۳-۴ جابجایی بر اساس میزان مشابهت مدل‌ها

در این پژوهش، هدف اصلی انتخاب دستگاه‌های نهایی بر اساس میزان شباهت مدل‌های شبکه عصبی و جابجایی آن‌ها با یکدیگر است. برای این منظور، باید به طور کامل با ساختار شبکه عصبی آشنا بوده و مدل‌های مختلف را با یکدیگر مقایسه کرد. این مقایسه به ما کمک می‌کند تا میزان شباهت و تفاوت بین مدل‌های شبکه عصبی دستگاه‌های نهایی را ارزیابی کنیم.

بعد از بررسی و تعیین میزان شباهت مدل‌ها، باید تصمیم گرفت که کدام یک از آن‌ها را با یکدیگر جابجا کنیم. بهترین انتخاب برای جابجایی، مدلی است که کمترین شباهت را با مدل شبکه عصبی دستگاه فعلی دارد. دلیل این انتخاب این است که اگر مدل دستگاه فعلی با مدل دستگاه مقصد شباهت زیادی داشته باشد، جابجایی آن‌ها مؤثر نخواهد بود. این شباهت بالا به این معناست که این دو دستگاه نهایی داده‌های مشابهی داشته و در طول زمان آموزش‌های مشابهی دیده‌اند، بنابراین جابجایی مدل‌ها تأثیر قابل‌توجهی بر بهبود یادگیری نخواهد داشت.

بنابراین، انتخاب مدل‌هایی با کمترین شباهت بین دستگاه‌های نهایی می‌تواند به بهبود فرآیند یادگیری کمک کند. این فرض بر این اساس است که دستگاه‌هایی با مدل‌های متفاوت احتمالاً داده‌هایی با ساختارهای متفاوت دارند. جابجایی مدل‌ها بین این دستگاه‌ها، مدل‌ها را با داده‌های جدیدی روبه‌رو می‌کند که می‌تواند به یادگیری بهتر و متنوع‌تر کمک کند. در نتیجه، مدل سراسری سریع‌تر به سمت مسیر بهینه همگرا می‌شود و دقت و کارایی آن افزایش می‌یابد.

این روش نه تنها تنوع داده‌ها را در فرآیند یادگیری افزایش می‌دهد، بلکه به کاهش انحراف وزن‌ها نیز کمک می‌کند. با داشتن دید گسترده‌تری از داده‌ها و تجربیات مختلف، مدل‌ها می‌توانند بهتر و جامع‌تر آموزش ببینند. این امر در نهایت منجر به بهبود عملکرد کلی مدل در شرایط واقعی می‌شود و کمک می‌کند که مدل‌های یادگیری فدرال بتوانند با چالش‌های داده‌های non-IID به نحو بهتری مقابله کنند.

بار محاسباتی جهت بررسی مشابهت

در این روش، ابتدا تمام مدل‌های شبکه عصبی از دستگاه‌های نهایی به سمت سرور ارسال می‌شوند. سپس سرور بر اساس معیارهای مشخصی، شباهت بین این مدل‌ها را بررسی و اقدام به جابجایی آن‌ها می‌کند. در این فرآیند، تمامی محاسبات پردازشی در سرور انجام می‌شود.

افزایش بار کاری در سرور از نظر محاسباتی مشکلی ایجاد نمی‌کند زیرا سرور به‌طور خاص برای انجام چنین عملیات‌هایی طراحی شده است. در یادگیری فدرال، فرض بر این است که دستگاه‌های نهایی دارای سخت‌افزار ضعیف و منابع محدود هستند، بنابراین بار محاسباتی سنگین بر عهده آن‌ها گذاشته نمی‌شود. در این روش نیز تمامی پردازش‌های سنگین بر روی سرور انجام می‌شود که از قدرت پردازشی بالایی برخوردار است و می‌تواند به راحتی این عملیات را مدیریت کند.

سرورهای مرکزی معمولاً دارای منابع پردازشی قوی، حافظه زیاد و قابلیت‌های پیشرفته برای انجام محاسبات پیچیده هستند. این ویژگی‌ها به سرور اجازه می‌دهد تا بدون هیچ مشکلی عملیات‌هایی مانند تعویض مدل‌ها و تجمیع نتایج را انجام دهد. این امر به خصوص در مورد یادگیری فدرال بسیار حیاتی است زیرا بار محاسباتی سنگین نباید بر دستگاه‌های نهایی با سخت‌افزار ضعیف تحمیل شود.

با انجام عملیات بر روی سرور، دستگاه‌های نهایی تنها به تبادل داده‌های لازم و اجرای به‌روزرسانی‌های محلی سبک می‌پردازند. این رویکرد باعث خواهد شد که فرآیند یادگیری بهینه‌تری ایجاد شود و مدل‌ها به طور موثر و کارآمدتری آموزش ببینند. در نتیجه، مشکلات محاسباتی به حداقل می‌رسد و عملکرد کلی سیستم بهبود خواهد یافت.

این روش نه تنها به حفظ منابع محدود دستگاه‌های نهایی کمک می‌کند، بلکه بهره‌وری بالاتری نیز از قدرت

پردازشی سرور به دست می‌آید. به این ترتیب، می‌توان اطمینان داشت که عملیات‌های پیچیده و محاسبات سنگین به درستی و با سرعت مناسب انجام می‌شوند، بدون اینکه فشار اضافی بر دستگاه‌های نهایی وارد شود. به این ترتیب، این روش می‌تواند به طور موثری در محیط‌های مختلف با دستگاه‌های متنوع و منابع محدود پیاده‌سازی شود و نتایج قابل اعتمادی ارائه دهد.

۴-۴ تعریف مسئله در معیارهای مشابهت

فرض کنید X ماتریسی با ابعاد $n \times p_1$ باشد که $X \in \mathbb{R}^{n \times p_1}$ شامل n نمونه و p_1 ویژگی است. همچنین، Y ماتریسی با ابعاد $n \times p_2$ باشد که $Y \in \mathbb{R}^{n \times p_2}$ شامل n نمونه و p_2 ویژگی است. فرض می‌شود که p_1 کمتر یا مساوی p_2 است.

هدف طراحی و تحلیل یک شاخص شباهت عددی $s(X, Y)$ است که بتواند بازنمایی‌های[1] موجود در ماتریس‌های X و Y را هم درون یک شبکه عصبی و هم بین شبکه‌های عصبی مختلف مقایسه کند. چنین شاخصی به ما کمک می‌کند تا تاثیر عوامل مختلف در یادگیری عمیق را بهتر درک کرده و این تاثیرات را به تصویر بکشیم.

به عنوان مثال، در بررسی شبکه‌های عصبی، ماتریس X می‌تواند نمایانگر فعال‌سازهای[2] نورون‌ها در یک لایه خاص برای n نمونه ورودی باشد و ماتریس Y می‌تواند نمایانگر فعال‌سازهای نورون‌ها در لایه‌ای دیگر یا حتی در یک شبکه عصبی دیگر برای همان n نمونه باشد. مقایسه این دو ماتریس اطلاعات مهمی درباره نحوه یادگیری و بازنمایی داده‌ها توسط شبکه عصبی ارائه می‌دهد.

شاخص $s(X, Y)$ باید توانایی اندازه‌گیری شباهت‌ها و تفاوت‌های بین بازنمایی‌های مختلف را داشته باشد. این شاخص می‌تواند به پژوهشگران کمک کند تا نحوه تغییر بازنمایی‌ها در اثر عوامل مختلف مانند تغییرات در داده‌های ورودی، تغییرات در معماری شبکه یا تغییرات در پارامترهای آموزش را بهتر درک کنند.

طراحی و تحلیل این شاخص شباهت می‌تواند به بهبود فهم ما از نحوه عملکرد شبکه‌های عصبی کمک کند و ابزار مفیدی برای بهبود روش‌های آموزش و بهینه‌سازی شبکه‌های عصبی فراهم کند.

۴-۵ پایداری در معیارهای مشابهت

این بخش ویژگی‌های لازم برای یک معیار مقایسه بازنمایی‌های شبکه عصبی را بررسی می‌کند. این بررسی شامل تحلیل پایداری شاخص‌های شباهت و تأثیرات آن‌ها در اندازه‌گیری شباهت بازنمایی‌های شبکه عصبی است.

[1] Representations
[2] Activations

اهمیت این موضوع در این است که معیار شباهت مورد استفاده باید نسبت به تبدیل متعامد[1] و مقیاس‌بندی همسان‌گرد[2] پایدار باشد. این ویژگی‌ها به معیار شباهت امکان می‌دهند تا بازنمایی‌های شبکه عصبی را به درستی مقایسه کرده و تأثیرات مختلف در فرآیند آموزش شبکه عصبی را بهتر درک کند.

۴-۵-۱ تبدیل متعامد

پایداری نسبت به تبدیل‌های متعامد به این معناست که اگر $s(X,Y)$ یک شاخص شباهت بین دو ماتریس X و Y باشد، این شاخص باید در مقابل تغییرات متعامد نیز پایدار باقی بماند. به عبارت دیگر، اگر U و V ماتریس‌های متعامد با رتبه کامل[3] باشند که $U^TU=I$ و $V^TV=I$ را برآورده کنند، باید $s(X,Y)=s(XU,YV)$ باشد. این ویژگی تضمین می‌کند که حتی در صورتی که ابعاد p بزرگتر از n باشند، شاخص شباهت همچنان به‌طور مطلوب عمل می‌کند. علاوه بر این، تبدیل‌های متعامد خواص مهمی از جمله حفظ حاصل‌ضرب‌های عددی و فاصله‌های اقلیدسی[4] بین نمونه‌ها را نیز حفظ می‌کنند. این امر باعث می‌شود که مقایسه‌های انجام شده توسط این شاخص‌ها دقیق و قابل اعتماد باشند [۳۷].

پایداری نسبت به تبدیل‌های متعامد برای شبکه‌های عصبی که با استفاده از روش نزول گرادیان آموزش داده می‌شوند، بسیار مطلوب است. این ویژگی نه تنها پایداری نسبت به تغییرات متعامد را تضمین می‌کند بلکه شامل پایداری نسبت به جایگشت نیز می‌شود. جایگشت در یک ماتریس به معنای این است که مقدارهای درون ماتریس فقط جابجا می‌شوند و ارزش‌های آن‌ها ثابت باقی می‌مانند. این پایداری برای تطبیق تقارن‌های شبکه‌های عصبی ضروری است [۳۸، ۳۹].

در حالت خطی، اگر ورودی‌ها با یک تبدیل متعامد تغییر کنند، روند آموزش با روش نزول گرادیان تحت تأثیر قرار نمی‌گیرد. برای شبکه‌های عصبی که با وزن‌های متقارن و تصادفی شروع می‌شوند، تبدیل‌های متعامد بر فعال‌سازها باعث می‌شود که روند آموزشی مشابه حالت بدون تغییر باقی بماند. اما اگر یک تغییر خطی دلخواه انجام شود، این ویژگی حفظ نمی‌شود و ممکن است روند آموزش تحت تأثیر منفی قرار گیرد [۴۰].

به‌طور کلی، پایداری نسبت به تبدیل‌های متعامد در شبکه‌های عصبی اهمیت زیادی دارد زیرا این ویژگی کمک می‌کند تا شبکه‌های عصبی در مواجهه با تغییرات متقارن در داده‌ها، به درستی عمل کنند و دقت و کارایی آن‌ها در فرآیند آموزش بهینه باقی بماند.

---
[1] Orthogonal Transformation
[2] Isotropic Scaling
[3] Full Rank
[4] Euclidean Distances

۴-۵-۲ مقیاس‌بندی همسان‌گرد

انتظار می‌رود که شاخص‌های شباهت زمانی که ورودی‌ها به صورت همسان مقیاس‌بندی می‌شوند، پایدار بمانند. به این معنا که اگر ورودی‌ها با ضرایب مثبت α و β ضرب شوند، شاخص شباهت نباید تغییری کند. به عبارتی دیگر، $s(X,Y) = s(\alpha X, \beta Y)$ باید برای هر α و β مثبت صحیح باشد.

این ویژگی اهمیت خاصی دارد زیرا تضمین می‌کند که مقایسه بازنمایی‌های شبکه‌های عصبی تحت تأثیر مقیاس‌بندی یکسان قرار نمی‌گیرد و دقت شاخص حفظ می‌شود. در واقع، این شاخص‌ها قادرند در شرایطی که شبکه‌های عصبی تحت تغییرات یکسان مقیاس قرار می‌گیرند، همچنان به درستی بازنمایی‌های مختلف را مقایسه کنند.

از سوی دیگر، شاخص‌هایی که در برابر مقیاس‌بندی همسان‌گرد پایدار هستند، اما نسبت به مقیاس‌بندی غیرهمسان‌گرد (یعنی تغییر مقیاس ویژگی‌های فردی) مقاوم نیستند، اهمیت ویژه‌ای دارند. این شاخص‌ها می‌توانند به دقت بیشتری در مقایسه بازنمایی‌های شبکه‌های عصبی کمک کنند و به درک بهتر تأثیرات مختلف در فرآیند یادگیری عمیق یاری رسانند [۳۷].

به عنوان مثال، وقتی شبکه‌های عصبی در معرض تغییرات یکسان مقیاس قرار می‌گیرند، این شاخص‌ها همچنان قادر خواهند بود بازنمایی‌های مختلف را به درستی مقایسه کنند. این ویژگی امکان فهم بهتر تأثیرات گوناگون در طول آموزش شبکه‌های عصبی و استفاده از این شاخص‌ها برای تحلیل و بهبود عملکرد مدل‌ها را فراهم می‌کند. بنابراین، شاخص‌های مقاوم در برابر مقیاس‌بندی همسان‌گرد می‌توانند ابزار مفیدی برای ارزیابی و بهینه‌سازی شبکه‌های عصبی باشند.

۴-۶ مقایسه ساختارهای مشابهت

در تحلیل بازنمایی‌های شبکه‌های عصبی، یکی از چالش‌های اصلی مقایسه ویژگی‌های چندگانه هر نمونه در بازنمایی‌های مختلف است. این روش ممکن است پیچیده و زمان‌بر باشد و نتایج گمراه‌کننده‌ای ایجاد کند. برای حل این مشکل، می‌توان از رویکردی استفاده کرد که به جای مقایسه مستقیم ویژگی‌های هر نمونه، ساختارهای شباهتی بین نمونه‌ها را بررسی کند.

ایده اصلی این است که به جای مقایسه مستقیم ویژگی‌های چندگانه هر نمونه در دو بازنمایی مختلف، می‌توان ابتدا شباهت بین هر جفت نمونه در هر بازنمایی را به صورت جداگانه سنجید و سپس این ساختارهای شباهتی را با هم مقایسه کرد [۳۷].

برای روشن‌تر شدن این موضوع، فرض کنید به جای اینکه به صورت مستقیم ویژگی‌های چند بعدی دو نمونه

را با هم مقایسه کنیم، ابتدا بررسی می‌کنیم که هر کدام از این نمونه‌ها چقدر به سایر نمونه‌ها شباهت دارند. سپس این ماتریس‌های شباهت بازنمایی را که هر کدام نشان‌دهنده میزان شباهت یک نمونه به سایر نمونه‌ها است، با هم مقایسه می‌کنیم.

نکته مهم این است که اگر برای اندازه‌گیری شباهت از ضرب داخلی استفاده شود، شباهت بین ماتریس‌های بازنمایی به یک مفهوم دیگر و قابل درک از شباهت بین ویژگی‌های جفتی تبدیل می‌شود. به عبارت دیگر، این روش امکان دست‌یابی به درک دقیق‌تری از شباهت بین ویژگی‌ها را بدون مقایسه مستقیم ویژگی‌های چندگانه هر نمونه فراهم می‌کند. این رویکرد می‌تواند به طور قابل توجهی در تحلیل و درک بازنمایی‌های شبکه‌های عصبی و داده‌های پیچیده مؤثر باشد، زیرا ساختارهای پیچیده را به شیوه‌ای ساده‌تر و قابل فهم‌تر بررسی می‌کند.

۴-۶-۱ انتخاب هسته[1]

در معیارهای مشابهت و اندازه‌گیری وابستگی، مفهوم هسته یا kernel نقش بسیار مهمی دارد. هسته در واقع یک تابع ریاضی است که برای محاسبه شباهت بین داده‌های ورودی استفاده می‌شود. این تابع، داده‌ها را به یک فضای ویژگی بالاتر نگاشت می‌کند تا بتوان همبستگی‌ها و مشابهت‌های پیچیده‌تر بین آن‌ها را بهتر سنجید.

به بیان ساده‌تر، تابع هسته k یک تابع مثبت معین است که دو بردار ورودی x_i و x_j را گرفته و یک عدد حقیقی تولید می‌کند که نشان‌دهنده میزان شباهت بین این دو بردار است. هسته‌ها می‌توانند به شکل‌های مختلفی باشند که هر کدام ویژگی‌ها و کاربردهای خاص خود را دارند. در اینجا به چند نمونه رایج اشاره می‌شود.

- هسته خطی[2]: این هسته به سادگی ضرب داخلی[3] دو بردار ورودی را محاسبه می‌کند.

$$k(x_i, x_j) = x_i^T x_j \qquad (۱-۴)$$

- هسته چندجمله‌ای[4]: این هسته ضرب داخلی را با یک توان مثبت، بالا می‌برد.

$$k(x_i, x_j) = (x_i^T x_j + c)^d \qquad (۲-۴)$$

که در آن c یک ثابت و d درجه چندجمله‌ای است.

- هسته گاوسی (RBF)[5]: این هسته فاصله اقلیدسی بین دو بردار را در یک تابع نمایی قرار می‌دهد، که باعث

[1] Kernel
[2] Linear Kernel
[3] Inner Product
[4] Polynomial Kernel
[5] Radial Basis Function Kernel

می‌شود داده‌هایی که نزدیک به هم هستند شباهت بیشتری داشته باشند.

$$k(x_i, x_j) = \exp\left(-\frac{||x_i - x_j||^2}{2\sigma^2}\right) \quad (۳-۴)$$

که در آن σ پارامتر پهنای باند است و تنظیم‌کننده میزان تاثیر فاصله می‌باشد.

برای هسته RBF، چندین استراتژی مختلف برای انتخاب پهنای باند σ وجود دارد که میزان تأکید بر شباهت فواصل کوچک نسبت به فواصل بزرگ را کنترل می‌کند. σ به عنوان کسری از فاصله میانه بین نمونه‌ها تنظیم می‌شود. در عمل، مشاهده می‌شود که هسته‌های RBF و خطی در بیشتر آزمایش‌ها نتایج مشابهی ارائه می‌دهند [۳۷].

استفاده از توابع هسته‌ای در معیارهای مشابهت و وابستگی، این امکان را فراهم می‌کند که داده‌های ورودی به یک فضای ویژگی بالاتر نگاشت شوند، جایی که روابط پیچیده و غیرخطی بین داده‌ها می‌توانند به صورت ساده‌تری مدل‌سازی شوند. به این ترتیب، معیارها با بهره‌گیری از هسته‌ها می‌توانند تحلیل دقیق‌تر و کارآمدتری از داده‌های پیچیده ارائه دهند. این ویژگی باعث می‌شود که هسته‌ها ابزار قدرتمندی در تحلیل داده‌ها باشند.

۴-۷ معیارهای سنجش مشابهت

در این بخش به بررسی روش‌های مختلفی که برای سنجش مشابهت بین بازنمایی‌های شبکه‌های عصبی استفاده می‌شود، پرداخته خواهد شد. یکی از روش‌های مهم در این زمینه استفاده از پایه‌های متعامد است. به طور خاص، فرض می‌کنیم که Q_X و Q_Y پایه‌های متعامدی برای ستون‌های ماتریس‌های X و Y هستند. به این معنا که Q_X و Q_Y به صورت $Q_X = X(X^TX)^{-1/2}$ و $Q_Y = Y(Y^TY)^{-1/2}$ تعریف شده‌اند. این پایه‌های متعامد به ما اجازه می‌دهند تا بازنمایی‌های شبکه عصبی را به صورت موثرتری تحلیل کنیم و شباهت‌های آن‌ها را با دقت بیشتری بسنجیم.

استفاده از پایه‌های متعامد به این معنی است که ما می‌توانیم بدون نگرانی از وابستگی‌های خطی بین ستون‌ها، شباهت‌ها را به صورت مستقیم مقایسه کنیم. این رویکرد به ویژه زمانی مفید است که می‌خواهیم درک کنیم چگونه شبکه عصبی ویژگی‌های مختلف داده‌ها را استخراج و بازنمایی می‌کند. با این روش می‌توانیم تحلیل‌های دقیقی انجام دهیم و بفهمیم که چگونه تغییرات در داده‌های ورودی یا ساختار شبکه، بازنمایی‌های داخلی شبکه را تحت تاثیر قرار می‌دهند. این نوع تحلیل‌ها می‌تواند به بهبود و بهینه‌سازی شبکه‌های عصبی و الگوریتم‌های یادگیری عمیق کمک شایانی کند.

۴-۷-۱ ضرب داخلی

یک رابطه ساده وجود دارد که ضرب داخلی بین نمونه‌ها را با ضرب داخلی بین ویژگی‌ها مرتبط می‌سازد:

$$\langle \text{vec}(XX^T), \text{vec}(YY^T) \rangle = \text{tr}(XX^TYY^T) = ||Y^TX||_F^2 \qquad (۴-۴)$$

که در آن عناصر XX^T و YY^T نشان‌دهنده ضرب داخلی بین بازنمایی نمونه‌های i و j هستند و شباهت بین این نمونه‌ها را بر اساس شبکه‌های مربوطه نشان می‌دهند.

به بیان دیگر، بخش چپ رابطه (۴-۴)، میزان مشابهت بین الگوهای شباهت، میان نمونه‌ها را ارزیابی می‌کند. این در حالی است که سمت راست، با جمع کردن مربعات ضرب‌های داخلی بین هر جفت از نمونه‌ها، به همان نتیجه مشابه می‌رسد و شباهت بین ویژگی‌های X و Y را اندازه‌گیری می‌کند.

این رابطه نشان می‌دهد که می‌توان به جای مقایسه مستقیم ویژگی‌ها، از شباهت‌های بین نمونه‌ها استفاده کرد تا به فهم بهتری از بازنمایی‌های شبکه‌های عصبی و داده‌های پیچیده دست یافت. به این ترتیب، تحلیل و درک داده‌ها ساده‌تر و مؤثرتر خواهد بود، زیرا این روش به ما اجازه می‌دهد تا به صورت غیرمستقیم و با استفاده از شباهت‌های موجود بین نمونه‌ها، به نتایج دقیق‌تری برسیم.

۴-۷-۲ تحلیل همبستگی کانونی (CCA)[1]

برای درک بهتر این معیار، از یک مثال ساده استفاده می‌شود. فرض کنید دو مجموعه داده مختلف در اختیار است، مجموعه‌ای شامل اطلاعاتی همچون قد و وزن افراد و مجموعه دیگر حاوی اطلاعاتی مانند سن و درآمد آن‌ها باشد. هدف تحلیل همبستگی کانونی، این است که ارتباط‌های پنهان بین این دو مجموعه داده را کشف کند. به بیان ساده، CCA به دنبال شناسایی ترکیب‌هایی از ویژگی‌ها در هر مجموعه داده است که با مقایسه آن‌ها، بیشترین همبستگی حاصل شود. تلاش بر این است که مشخص شود کدام ترکیب قد و وزن با کدام ترکیب سن و درآمد بیشترین ارتباط را دارد.

ابتدا داده‌ها استاندارد می‌شوند، به این صورت که میانگین هر ویژگی صفر شده و داده‌ها به گونه‌ای تغییر می‌کنند که انحراف معیارشان یک شود. سپس، CCA بردارهای وزنی را برای هر مجموعه داده محاسبه می‌کند تا ترکیب‌های خطی از این داده‌ها ایجاد کند. این ترکیب‌ها به گونه‌ای انتخاب می‌شوند که حداکثر همبستگی بین آن‌ها وجود داشته باشد. به عنوان مثال، CCA می‌خواهد ترکیبی از قد و وزن (مثلاً $0.5 \times \text{قد} + 0.5 \times \text{وزن}$) و ترکیبی از سن و درآمد (مثلاً $0.3 \times \text{سن} + 0.7 \times \text{درآمد}$) را بیابد که بیشترین ارتباط را با هم داشته باشند.

با استفاده از این بردارهای وزنی، ترکیب‌های جدیدی از داده‌ها ایجاد می‌شوند و سپس همبستگی بین این

[1] Canonical Correlation Analysis

ترکیب‌ها محاسبه می‌شود. CCA به دنبال یافتن پایه‌هایی برای دو ماتریس است به طوری که وقتی ماتریس‌های اصلی بر روی این پایه‌ها توزیع می‌شوند، همبستگی به حداکثر برسد. برای هر i (که بین ۱ و $p1$ قرار دارد)، ضریب همبستگی کانونی ρ_i به صورت زیر تعریف می‌شود:

$$\rho_i = \max_{\mathbf{w}_X^i, \mathbf{w}_Y^i} \text{corr}(X\mathbf{w}_X^i, Y\mathbf{w}_Y^i) \quad (۵-۴)$$

با در نظر گرفتن بردارهای $\mathbf{w}_X^i \in \mathbb{R}^{p1}$ و $\mathbf{w}_Y^i \in \mathbb{R}^{p2}$، ضریب همبستگی کانونی ρ_i هدفش این است که همبستگی بین ترکیب خطی $X\mathbf{w}_X^i$ و $Y\mathbf{w}_Y^i$ را به حداکثر برساند.

برای اطمینان از اینکه ترکیب‌های جدید داده‌ها مستقل و متفاوت از هم باشند، شرط‌های زیر باید رعایت شوند:

$$\begin{aligned} &\forall_{j<i} \quad X\mathbf{w}_X^i \perp X\mathbf{w}_X^j \\ &\forall_{j<i} \quad Y\mathbf{w}_Y^i \perp Y\mathbf{w}_Y^j \end{aligned} \quad (۶-۴)$$

این شرط‌ها اطمینان می‌دهند که ترکیب‌های جدید از داده‌ها با ترکیب‌های قبلی همپوشانی نداشته و متعامد باقی بمانند.

در نهایت برای مقایسه دو شبکه عصبی و اندازه‌گیری شباهت بین آن‌ها، از معیاری به نام R^2_{CCA} استفاده می‌شود. این معیار نشان می‌دهد که چقدر از اطلاعات داده‌ها، توسط ترکیب‌های خطی حاصل از روش CCA توضیح داده می‌شود. رابطه این معیار به این صورت است:

$$R^2_{CCA} = \frac{\sum_{i=1}^{p1} \rho_i^2}{p1} = \frac{||Q_Y^T Q_X||_F^2}{p1} \quad (۷-۴)$$

که با محاسبه و جمع کردن مربعات ضرایب همبستگی کانونی و سپس تقسیم آن‌ها بر تعداد ضرایب، میزان شباهت بین دو شبکه عصبی را ارزیابی می‌کند.

با استفاده از روش CCA، می‌توان ترکیب‌های خطی از ویژگی‌های دو مجموعه داده مختلف را شناسایی کرد که بالاترین همبستگی را با هم دارند. این روش کمک می‌کند تا روابط پنهان و مهم بین هر دو مجموعه داده آشکار شود و تحلیل‌های دقیق‌تری صورت گیرد.

۴-۷-۳ معیار استقلال هیلبرت‌ـاشمیت (HSIC)[1]

برای بررسی میزان وابستگی و شباهت بین دو مجموعه داده، می‌توان از معیار استقلال هیلبرت‌ـاشمیت استفاده کرد. این معیار به طور خاص برای اندازه‌گیری همبستگی بین داده‌های مختلف طراحی شده است. به طور

[1] Hilbert-Schmidt Independence Criterion

دقیق‌تر، برای داده‌های مرکزیت‌یافته (میانگین صفر در هر ستون) X و Y، رابطه زیر برقرار است:

$$\frac{1}{(n-1)^2}\text{tr}(XX^TYY^T) = ||\text{cov}(X^T, Y^T)||_F^2 \quad (۸-۴)$$

معیار HSIC این معادله را با استفاده از فضایی خاص تعمیم می‌دهد و امکان بررسی مؤثر وابستگی‌ها را فراهم می‌کند. به بیان دیگر، این معیار با بهره‌گیری از توابع هسته‌ای، همبستگی بین ماتریس‌های داده را اندازه‌گیری می‌کند. به این نحو که عناصر K_{ij} و L_{ij} به ترتیب از طریق $k(x_i, x_j)$ و $l(y_i, y_j)$ محاسبه می‌شوند، که در اینجا k و l توابع هسته‌ای هستند [۴۱].

برآورد HSIC به صورت زیر تعریف می‌شود:

$$\text{HSIC}(K, L) = \frac{1}{(n-1)^2}\text{tr}(KHLH) \quad (۹-۴)$$

که در آن H ماتریس مرکزیت‌دهنده است و به شکل $H_n = I_n - \frac{1}{n}11^T$ تعریف می‌شود.

نکته جالب اینجاست که اگر هسته‌های خطی k و l به صورت $k(x, y) = l(x, y) = x^Ty$ باشند، HSIC به همان معادله اولیه برمی‌گردد. این بدین معناست که معیار HSIC می‌تواند به ما کمک کند تا به شکلی دقیق و قابل اعتماد، میزان وابستگی و شباهت بین داده‌ها را ارزیابی کنیم و درک بهتری از ساختارهای پیچیده داده‌ها به دست آوریم.

۴-۷-۴ هم‌ترازی هسته مرکزی (CKA)[1]

معیار HSIC در اندازه‌گیری همبستگی‌ها با مشکل عدم پایداری نسبت به مقیاس‌بندی همسان‌گرد ویژگی‌ها مواجه است. این بدان معناست که در صورت تغییر مقیاس ویژگی‌ها، نتیجه HSIC ممکن است دستخوش تغییراتی شود که به درستی شباهت‌های بین داده‌ها را نشان ندهد. برای رفع این مورد، از فرم نرمال‌شده‌ای به نام هم‌ترازی هسته مرکزی استفاده می‌شود.

CKA یک شاخص نرمال‌شده است که تأثیر مقیاس‌بندی همسان‌گرد را حذف می‌کند و به این ترتیب، دقت و پایداری بیشتری در مقایسه بازنمایی‌های شبکه‌های عصبی فراهم می‌آورد [۴۲، ۴۳]. رابطه CKA به شکل زیر تعریف می‌شود:

$$\text{CKA}(K, L) = \frac{\text{HSIC}(K, L)}{\sqrt{\text{HSIC}(K, K) \cdot \text{HSIC}(L, L)}} \quad (۱۰-۴)$$

که در آن عبارت $\text{HSIC}(K, L)$ میزان همبستگی بین دو ماتریس هسته‌ای K و L را اندازه‌گیری می‌کند. صورت کسر، همان HSIC اصلی است که میزان همبستگی بین دو ماتریس داده را نشان می‌دهد. اما برای نرمال‌سازی

[1] Centered Kernel Alignment

این مقدار و حذف تأثیر مقیاس‌بندی، مخرج کسر به کار می‌رود که شامل ضرب دو HSIC مربوط به هر یک از ماتریس‌ها با خودشان است. این نرمال‌سازی باعث می‌شود که نتیجه نهایی مستقل از مقیاس‌بندی ویژگی‌ها باشد و شباهت‌های واقعی بین داده‌ها را بهتر منعکس کند.

استفاده از CKA در مقایسه با HSIC، به‌ویژه در تحلیل بازنمایی‌های شبکه‌های عصبی و داده‌های پیچیده، کارایی بهتری دارد. زیرا این شاخص نرمال‌شده، نه تنها شباهت‌های بین داده‌ها را دقیق‌تر می‌سنجد، بلکه در برابر تغییرات مقیاس‌بندی نیز مقاوم است. به این ترتیب، می‌توان از CKA به عنوان یک ابزار قدرتمند برای درک و تحلیل بازنمایی‌های مختلف در یادگیری عمیق و دیگر زمینه‌های مرتبط استفاده کرد.

۴-۷-۵ هم‌ترازی هسته مرکزی بدون مداخله (dCKA)[1]

فرض می‌شود X یک مجموعه داده ورودی با n نمونه و p ویژگی باشد. نمایش لایه‌های m_1 و m_2 از دو شبکه عصبی که به ترتیب f_1 و f_2 نامیده می‌شوند، به صورت $X_{f_1}^{m_1}$ و $X_{f_2}^{m_2}$ هستند. شبکه‌های عصبی f_1 و f_2 دو مدل متفاوت یادگیری ماشین با معماری و ساختار مخصوص به خود هستند. لایه‌های m_1 و m_2 نیز لایه‌های خاصی در این شبکه‌ها هستند که مورد بررسی قرار می‌گیرند. برای مثال، m_1 می‌تواند لایه دوم از شبکه f_1 و m_2 می‌تواند لایه چهارم از شبکه f_2 باشد.

برای مقایسه نمایش‌های دو شبکه عصبی، ساختارهای شباهت در هر شبکه بررسی می‌شود. این کار با محاسبه شباهت بین هر جفت از نمونه‌ها در $X_{f_1}^{m_1}$ و $X_{f_2}^{m_2}$ با استفاده از یک معیار شباهت $k(\cdot,\cdot)$ انجام می‌شود:

$$K_{f_1}^{m_1} = k(X_{f_1}^{m_1}, X_{f_1}^{m_1}), \quad K_{f_2}^{m_2} = k(X_{f_2}^{m_2}, X_{f_2}^{m_2}) \tag{۱۱-۴}$$

در این رابطه ماتریس‌های $K_{f_1}^{m_1}$ و $K_{f_2}^{m_2}$ نشان‌دهنده شباهت بین هر جفت از نمونه‌ها در لایه‌های m_1 و m_2 از شبکه‌های f_1 و f_2 هستند. در مرحله بعد، برای مقایسه این دو ماتریس از معیاری به نام $s(\cdot,\cdot)$ استفاده می‌شود:

$$s_{f_1,f_2}^{m_1,m_2} = s(K_{f_1}^{m_1}, K_{f_2}^{m_2}) \tag{۱۲-۴}$$

این مقدار بیانگر میزان شباهت بین دو نمایش شبکه‌های عصبی است [۴۴]. این روش امکان درک دقیقی از شباهت‌ها و تفاوت‌های بین دو شبکه عصبی را فراهم می‌کند و می‌توان از آن برای بهبود عملکرد مدل‌های یادگیری ماشین بهره برد.

روش‌های فعلی برای مقایسه نمایش‌های شبکه‌های عصبی از معیارهای شباهت متفاوتی در دو مرحله استفاده می‌کنند. برای مثال روش CKA در مرحله اول از یک تابع هسته‌ای برای اندازه‌گیری شباهت استفاده می‌کند که

[1] Deconfounded Centered Kernel Alignment

با $k(\cdot,\cdot)$ نشان داده می‌شود. در مرحله دوم، برای اندازه‌گیری شباهت بین تابع‌های هسته‌ای از معیاری به نام HSIC که با $s(\cdot,\cdot)$ نشان داده می‌شود، استفاده می‌کند.

۴-۷-۵-۱ تأثیر متغیرهای مداخله‌گر[1] در بررسی شباهت

فرض کنید مجموعه داده‌ای به نام X دارید که شباهت بین نمونه‌ها در آن با ماتریسی به نام K^0 نمایش داده می‌شود. این ماتریس شباهت، شباهت‌های اولیه بین داده‌ها در فضای ورودی را نشان می‌دهد. حالا وقتی پیش‌بینی با استفاده از شبکه‌های عصبی انجام می‌شود، می‌توان این پیش‌بینی‌ها را به‌صورت مرحله به مرحله در نظر گرفت.

در روش CKA، شباهت بین دو ماتریس $K_{f_1}^{m_1}$ و $K_{f_2}^{m_2}$، میزان شباهت بین دو شبکه عصبی را نشان می‌دهد. اما هر دو ماتریس $K_{f_1}^{m_1}$ و $K_{f_2}^{m_2}$ تحت تأثیر ماتریس شباهت اولیه K^0 قرار می‌گیرند. این مسئله ممکن است باعث شود که شباهت بین شبکه‌ها بیش از حد بالا برود و واقعی نباشد. به‌طور کلی، داده‌های مشابه در فضای ورودی احتمالاً در لایه‌های ابتدایی شبکه‌های عصبی نیز مشابه خواهند بود، حتی اگر عملکرد شبکه‌های عصبی متفاوت باشد. بنابراین، CKA ممکن است تحت تأثیر ویژگی‌های خاص مجموعه داده قرار بگیرد و نتایج ناسازگاری بین مجموعه داده‌های مختلف ایجاد کند.

برای حل این مشکل، شباهت ورودی K^0 از ماتریس‌های $K_{f_1}^{m}$ و $K_{f_2}^{m}$ حذف می‌شود. این فرآیند، حذف مداخله‌گر[2] نامیده می‌شود. با این روش، تنها عملکرد شبکه عصبی علت شباهت‌های موجود در فضای پنهان خواهد بود و این شباهت‌ها ناشی از شباهت اولیه داده‌ها نخواهد بود. به این ترتیب، معیار بدون مداخله به مقایسه عملکرد واقعی شبکه‌های عصبی می‌پردازد و کمتر تحت تأثیر ساختار اولیه مجموعه داده قرار می‌گیرد [۴۴].

۴-۷-۵-۲ حذف تأثیر متغیر مداخله‌گر

برای تنظیم شباهت‌های نادرست که به خاطر متغیرهای مداخله‌گر به وجود می‌آیند، از روشی استفاده می‌شود که در آن ساختار شباهت ورودی از ساختار شباهت بازنمایی حذف می‌شود. این روش به شکل زیر تعریف شده است:

$$dK_{f_1}^{m_1} = K_{f_1}^{m_1} - \hat{\alpha}_{f_1}^{m_1} K^0, \quad dK_{f_2}^{m_2} = K_{f_2}^{m_2} - \hat{\alpha}_{f_2}^{m_2} K^0 \qquad (۱۳-۴)$$

در اینجا، K^0 نمایانگر ساختار شباهت ورودی است و $\hat{\alpha}_{f_1}^{m_1}$ و $\hat{\alpha}_{f_2}^{m_2}$ ضرایبی هستند که برای حداقل کردن اندازه ماتریس‌های $dK_{f_1}^{m_1}$ و $dK_{f_2}^{m_2}$ در نظر گرفته می‌شوند. حرف d قبل از ماتریس شباهت، نشان می‌دهد که این ماتریس بدون تأثیر متغیر مداخله‌گر است [۴۴].

[1] Confounded
[2] Confounder

ساختار شباهت ورودی K^0 تأثیری خطی و قابل جمع شدن بر K_f^m دارد:

$$\text{vec}(K_f^m) = \alpha_f^m \text{vec}(K^0) + \epsilon_f^m \quad (۱۴-۴)$$

که در اینجا تابع $\text{vec}(\cdot)$ یک ماتریس را به یک بردار تبدیل می‌کند. نویز ϵ_f^m نیز مستقل از متغیر مداخله‌گر فرض شده است و معادلات زیر را شامل می‌شود:

$$\hat{\epsilon}_f^m = \text{vec}(dK_f^m) \quad (۱۵-۴)$$

$$\hat{\alpha}_f^m = (\text{vec}(K^0)^T \text{vec}(K^0))^{-1} \text{vec}(K^0)^T \text{vec}(K_f^m) \quad (۱۶-۴)$$

در اینجا برای حذف تأثیر یک ماتریس از روی دیگری، از معکوس آن ماتریس استفاده می‌شود. معکوس ماتریس A، ماتریسی است که وقتی در A ضرب شود، نتیجه یک ماتریس همانی[1] است. در اینجا، معکوس ماتریس $(\text{vec}(K^0)^T \text{vec}(K^0))$ به ما کمک می‌کند تا تأثیرات اولیه را حذف کنیم. در نهایت، ضرب داخلی $\text{vec}(K^0)$ با $\text{vec}(K_f^m)$ انجام می‌شود تا شباهت‌های واقعی بین داده‌ها را به دست آوریم. سپس، معکوس ماتریس مرحله قبلی در این عبارت ضرب می‌شود تا تأثیرات مزاحم حذف و شباهت‌های خالص نمایش داده شوند.

پس از به دست آوردن ساختارهای شباهت، بدون تأثیر متغیر مداخله‌گر، از همان معیار شباهت در رابطه (۱۲-۴) برای محاسبه استفاده می‌شود:

$$ds_{f_1,f_2}^{m_1,m_2} = s(dK_{f_1}^{m_1}, dK_{f_2}^{m_2}) \quad (۱۷-۴)$$

این روش به ما کمک می‌کند تا شباهت‌های واقعی و معتبر بین ساختارهای بازنمایی را بررسی کنیم، بدون اینکه متغیرهای مداخله‌گر نتایج را تحت تأثیر قرار دهند.

۸-۴ شاخص مشابهت بین شبکه‌های عصبی

برای بررسی و مقایسه کامل دو شبکه عصبی، ارزیابی جداگانه تمامی لایه‌های آن‌ها و ارائه یک شاخص شباهت کلی ضروری است. این کار با استفاده از معیارهای شباهت برای هر لایه انجام شده و سپس این معیارها به صورت میانگین ترکیب می‌شوند تا یک نمای کلی از شباهت بین دو شبکه عصبی به دست آید.

در لایه‌های کاملاً متصل[2] که تمامی نرون‌ها به هم متصل هستند، ساختار لایه‌ها به صورت ماتریس‌های دو بعدی است و بنابراین مقایسه آن‌ها با استفاده از معیارهای شباهت به راحتی امکان‌پذیر است. اما در لایه‌های

[1] Identity Matrix
[2] Fully connected

پیچشی[1] که دارای ساختار چند بعدی هستند، ابتدا باید ابعاد این لایه‌ها به دو بعد تبدیل شوند تا امکان مقایسه فراهم شود. این تبدیل معمولاً با تخت‌کردن[2] لایه‌ها انجام می‌گیرد. پس از تبدیل، از معیارهای مشابه لایه‌های کاملاً متصل استفاده می‌شود.

برای مثال، دو شبکه عصبی فرض می‌شود که هر کدام شامل چندین لایه مختلف هستند. ابتدا برای هر لایه از شبکه اول، لایه متناظر در شبکه دوم پیدا می‌شود. سپس با استفاده از معیارهای شباهت، میزان شباهت بین این دو لایه محاسبه می‌شود و این فرآیند برای تمامی لایه‌ها تکرار می‌شود.

پس از محاسبه معیارهای شباهت برای تمامی لایه‌ها، این معیارها به صورت میانگین ترکیب می‌شوند تا یک شاخص کلی از شباهت بین دو شبکه عصبی به دست آید. این شاخص میانگین می‌تواند نشان دهد که دو شبکه چقدر به یکدیگر شبیه هستند.

این روش به درک بهتر از عملکرد و ساختار داخلی شبکه‌های عصبی کمک می‌کند. با داشتن این شاخص شباهت، شناسایی تفاوت‌ها و شباهت‌های دو شبکه آسان‌تر شده و می‌توان بر اساس آن‌ها تصمیمات بهتری برای بهبود مدل‌های یادگیری گرفت.

۴-۹ نحوه تعیین کاربران نهایی جهت جابجایی مدل‌ها

زمانی که همه مدل‌های شبکه عصبی در سرور مرکزی قرار دارند، وظیفه سرور این است که تصمیم بگیرد کدام کاربران مدل‌های خود را با یکدیگر جابجا کنند. برای انجام این کار، ابتدا باید مدل‌ها با یکدیگر مقایسه شوند تا میزان شباهت بین آن‌ها مشخص شود. سپس، بر اساس این شباهت‌ها تعیین می‌شود که کدام کاربران مدل‌های شبکه عصبی خود را با یکدیگر مبادله کنند، یا به عبارت دیگر، سرور مشخص می‌کند کدام مدل به کدام کاربر ارسال شود.

نکته مهمی که باید مد نظر قرار داد این است که فرآیند بررسی شباهت بین مدل‌های شبکه عصبی ممکن است زمان‌بر باشد. بنابراین، برای تصمیم‌گیری سریع درباره جابجایی مدل‌ها، باید از روش‌های مؤثری استفاده شود. در ادامه، دو روش برای تعیین کاربران نهایی جهت جابجایی مدل‌ها معرفی می‌شود.

۴-۹-۱ روش تعویض حریصانه[3]

در روش حریصانه، از بین تمام کاربران موجود، یک کاربر به صورت تصادفی انتخاب می‌شود. سپس مدل شبکه عصبی این کاربر با مدل‌های تمامی کاربران دیگر مقایسه می‌شود تا میزان شباهت آن‌ها سنجیده شود. در

[1] Convolutional Layers
[2] Flattening
[3] Greedy Swapping

این مرحله، کاربری که مدل شبکه عصبی او کمترین شباهت را با مدل کاربر انتخاب شده دارد، به عنوان کاربر مقصد برای تعویض مدل انتخاب می‌شود. پس از این انتخاب، سرور مدل‌های این دو کاربر را با یکدیگر جابجا می‌کند و در نهایت این دو کاربر را از لیست انتخاب حذف خواهد کرد.

پس از انجام این تعویض، فرایند مشابهی برای کاربران باقی‌مانده تکرار می‌شود. ابتدا یک کاربر دیگر به صورت تصادفی انتخاب می‌شود و دقیقا همان روند بالا برای آن تکرار خواهد شد، با توجه به اینکه در این فرایند، دو کاربری که مدل‌های آن‌ها در مرحله قبل تعویض شده بودند، در مقایسه‌های بعدی شرکت نمی‌کنند. شبه کد کامل این روش در الگوریتم ۴-۲ ارائه شده است. علاوه بر این، نمادهای مختص به این الگوریتم در جدول ۴-۲ و همچنین تمامی نمادهای پایه در جدول ۲-۱ توضیح داده شده‌اند. هدف از این جداول، فراهم کردن درکی جامع از نحوه عملکرد و پیاده‌سازی الگوریتم می‌باشد.

۴-۹-۱-۱ تعداد اجرای تابع ModelSimilarity

در ادامه این مورد بررسی خواهد شد که تابع ModelSimilarity در الگوریتم ۴-۲ چند مرتبه اجرا می‌شود. برای ساده‌تر کردن موضوع و فهم بهتر آن، لیست LRS با n مقدار اولیه در نظر گرفته می‌شود (لیستی با n عضو) و همچنین مقدار NS برابر $n/2$ لحاظ خواهد شد. دلیل این انتخاب این است که با انجام $n/2$ تعویض، همه مدل‌ها یک بار جابجا می‌شوند. سپس حلقه اصلی به تعداد NS تکرار می‌شود. در هر تکرار از این حلقه یک عنصر تصادفی از LRS انتخاب و حذف خواهد شد. سپس برای هر عنصر باقی‌مانده در LRS، تابع ModelSimilarity فراخوانی می‌شود.

تعداد تکرارهای حلقه داخلی که در آن تابع ModelSimilarity فراخوانی می‌شود وابسته به تعداد عناصر باقی‌مانده در LRS است. به طور دقیق‌تر، در اولین تکرار حلقه اصلی، LRS شامل $n-1$ عنصر است و در دومین تکرار، LRS شامل $n-3$ عنصر خواهد بود؛ زیرا در هر تکرار از حلقه اصلی، یک عنصر به صورت تصادفی و یک عنصر دیگر با کمترین شباهت حذف می‌شوند.

به این ترتیب، تعداد کل فراخوانی‌های تابع ModelSimilarity برابر است با مجموع تعداد عناصر باقی‌مانده در هر تکرار از حلقه اصلی:

$$\sum_{i=0}^{NS-1} (n-1-2i) \qquad (۱۸-۴)$$

که این مجموع برای $NS = n/2$ به صورت زیر است:

$$\sum_{i=0}^{(n/2)-1} (n-1-2i) \qquad (۱۹-۴)$$

الگوریتم ۴-۲: تعویض حریصانه (Greedy Swapping)

```
Function GreedySwapping():
    LRS = copy of U_t;
    NS = (length of U_t // 2) * SP;
    for NS times do
        RandomIndex = random integer between 0 and length of LRS;
        SCB = LRS[RandomIndex];
        remove SCB from LRS;
        Initialize LstSimilarity as an empty list;
        for each RC ∈ LRS do
            Sim = ModelSimilarity(w^SCB, w^RC);
            Append Sim to LstSimilarity;
        end
        MinSimilarityIndex = index of the minimum value in LstSimilarity;
        SCD = LRS[MinSimilarityIndex];
        remove SCD from LRS;
        FedSwap(SCB, SCD);
    end
end
```

جدول ۴-۲: نمادهای مختص الگوریتم تعویض حریصانه

متغیر	توضیحات
$LRS\,(LstRemainSwap)$	لیست باقی‌مانده تعویض
$NS\,(NumSwaps)$	تعداد تعویض‌ها
$SP\,(SwapPercentage)$	ضریب کنترلی برای تعداد تعویض‌ها
$RandomIndex$	شاخص تصادفی
$SCB\,(SwapClientBase)$	کاربر مبدا تعویض
$LstSimilarity$	لیست مشابهت
$RC\,(RemainClient)$	کاربر باقی‌مانده
Sim	معیار مشابهت بین دو مدل شبکه عصبی
$SCD\,(SwapClientDest)$	کاربر مقصد تعویض

این یک دنباله حسابی با مقدار اولیه $a = n - 1$ و قدر نسبت $d = -2$ است و تعداد جملات آن برابر با NS است. مجموع این دنباله حسابی به صورت زیر محاسبه می‌شود:

$$S = NS \times \left(\frac{a+l}{2}\right) \tag{۴-۲۰}$$

که در آن l مقدار آخرین جمله است و به این شکل به دست می‌آید:

$$\begin{aligned} l &= n - 1 - 2(NS - 1) \\ &= n - 1 - 2(n/2 - 1) \\ &= n - 1 - n + 2 \\ &= 1 \end{aligned} \tag{۴-۲۱}$$

بنابراین مجموع نهایی به این صورت خواهد بود:

$$\begin{aligned} S &= (n/2) \times \left(\frac{(n-1)+1}{2} \right) \\ &= (n/2) \times \left(\frac{n}{2} \right) \\ &= \frac{n^2}{4} \end{aligned} \qquad (۲۲-۴)$$

پس در نهایت تابع ModelSimilarity به تعداد $\left\lfloor \frac{n^2}{4} \right\rfloor$ بار اجرا می‌شود.

۴-۹-۱-۲ مرتبه زمانی

با توجه به فرض‌هایی که در بخش قبل مطرح شد، زمان اجرای الگوریتم تعویض حریصانه برابر با $O(n^2)$ است. این نتیجه‌گیری به این دلیل است که NS برابر با $n/2$ در نظر گرفته شده و هر بار اجرای حلقه اصلی شامل یک حذف از لیست LRS است که مرتبه زمانی آن $O(n)$ است. همچنین، حلقه داخلی نیز همان‌طور که در بخش ۴-۹-۱-۱ بررسی شد، از مرتبه $O(n)$ است. بنابراین، با ترکیب این دو عامل، مرتبه زمانی کل اجرای الگوریتم برابر با $O(n^2)$ خواهد بود.

۴-۹-۱-۳ حافظه مورد نیاز برای لیست $LstSimilarity$

در نهایت، میزان مصرف حافظه توسط لیست $LstSimilarity$ در الگوریتم ۴-۲ بررسی می‌شود. همان‌طور که در بخش قبل توضیح داده شد، حلقه داخلی این الگوریتم در بیشترین حالت به تعداد $n-1$ مرتبه اجرا می‌شود. بنابراین، از نظر حافظه، لیست $LstSimilarity$ در مرتبه $O(n)$ قرار دارد. با این حال، باید توجه داشت که برای یافتن کمترین مقدار در یک مجموعه، حافظه‌ای به میزان $O(1)$ نیز کافی است.

در این الگوریتم، استفاده از لیست $LstSimilarity$ به منظور افزایش خوانایی و سادگی کد صورت گرفته است، اگرچه از لحاظ بهینه‌سازی حافظه می‌توان از روش‌های کم‌حافظه‌تری نیز استفاده کرد. به بیان دیگر، به جای نگهداری همه شباهت‌ها در یک لیست و سپس پیدا کردن کمترین مقدار، می‌توان به صورت مستقیم در همان حلقه داخلی کمترین شباهت را دنبال کرد و در هر تکرار، فقط مقدار کمینه فعلی را به‌روزرسانی کرد. این روش نیاز به حافظه کمتری دارد و با حافظه $O(1)$ قابل انجام است.

با این حال، استفاده از لیست $LstSimilarity$ در اینجا به منظور ساده‌تر و قابل فهم‌تر کردن کد انجام شده است. این انتخاب به توسعه‌دهندگان امکان می‌دهد تا الگوریتم را بهتر درک کنند و روند مقایسه شباهت‌ها را به وضوح مشاهده کنند، با این شرط که بهینه‌سازی حافظه در اولویت نباشد.

۴-۹-۲ روش تعویض حداقل شباهت[1]

در این روش، برای اینکه حداقل شباهت ممکن بین تمامی مدل‌های شبکه عصبی به دست آید، لازم است تمامی مدل‌ها با یکدیگر مقایسه شوند و دو مدلی که کمترین شباهت را دارند با هم تعویض شوند. به عنوان مثال، اگر n مدل وجود داشته باشد، باید یک ماتریس $n \times n$ برای بررسی میزان شباهت‌ها ایجاد شود. در این ماتریس، شباهت مدل ۱ با مدل ۲ برابر با شباهت مدل ۲ با مدل ۱ در نظر گرفته می‌شود. همچنین به دلیل اینکه مقایسه یک مدل با خودش بی‌معنی است، ماتریس نهایی به شکل یک ماتریس مثلثی[2] البته بدون قطر اصلی تبدیل می‌شود. به این صورت که تنها حدود نیمی از ماتریس، شامل شباهت‌های مورد نیاز برای مقایسه است. در نتیجه ماتریس مورد نظر به شکل زیر در خواهد آمد.

$$\begin{bmatrix} \infty & a^{12} & a^{13} & \cdots & a^{1n} \\ \infty & \infty & a^{23} & \cdots & a^{2n} \\ \infty & \infty & \infty & \cdots & a^{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \infty & \infty & \infty & \cdots & \infty \end{bmatrix} \quad (۲۳-۴)$$

ابتدا، تمامی شباهت‌ها بین مدل‌ها محاسبه می‌شود و سپس از ماتریس ایجاد شده، کمترین مقدار شباهت انتخاب می‌شود. شماره سطر و ستون متناظر با این مقدار نشان می‌دهد که این دو مدل باید با یکدیگر تعویض شوند. پس از انجام این تعویض، تمامی مقدارهای مربوط به سطر و ستون متناظر با این دو مدل باید به بی‌نهایت تغییر داده شوند تا در مراحل بعدی مجدد انتخاب نشوند. همچنین باید دقت کرد که هر دو سطر و ستون مرتبط با این دو مدل باید به بی‌نهایت تغییر داده شوند.

برای درک بهتر، فرض کنید کمترین مقدار شباهت در سطر سوم و ستون ششم ماتریس قرار دارد. در این حالت، مدل‌های سوم و ششم باید با یکدیگر تعویض شوند. پس از این تعویض، لازم است که تمامی مقادیر در سطرهای سوم و ششم و همچنین ستون‌های سوم و ششم به بی‌نهایت تغییر کنند تا این دو مدل دیگر مورد بررسی قرار نگیرند. به این ترتیب، در دور بعدی، کوچک‌ترین مقدار شباهت از ماتریس انتخاب می‌شود و چون مقادیر مربوط به مدل‌های سوم و ششم به بی‌نهایت تغییر کرده‌اند، دیگر در این مرحله حضور نخواهند داشت و انتخاب نمی‌شوند. شبه کد کامل این روش در الگوریتم ۴-۳ ارائه شده است. علاوه بر این، نمادهای مختص به این الگوریتم در جدول ۴-۳ توضیح داده شده‌اند.

۴-۹-۲-۱ تعداد اجرای تابع ModelSimilarity

در این الگوریتم، تعداد دفعات اجرای تابع ModelSimilarity به تعداد کل جابجایی‌ها بین کاربران نهایی وابسته نیست. همان‌طور که در الگوریتم ۴-۳ مشاهده می‌شود، ماتریس شباهت تنها یک بار محاسبه خواهد شد. با

[1] Minimum Similarity Swapping
[2] Triangular Matrix

الگوریتم ۴-۳: تعویض حداقل شباهت (Minimum Similarity Swapping)

```
Function MinSimilaritySwapping():
    Initialize SimArray;
    L = length of U_t;
    for each row from 0 to L do
        for each col from (row + 1) to L do
            Sim = ModelSimilarity(w^row, w^col);
            Append Sim to SimArray;
        end
    end
    NS = (L // 2) * SP;
    for NS times do
        MI = index of minimum value in SimArray;
        row, col = find row and col number, based on MI;
        Set all values of SimArray[row, col] to ∞;
        FedSwap(w^RI, w^CI);
    end
end
```

جدول ۴-۳: نمادهای مختص الگوریتم تعویض حداقل شباهت

متغیر	توضیحات
$SimArray$	آرایه مشابهت
L	طول مجموعه‌ای از مشتریان در گام t
$MI\ (MinIndex)$	شاخص مقدار کمینه در آرایه مشابهت
row	شماره سطر بر اساس دید ماتریس مشابهت
col	شماره ستون بر اساس دید ماتریس مشابهت

توجه به ساختار ماتریس بالا مثلثی در رابطه ۴-۲۳ و با فرض اینکه تعداد مدل‌های کاربران برابر n در نظر گرفته شود، مقدار L در الگوریتم نیز برابر n خواهد شد. در نتیجه، تعداد دفعات اجرای تابع ModelSimilarity در الگوریتم تعویض حداقل شباهت، برابر با $\frac{n(n-1)}{2}$ خواهد بود.

۴-۹-۲-۲ مرتبه زمانی

همانطور که در بخش قبل بررسی شد، تعداد دفعات اجرای تابع ModelSimilarity به تعداد جابجایی مدل‌ها بین کاربران وابسته نیست. در حقیقت، حلقه اصلی این الگوریتم، حلقه دوم است که $n/2$ بار اجرا می‌شود. این حلقه شامل پیدا کردن عنصر کمینه در ماتریس مشابهت است.

پیدا کردن عنصر کمینه در ماتریس مشابهت، خود از مرتبه زمانی $O(n^2)$ است. بنابراین، وقتی که حلقه اصلی $n/2$ بار اجرا شود و هر بار نیاز به پیدا کردن عنصر کمینه در ماتریس مشابهت داشته باشد، مرتبه زمانی کل الگوریتم برابر $O(n^2) \times O(n/2)$ خواهد بود که به $O(n^3)$ ساده‌سازی می‌شود.

۴-۹-۲-۳ علت پیاده‌سازی الگوریتم با استفاده از آرایه مشابهت

در بخش قبل، مشخص شد که مرتبه زمانی این الگوریتم برابر با $O(n^3)$ است. این امر به دلیل وجود حلقه دوم در کد است که شامل محاسبه مقدار کمینه در ماتریس مشابهت می‌شود. ماتریس مشابهت، در حالت عادی، دارای n^2 عنصر است.

با بررسی دقیق‌تر الگوریتم و تعداد اجراهای حلقه دوم، در صورتی که از ماتریس مشابهت در پیاده‌سازی استفاده شود، از نظر زمان اجرا به رابطه زیر خواهیم رسید:

$$\begin{aligned}\left(\frac{n}{2}\right) \times (n^2+4n) &= \left(\frac{n^3}{2}\right) + 2n^2 \\ &\xlongequal{\times 4} 2n^3 + 8n^2\end{aligned} \qquad (۲۴\text{-}۴)$$

در این رابطه، تعداد اجراهای حلقه دوم برابر $n/2$، پیداکردن مقدار کمینه در ماتریس مشابهت برابر با n^2 و انتساب مقدار بی‌نهایت برای دو سطر و ستون ماتریس مشابهت برابر با $4n$ است.

با توجه به رابطه ۴-۲۳، بیش از نصف ماتریس، شامل مقادیر بی‌نهایت می‌باشد. بنابراین، با پیاده‌سازی ماتریس به صورت یک آرایه یک‌بعدی و تنها ذخیره‌سازی مقادیر بالا مثلثی، می‌توان با انجام چند عملیات ساده ریاضی به مقدار سطر و ستون مورد نظر در ماتریس مشابهت دست یافت.

در این پیاده‌سازی، آرایه یک‌بعدی جدید دارای $\frac{n(n-1)}{2}$ عنصر خواهد بود. اگر حلقه دوم الگوریتم، مجدد بررسی شود، زمان اجرای آن به صورت زیر خواهد بود:

$$\begin{aligned}\left(\frac{n}{2}\right) \times \left(\frac{n(n-1)}{2}+4n\right) &= \left(\frac{n^2(n-1)}{4}\right) + 2n^2 \\ &= \left(\frac{n^3}{4} - \frac{n^2}{4}\right) + 2n^2 \\ &\xlongequal{\times 4} n^3 - n^2 + 8n^2 \\ &= n^3 + 7n^2\end{aligned} \qquad (۲۵\text{-}۴)$$

در این عبارت، تعداد اجراهای حلقه دوم برابر $n/2$، پیداکردن مقدار کمینه در آرایه برابر $\frac{n(n-1)}{2}$ و در نهایت انتساب مقدار بی‌نهایت در آرایه مربوطه برابر با $4n$ خواهد بود.

همان‌طور که مشاهده می‌شود، این پیاده‌سازی تقریبا سرعت اجرای الگوریتم را دو برابر می‌کند. اگرچه از نظر مرتبه زمانی بهبودی حاصل نشد، اما افزایش سرعت اجرا به میزان دو برابر، بهبود قابل توجهی در روند آموزش محسوب می‌شود. همچنین از نظر حافظه نیز بهبود حاصل شده است، زیرا در صورت استفاده از ماتریس مشابهت نیاز به ذخیره‌سازی n^2 عنصر است، در حالی که با به‌کارگیری آرایه مشابهت تنها $\frac{n(n-1)}{2}$ عنصر ذخیره خواهد شد. بنابراین، استفاده از ساختار آرایه یک بعدی می‌تواند بسیار مفید بوده و به کارایی الگوریتم کمک کند.

فصل پنجم

پیاده‌سازی و بررسی نتایج

۵-۱ مقدمه

تست

۵-۲ انواع مجموعه داده

در این پژوهش از انواع مجموعه داده‌ها برای بررسی و ارزیابی روش مورد نظر استفاده شده است. این مجموعه داده‌ها هر یک ویژگی‌های منحصر به فردی دارند که به تحلیل‌های دقیق‌تر و جامع‌تر کمک می‌کنند. در ادامه، هر یک از این مجموعه داده‌ها به طور مفصل معرفی و توضیح داده خواهد شد تا اهمیت و کاربردهای آن‌ها مشخص گردد.

۵-۲-۱ مجموعه داده MNIST[1]

مجموعه داده MNIST یکی از مشهورترین و پر استفاده‌ترین مجموعه داده‌ها در زمینه یادگیری ماشین است. این مجموعه شامل تصاویر دست‌نویس از اعداد ۰ تا ۹ می‌باشد و به طور گسترده‌ای برای آموزش و ارزیابی مدل‌های مختلف یادگیری ماشین به کار گرفته می‌شود. مجموعه داده MNIST در دهه ۱۹۹۰ توسط یان لکون[2]، کورینا

[1] Modified National Institute of Standards and Technology
[2] Yann LeCun

کورتس[1] و کریستوفر برجس[2] ایجاد شد. هدف اصلی این مجموعه داده، فراهم کردن یک مجموعه استاندارد برای ارزیابی الگوریتم‌های یادگیری ماشین و بینایی کامپیوتر بود.

مجموعه داده MNIST شامل ۷۰,۰۰۰ تصویر از ارقام دست‌نویس است که به دو بخش شامل مجموعه آموزش با ۶۰,۰۰۰ تصویر و مجموعه تست با ۱۰,۰۰۰ تصویر تقسیم می‌شود. هر تصویر دارای ابعاد (۲۸×۲۸) پیکسل است که به صورت خاکستری[3] ذخیره شده‌اند و هر پیکسل دارای مقداری بین ۰ (سیاه) تا ۲۵۵ (سفید) است. همچنین تمامی تصاویر با یک برچسب عددی بین ۰ تا ۹ همراه هستند که نمایانگر رقم موجود در تصویر می‌باشد [۴۵]. چند نمونه از اعضای این مجموعه داده را می‌توانید در شکل ۵-۱ مشاهده کنید.

داده‌ها معمولاً در قالب دو فایل باینری شامل یکی برای تصاویر و دیگری برای برچسب‌ها ذخیره می‌شوند. هر تصویر به صورت یک بردار از اعداد بین ۰ تا ۲۵۵ با طول ۷۸۴ (۲۸×۲۸) ذخیره می‌شود. به دلیل یکنواختی تصاویر و اندازه کوچک آن‌ها، نیاز به پیش‌پردازش پیچیده‌ای ندارند. یکی از مراحل پیش‌پردازش شامل نرمال‌سازی یا همان تبدیل مقادیر پیکسل‌ها به مقادیر بین ۰ و ۱ می‌باشد.

مجموعه داده MNIST به عنوان یک نقطه شروع استاندارد برای آموزش و ارزیابی مدل‌های مختلف یادگیری عمیق و شبکه‌های عصبی استفاده می‌شود. محققان اغلب از MNIST برای مقایسه کارایی الگوریتم‌های جدید با الگوریتم‌های موجود استفاده می‌کنند. این مجموعه شامل نمونه‌های متنوعی از ارقام دست‌نویس از افراد مختلف است که موجب می‌شود به عنوان یک معیار استاندارد برای مقایسه مدل‌ها و الگوریتم‌ها مورد استفاده قرار گیرد.

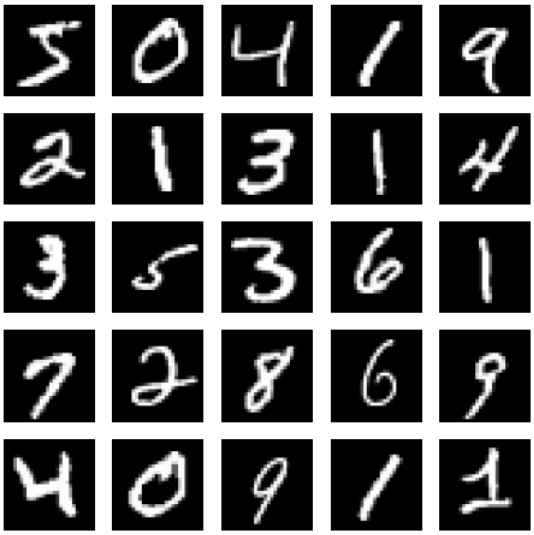
مجموعه داده MNIST دارای مزایای زیادی از جمله سادگی، در دسترس بودن، استاندارد بودن و پراکندگی داده‌ها می‌باشد. با این حال، این مجموعه داده دارای معایبی نیز هست. به عنوان مثال، برای مسائل پیچیده‌تر و واقعی‌تر ممکن است MNIST خیلی ساده باشد و نتواند چالش‌های واقعی را نشان دهد. همچنین، این مجموعه داده شامل تنها اعداد ۰ تا ۹ است و برای سایر کاربردهای دسته‌بندی تصویر ممکن است کافی نباشد.

کاربردهای عملی این مجموعه داده شامل آموزش شبکه‌های عصبی متفاوت برای بهبود دقت دسته‌بندی، تست و ارزیابی مدل‌های مختلف یادگیری عمیق و الگوریتم‌های بهینه‌سازی است. بسیاری از مدل‌ها و الگوریتم‌های پیشرفته امروزی با استفاده از مجموعه داده MNIST توسعه و ارزیابی شده‌اند.

به طور کلی، مجموعه داده MNIST با توجه به دلایل ذکر شده، یکی از مهم‌ترین و پراستفاده‌ترین مجموعه داده‌ها در زمینه یادگیری ماشین و بینایی کامپیوتر است. این مجموعه به محققان و دانشجویان کمک می‌کند تا مفاهیم پایه‌ای یادگیری ماشین را به خوبی درک کرده و الگوریتم‌های جدید را ارزیابی کنند.

---

[1] Corinna Cortes
[2] Christopher Burges
[3] Grayscale



شکل ۵-۱: چند نمونه از اعضای مجموعه داده MNIST [۴۵].

۵-۲-۲ مجموعه داده CIFAR-10[1]

مجموعه داده CIFAR-10 یکی از معروف‌ترین و پرکاربردترین مجموعه داده‌های مورد استفاده در حوزه یادگیری ماشین و بینایی کامپیوتر است. این مجموعه داده توسط گروهی به سرپرستی الکس کریژفسکی[2] و جفری هینتون[3] در دانشگاه تورنتو گردآوری شده و برای ارزیابی و آزمایش مدل‌های یادگیری عمیق به کار می‌رود.

مجموعه داده CIFAR-10 شامل ۶۰,۰۰۰ تصویر رنگی با اندازه ۳۲×۳۲ پیکسل است که به ۱۰ کلاس مختلف تقسیم شده‌اند. هر کلاس شامل ۶,۰۰۰ تصویر است که به صورت مساوی بین مجموعه‌های آموزشی و آزمایشی توزیع شده‌اند. این کلاس‌ها شامل مواردی مانند هواپیما، اتومبیل، پرنده، گربه، گوزن، سگ، قورباغه، اسب، کِشتی و کامیون هستند. هر یک از این کلاس‌ها دارای تصاویری است که تنوع بالایی از زوایا، پس‌زمینه‌ها و شرایط نوری مختلف را شامل می‌شود [۴۶].

یکی از ویژگی‌های مهم مجموعه داده CIFAR-10 تنوع بالای تصاویر در هر کلاس است. این تنوع باعث می‌شود که مدل‌های یادگیری عمیق نیاز به توانایی تعمیم‌دهی بالا برای تشخیص صحیح کلاس‌ها داشته باشند. این مجموعه داده برای آموزش و ارزیابی مدل‌های مختلفی مورد استفاده قرار می‌گیرد و بسیاری از پژوهش‌ها و مقالات علمی از آن به عنوان مبنای مقایسه عملکرد مدل‌ها استفاده کرده‌اند.

مجموعه داده CIFAR-10 به دو بخش آموزشی و آزمایشی تقسیم شده است. بخش آموزشی شامل ۵۰,۰۰۰ تصویر و بخش آزمایشی شامل ۱۰,۰۰۰ تصویر است. این تقسیم‌بندی، استانداردی برای ارزیابی مدل‌ها فراهم می‌کند، به طوری که مدل‌ها می‌توانند بر روی مجموعه آموزشی، آموزش دیده و سپس بر روی مجموعه آزمایشی

[1] Canadian Institute For Advanced Research
[2] Alex Krizhevsky
[3] Geoffrey Hinton

ارزیابی شوند. این روش به محققان امکان می‌دهد تا عملکرد مدل‌ها را به صورت عینی و قابل تکرار مقایسه کنند.

به دلیل اندازه کوچک تصاویر (۳۲×۳۲ پیکسل)، پردازش و آموزش مدل‌ها بر روی CIFAR-10 نسبتاً سریع و کم هزینه است. این ویژگی باعث شده تا مجموعه داده CIFAR-10 برای آزمایش مدل‌ها بسیار مناسب باشد. بسیاری از ابزارها و چارچوب‌های[1] یادگیری ماشین مانند PyTorch و TensorFlow شامل توابع و ابزارهای آماده برای بارگذاری و استفاده از این مجموعه داده هستند که این امر نیز به سهولت استفاده از آن کمک می‌کند.

در نهایت، مجموعه داده CIFAR-10 با ارائه تصاویری متنوع و چالش‌برانگیز در کلاس‌های مختلف، ابزاری قدرتمند برای آموزش و ارزیابی مدل‌های یادگیری عمیق فراهم می‌کند. این مجموعه داده نه تنها در پژوهش‌های دانشگاهی بلکه در صنعت نیز به عنوان معیاری برای ارزیابی پیشرفت‌ها در حوزه بینایی کامپیوتر استفاده می‌شود.

۳-۲-۵ مجموعه داده CINIC-10[2]

مجموعه داده CINIC-10 یک مجموعه داده تصویری گسترده و متنوع است که برای ارزیابی عملکرد مدل‌های یادگیری ماشین به ویژه در زمینه‌های مرتبط با طبقه‌بندی تصاویر مورد استفاده قرار می‌گیرد. این مجموعه داده ترکیبی از تصاویر موجود در مجموعه داده‌های معروف CIFAR-10 و ImageNet است. این ترکیب به منظور ایجاد مجموعه‌ای گسترده‌تر و متنوع‌تر از تصاویر انجام شده است که می‌تواند به ارزیابی دقیق‌تر و واقع‌گرایانه‌تر مدل‌ها کمک کند.

مجموعه داده CINIC-10 شامل ۲۷۰,۰۰۰ تصویر است که در ۱۰ کلاس مختلف دسته‌بندی شده‌اند. هر کلاس شامل ۲۷,۰۰۰ تصویر است که به دو بخش آموزشی و آزمایشی تقسیم شده‌اند. بخش آموزشی شامل ۱۸۰,۰۰۰ تصویر و بخش آزمایشی شامل ۹۰,۰۰۰ تصویر است. این تقسیم‌بندی منظم به محققان و مهندسان یادگیری ماشین این امکان را می‌دهد که به راحتی مدل‌های خود را آموزش داده، اعتبارسنجی و آزمایش کنند.

تصاویر موجود در CINIC-10 دارای ابعاد ۳۲×۳۲ پیکسل هستند که مشابه ابعاد تصاویر موجود در مجموعه داده CIFAR-10 است. این ویژگی باعث می‌شود که مدل‌های از پیش آموزش دیده بر روی CIFAR-10 بتوانند به راحتی بر روی این مجموعه داده نیز مورد استفاده قرار گیرند و ارزیابی شوند. با این حال، تنوع بیشتر تصاویر در CINIC-10 نسبت به CIFAR-10 به دلیل ترکیب تصاویر از ImageNet، چالشی جدی‌تر برای مدل‌های یادگیری ماشین فراهم می‌کند [۴۷].

یکی از اهداف اصلی ایجاد CINIC-10، افزایش تنوع و پیچیدگی تصاویر مورد استفاده برای آموزش و

[1] Frameworks
[2] CIFAR-10 and ImageNet Combined

ارزیابی مدل‌ها بود. این مجموعه داده شامل تصاویری از دنیای واقعی است که در شرایط نوری مختلف و با پس‌زمینه‌های متنوع گرفته شده‌اند. این ویژگی به مدل‌ها کمک می‌کند تا به جای اینکه تنها بر روی مجموعه‌ای محدود از تصاویر آموزش ببینند، توانایی تعمیم‌دهی خود را به تصاویر جدید و غیرمنتظره نیز افزایش دهند.

در نهایت، CINIC-10 با هدف ارتقای استانداردهای ارزیابی مدل‌های یادگیری عمیق و بهبود عملکرد آن‌ها در مواجهه با داده‌های واقعی و متنوع ایجاد شده است. این مجموعه داده به محققان این امکان را می‌دهد که مدل‌های خود را در شرایط نزدیک به دنیای واقعی آزمایش کرده و نقاط ضعف و قوت آن‌ها را بهتر شناسایی کنند. به همین دلیل، CINIC-10 به عنوان یک ابزار ارزشمند در جامعه یادگیری ماشین شناخته می‌شود و به طور گسترده‌ای مورد استفاده قرار می‌گیرد.

۵-۲-۴ مجموعه داده FEMNIST[1]

مجموعه داده FEMNIST یک مجموعه داده توسعه‌یافته از مجموعه مشهور MNIST است که برای کاربردهای یادگیری فدرال طراحی شده است. این مجموعه داده از تصاویر دست‌نوشته به وجود آمده است که شامل اعداد و حروف الفبای انگلیسی می‌شود. برخلاف مجموعه داده MNIST که تنها شامل اعداد دست‌نوشته از صفر تا نه است، مجموعه داده FEMNIST شامل حروف بزرگ و کوچک الفبای انگلیسی نیز می‌باشد. این ویژگی باعث می‌شود که FEMNIST نسبت به MNIST تنوع بیشتری داشته باشد و برای آزمایش مدل‌های پیچیده، مناسب‌تر باشد [۴۸].

یکی از ویژگی‌های برجسته مجموعه داده FEMNIST، نحوه سازماندهی داده‌ها است. این مجموعه داده بر اساس کاربران مختلف تقسیم‌بندی شده است، به طوری که هر کاربر دارای مجموعه‌ای از داده‌های دست‌نوشته خود است. این سازماندهی امکان آزمایش و ارزیابی روش‌های یادگیری فدرال را فراهم می‌کند، زیرا در یادگیری فدرال داده‌ها به صورت محلی بر روی دستگاه‌های کاربران نگه‌داری می‌شوند و مدل‌ها بر روی این داده‌ها آموزش می‌بینند. این ویژگی به محققان اجازه می‌دهد تا سناریوهای واقعی‌تری از یادگیری فدرال را شبیه‌سازی و بررسی کنند.

تصاویر در مجموعه داده FEMNIST به صورت سیاه و سفید و با اندازه ۲۸×۲۸ پیکسل هستند. هر تصویر نمایانگر یک کاراکتر دست‌نوشته است. این تصاویر از مجموعه داده NIST استخراج شده‌اند و به صورت مناسبی برای کاربردهای یادگیری فدرال سازماندهی شده‌اند. در حقیقت این تصاویر شامل نویسه‌های مختلف از کاربران مختلف است که تنوع در سبک نوشتن و کیفیت دست‌نوشته‌ها را افزایش می‌دهد.

به طور کلی، مجموعه داده FEMNIST یک ابزار قدرتمند برای تحقیقات در زمینه یادگیری فدرال است.

[1] Federated Extended MNIST

با ارائه تنوع بالای داده‌ها و سازماندهی مناسب برای سناریوهای یادگیری فدرال، این مجموعه داده به محققان کمک می‌کند تا روش‌ها و الگوریتم‌های جدید را در محیط‌های واقعی‌تر آزمایش کنند. این ویژگی‌ها باعث شده تا FEMNIST به عنوان یکی از مجموعه داده‌های مرجع در این حوزه شناخته شود و در بسیاری از تحقیقات علمی و صنعتی مورد استفاده قرار گیرد.

۳-۵ پیاده‌سازی مدل‌های شبکه‌عصبی

در این بخش، دو ساختار و مدل اصلی شبکه‌های عصبی که طراحی شده‌اند، مورد بررسی قرار خواهند گرفت. این دو ساختار اصلی شامل شبکه عصبی پرسپترون چندلایه (MLP)[۱] و شبکه عصبی پیچشی (CNN)[۲] می‌باشند. در ادامه، هر یک از این مدل‌های طراحی شده به صورت مفصل توضیح داده خواهند شد. این توضیحات به منظور فراهم آوردن درکی جامع از نحوه عملکرد هر مدل و نقش آن‌ها در بررسی نتایج و مقایسه روش‌ها در پژوهش انجام خواهد شد. استفاده از این مدل‌ها به تحلیل و ارزیابی دقیق‌تر نتایج کمک کرده و مقایسه‌ای جامع از روش‌های مختلف را ممکن می‌سازد.

۱-۳-۵ مدل MLP

در شبکه عصبی چندلایه تعریف شده، ابتدا لایه‌های شبکه عصبی در قالب یک ساختار ترتیبی[۳] با استفاده از یک دیکشنری مرتب[۴] تعریف می‌شوند. این ساختار ترتیب‌دار باعث می‌شود که لایه‌ها به صورت متوالی اجرا شوند و خروجی هر لایه به عنوان ورودی به لایه بعدی منتقل شود.

اولین لایه، یک لایه کاملاً متصل است که تعداد نورون‌های ورودی آن برابر با تعداد ویژگی‌های ورودی مدل به صورت تخت‌شده[۵] و تعداد نورون‌های خروجی آن ۲۵۶ است. این لایه تمام اتصالات ممکن بین نورون‌های ورودی و خروجی را دارد. پس از این لایه، یک تابع فعال‌سازی ReLU قرار دارد که وظیفه آن این است که تمامی مقادیر منفی خروجی را به صفر تبدیل کند و مقادیر مثبت را بدون تغییر نگه دارد.

لایه دوم، یک لایه کاملاً متصل دیگر است که ۲۵۶ نورون ورودی و ۱۲۸ نورون خروجی دارد. پس از این لایه نیز یک تابع فعال‌سازی ReLU قرار دارد که مشابه تابع فعال‌ساز قبلی عمل می‌کند. سومین لایه نیز دقیقا مشابه لایه دوم است با این تفاوت که ۱۲۸ نورون به عنوان ورودی و ۶۴ نورون به عنوان خروجی دارد و در ادامه آن هم تابع فعال‌سازی ReLU وجود دارد.

---
[1] Multilayer Perceptron
[2] Convolutional Neural Network
[3] Sequential
[4] OrderedDict
[5] Flatten

چهارمین و آخرین لایه، یک لایه کاملاً متصل است که ۶۴ نورون ورودی و تعداد نورون‌های خروجی آن برابر با تعداد کلاس‌های موجود در مسئله طبقه‌بندی است. این لایه خروجی‌های نهایی شبکه را تولید می‌کند که نشان‌دهنده میزان تعلق هر ورودی به هر یک از کلاس‌ها است.

در نهایت، یک لایه Softmax اضافه شده است که وظیفه آن تبدیل خروجی‌های نهایی شبکه به توزیع احتمالاتی است. این لایه کمک می‌کند تا بتوان احتمال تعلق هر ورودی به هر کلاس را به صورت عددی بین صفر و یک بدست آورد که جمع کل این احتمالات برای همه کلاس‌ها برابر با یک خواهد بود. این توزیع احتمالاتی برای انجام پیش‌بینی‌های نهایی مورد استفاده قرار می‌گیرد. در پایان می‌توانید ساختار این مدل را در شکل ۵-۲ مشاهده نمایید.

۵-۳-۲ مدل CNN

در شبکه عصبی پیچشی، ابتدا یک بلوک ترتیبی شامل لایه‌های مختلف به کمک دیکشنری مرتب تعریف شده است. این لایه‌ها به ترتیب وظایف مختلفی در استخراج ویژگی‌ها و انجام طبقه‌بندی نهایی دارند.

اولین لایه، یک لایه پیچشی تعریف شده است که تعداد کانال‌های ورودی تصویر را به ۳۲ کانال خروجی تبدیل می‌کند. این لایه از یک کرنل پیچشی با اندازه ۳×۳ استفاده می‌کند. این لایه وظیفه دارد تا ویژگی‌های ابتدایی تصویر ورودی را استخراج کند. پس از این لایه، یک تابع فعال‌سازی ReLU قرار دارد که مقادیر منفی را به صفر تبدیل کرده و مقادیر مثبت را بدون تغییر نگه می‌دارد که این کار باعث ایجاد غیرخطی شدن شبکه می‌شود.

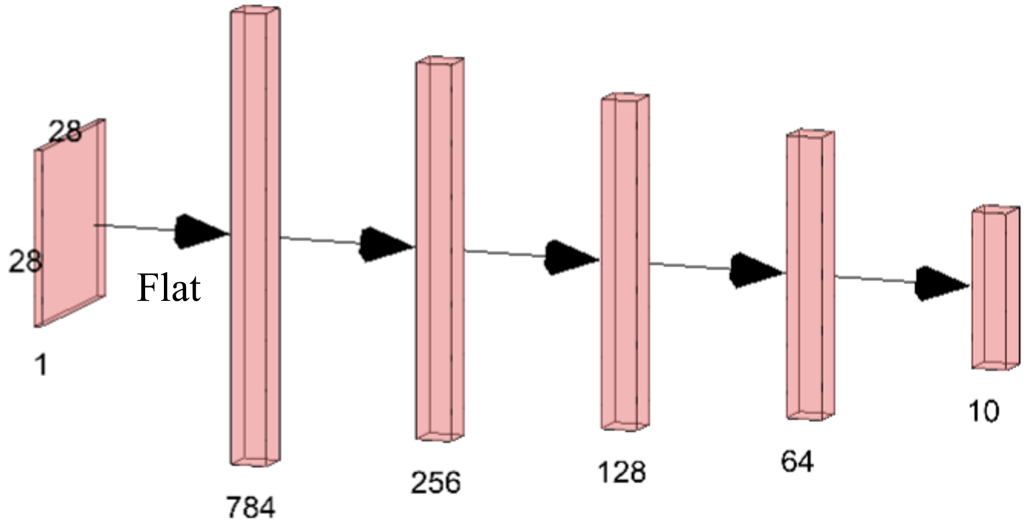
سپس یک لایه تجمیع حداکثر[1] قرار دارد که اندازه فضای ویژگی‌های خروجی را کاهش می‌دهد و به کم کردن تعداد پارامترها و افزایش کارایی مدل کمک می‌کند. این لایه با انتخاب حداکثر مقدار در هر ناحیه کوچک ۲×۲، منجر به کاهش ابعاد تصاویر خواهد شد.

در ادامه، یک لایه پیچشی دیگر قرار دارد که تعداد کانال‌های خروجی را به ۶۴ کانال افزایش می‌دهد. این لایه نیز از یک کرنل پیچشی با اندازه ۳×۳ استفاده می‌کند و وظیفه استخراج ویژگی‌های پیچیده‌تر را بر عهده دارد. پس از این لایه نیز یک تابع فعال‌سازی ReLU قرار دارد که مشابه قبل عمل می‌کند.

سپس یک لایه تجمیع حداکثر دیگر قرار دارد که اندازه فضای ویژگی‌های خروجی را مجدداً کاهش می‌دهد. این لایه نیز با انتخاب حداکثر مقدار در هر ناحیه کوچک ۲×۲، به کاهش ابعاد تصاویر کمک می‌کند.

پس از این لایه‌ها، یک لایه تخت‌کننده قرار دارد که ویژگی‌های چند بعدی خروجی را به یک بردار یک بعدی تبدیل می‌کند. این کار برای آماده‌سازی داده‌ها جهت ورود به لایه‌های کاملاً متصل انجام می‌شود.

[1] Max Pooling



شکل ۵-۲: ساختار مدل MLP

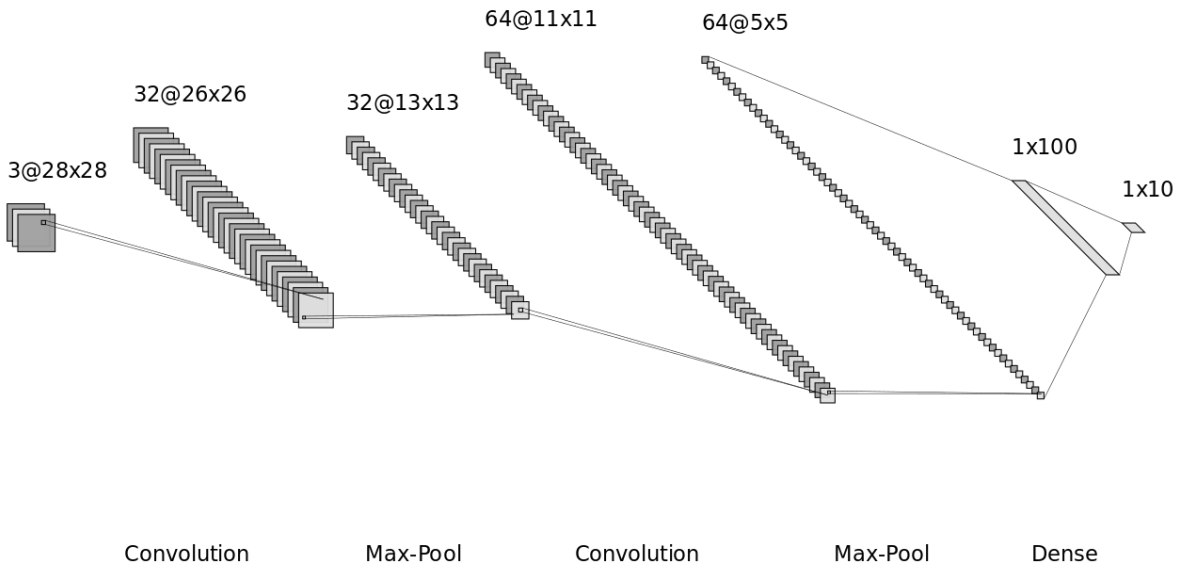
در مرحله بعد، یک لایه کاملاً متصل قرار دارد که بردار ویژگی‌ها را به یک بردار با ۱۰۰ نورون تبدیل می‌کند. این لایه تمام اتصالات ممکن بین نورون‌های ورودی و خروجی را دارد. پس از این لایه، یک تابع فعال‌سازی ReLU وجود دارد که مشابه توابع فعال‌سازی قبلی عمل می‌کند و غیرخطی‌بودن را به شبکه اضافه می‌کند.

در پایان، یک لایه کاملاً متصل دیگر قرار دارد که بردار ویژگی‌ها را به یک بردار جدید با تعداد نورون‌هایی برابر با تعداد کلاس‌ها تبدیل می‌کند. این لایه، خروجی نهایی شبکه را تولید می‌کند که نشان می‌دهد هر ورودی به چه میزان به هر یک از کلاس‌ها تعلق دارد.

در انتها، یک لایه Softmax قرار داده شده که خروجی‌های نهایی شبکه را به توزیع احتمالاتی تبدیل می‌کند. این لایه باعث می‌شود که احتمال تعلق هر ورودی به هر کلاس به صورت عددی بین صفر و یک محاسبه شود، به طوری که مجموع این احتمالات برای همه کلاس‌ها برابر با یک باشد. این توزیع احتمالاتی در انتها برای انجام پیش‌بینی‌های نهایی به کار می‌رود. در پایان جهت درک بهتر می‌توانید ساختار این مدل را در شکل ۵-۳ مشاهده نمایید.

۵-۴ بررسی نتایج

تست



شکل ۵-۳: ساختار مدل CNN

مراجع

[1] Elbir, Ahmet M, Coleri, Sinem, Papazafeiropoulos, Anastasios K, Kourtessis, Pandelis, and Chatzinotas, Symeon. A family of hybrid federated and centralizedlearning architectures in machine learning. *IEEE Transactions on Cognitive Communications and Networking*, 2022.

[2] Zhou, Zhi, Chen, Xu, Li, En, Zeng, Liekang, Luo, Ke, and Zhang, Junshan. Edge intelligence: Paving the last mile of artificial intelligence with edge computing. *Proceedings of the IEEE*, 107(8):1738–1762, 2019.

[3] Ma, Xiaodong, Zhu, Jia, Lin, Zhihao, Chen, Shanxuan, and Qin, Yangjie. A state-of-the-art survey on solving non-iid data in federated learning. *Future Generation Computer Systems*, 135:244–258, 2022.

[4] Smith, Virginia, Chiang, Chao-Kai, Sanjabi, Maziar, and Talwalkar, Ameet S. Federated multi-task learning. *Advances in neural information processing systems*, 30, 2017.

[5] McMahan, Brendan, Ramage Daniel. Federated learning: Collaborative machine learning without centralized training data. https://www.omron.com/global/en/technology/information/dcx, 6 Apr 2017. [Accessed: 18 Apr 2024].

[6] Li, Tian, Sahu, Anit Kumar, Talwalkar, Ameet, and Smith, Virginia. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3):50–60, 2020.

[7] Talaei, Mahtab. Algorithm development and performance analysis for adaptive differential privacy in federated learning, 21 Aug 2022.

[8] Rieke, Nicola. What is federated learning? https://blogs.nvidia.com/blog/what-is-federated-learning/, 13 Oct 2019. [Accessed: 10 Apr 2024].

[9] McMahan, Brendan, Moore, Eider, Ramage, Daniel, Hampson, Seth, and y Arcas, Blaise Aguera. Communication-efficient learning of deep networks from decentralized data. in *Artificial intelligence and statistics*, pp. 1273–1282. PMLR, 2017.

[10] Wang, Hongyi, Sievert, Scott, Liu, Shengchao, Charles, Zachary, Papailiopoulos, Dimitris, and Wright, Stephen. Atomo: Communication-efficient learning via atomic sparsification. *Advances in neural information processing systems*, 31, 2018.

[11] Konečnỳ, Jakub, McMahan, H Brendan, Yu, Felix X, Richtárik, Peter, Suresh, Ananda Theertha, and Bacon, Dave. Federated learning: Strategies for improving communication efficiency. *arXiv preprint arXiv:1610.05492*, 2016.

[12] Fang, Chen, Guo, Yuanbo, Hu, Yongjin, Ma, Bowen, Feng, Li, and Yin, Anqi. Privacy-preserving and communication-efficient federated learning in internet of things. *Computers & Security*, 103:102199, 2021.

[13] Konečnỳ, Jakub, McMahan, Brendan, and Ramage, Daniel. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.

[14] Hasan, Jahid. Security and privacy issues of federated learning. *arXiv preprint arXiv:2307.12181*, 2023.

[15] Yin, Xuefei, Zhu, Yanming, and Hu, Jiankun. A comprehensive survey of privacy-preserving federated learning: A taxonomy, review, and future directions. *ACM Computing Surveys (CSUR)*, 54(6):1–36, 2021.

[16] Ioffe, Sergey and Szegedy, Christian. Batch normalization: Accelerating deep network training by reducing internal covariate shift. in *International conference on machine learning*, pp. 448–456. pmlr, 2015.

[17] Li, Tian, Sahu, Anit Kumar, Zaheer, Manzil, Sanjabi, Maziar, Talwalkar, Ameet, and Smith, Virginia. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020.

[18] Zhao, Yue, Li, Meng, Lai, Liangzhen, Suda, Naveen, Civin, Damon, and Chandra, Vikas. Federated learning with non-iid data. *arXiv preprint arXiv:1806.00582*, 2018.

[19] Collins, Liam, Hassani, Hamed, Mokhtari, Aryan, and Shakkottai, Sanjay. Exploiting shared representations for personalized federated learning. in *International conference on machine learning*, pp. 2089–2099. PMLR, 2021.

[20] Jeong, Eunjeong, Oh, Seungeun, Kim, Hyesung, Park, Jihong, Bennis, Mehdi, and Kim, Seong-Lyun. Communication-efficient on-device machine learning: Federated distillation and augmentation under non-iid private data. *arXiv preprint arXiv:1811.11479*, 2018.

[21] Taïk, Afaf, Moudoud, Hajar, and Cherkaoui, Soumaya. Data-quality based scheduling for federated edge learning. in *2021 IEEE 46th Conference on Local Computer Networks (LCN)*, pp. 17–23. IEEE, 2021.

[22] Zeng, Yan, Wang, Xin, Yuan, Junfeng, Zhang, Jilin, and Wan, Jian. Local epochs inefficiency caused by device heterogeneity in federated learning. *Wireless Communications & Mobile Computing*, 2022.

[23] Sannara, EK, Portet, François, Lalanda, Philippe, and German, VEGA. A federated learning aggregation algorithm for pervasive computing: Evaluation and comparison. in *2021 IEEE International Conference on Pervasive Computing and Communications (PerCom)*, pp. 1–10. IEEE, 2021.

[24] Qin, Yang and Kondo, Masaaki. Mlmg: Multi-local and multi-global model aggregation for federated learning. in *2021 IEEE international conference on pervasive computing and communications workshops and other affiliated events (PerCom Workshops)*, pp. 565–571. IEEE, 2021.

[25] Ma, Qianpiao, Xu, Yang, Xu, Hongli, Jiang, Zhida, Huang, Liusheng, and Huang, He. Fedsa: A semi-asynchronous federated learning mechanism in heterogeneous edge computing. *IEEE Journal on Selected Areas in Communications*, 39(12):3654–3672, 2021.

[26] Li, Li, Duan, Moming, Liu, Duo, Zhang, Yu, Ren, Ao, Chen, Xianzhang, Tan, Yujuan, and Wang, Chengliang. Fedsae: A novel self-adaptive federated learning framework in heterogeneous systems. in *2021 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–10. IEEE, 2021.

[27] Reddi, Sashank, Charles, Zachary, Zaheer, Manzil, Garrett, Zachary, Rush, Keith, Konečnỳ, Jakub, Kumar, Sanjiv, and McMahan, H Brendan. Adaptive federated optimization. *arXiv preprint arXiv:2003.00295*, 2020.

[28] Li, Xiaoli, Liu, Nan, Chen, Chuan, Zheng, Zibin, Li, Huizhong, and Yan, Qiang. Communication-efficient collaborative learning of geo-distributed jointcloud from heterogeneous datasets. in *2020 IEEE international conference on joint cloud computing*, pp. 22–29. IEEE, 2020.

[29] Ghosh, Avishek, Hong, Justin, Yin, Dong, and Ramchandran, Kannan. Robust federated learning in a heterogeneous environment. *arXiv preprint arXiv:1906.06629*, 2019.

[30] Itahara, Sohei, Nishio, Takayuki, Koda, Yusuke, Morikura, Masahiro, and Yamamoto, Koji. Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-iid private data. *IEEE Transactions on Mobile Computing*, 22(1):191–205, 2021.

[31] Chai, Zheng, Ali, Ahsan, Zawad, Syed, Truex, Stacey, Anwar, Ali, Baracaldo, Nathalie, Zhou, Yi, Ludwig, Heiko, Yan, Feng, and Cheng, Yue. Tifl: A tier-based federated learning system. in *Proceedings of the 29th international symposium on high-performance parallel and distributed computing*, pp. 125–136, 2020.

[32] Jiang, Yihan, Konečnỳ, Jakub, Rush, Keith, and Kannan, Sreeram. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.

[33] Zhang, Xinwei, Hong, Mingyi, Dhople, Sairaj, Yin, Wotao, and Liu, Yang. Fedpd: A federated learning framework with adaptivity to non-iid data. *IEEE Transactions on Signal Processing*, 69:6055–6070, 2021.

[34] Corinzia, Luca, Beuret, Ami, and Buhmann, Joachim M. Variational federated multi-task learning. *arXiv preprint arXiv:1906.06268*, 2019.

[35] Shoham, Neta, Avidor, Tomer, Keren, Aviv, Israel, Nadav, Benditkis, Daniel, Mor-Yosef, Liron, and Zeitak, Itai. Overcoming forgetting in federated learning on non-iid data. *arXiv preprint arXiv:1910.07796*, 2019.

[36] Chiu, Te-Chuan, Shih, Yuan-Yao, Pang, Ai-Chun, Wang, Chieh-Sheng, Weng, Wei, and Chou, Chun-Ting. Semisupervised distributed learning with non-iid data for aiot service platform. *IEEE Internet of Things Journal*, 7(10):9266–9277, 2020.

[37] Kornblith, Simon, Norouzi, Mohammad, Lee, Honglak, and Hinton, Geoffrey. Similarity of neural network representations revisited. in *International conference on machine learning*, pp. 3519–3529. PMLR, 2019.

[38] Chen, An Mei, Lu, Haw-minn, and Hecht-Nielsen, Robert. On the geometry of feedforward neural network error surfaces. *Neural computation*, 5(6):910–927, 1993.

[39] Orhan, A Emin and Pitkow, Xaq. Skip connections eliminate singularities. *arXiv preprint arXiv:1701.09175*, 2017.

[40] LeCun, Yann, Kanter, Ido, and Solla, Sara. Second order properties of error surfaces: Learning time and generalization. *Advances in neural information processing systems*, 3, 1990.

[41] Gretton, Arthur, Bousquet, Olivier, Smola, Alex, and Schölkopf, Bernhard. Measuring statistical dependence with hilbert-schmidt norms. in *International conference on algorithmic learning theory*, pp. 63–77. Springer, 2005.

[42] Cortes, Corinna, Mohri, Mehryar, and Rostamizadeh, Afshin. Algorithms for learning kernels based on centered alignment. *The Journal of Machine Learning Research*, 13:795–828, 2012.

[43] Cristianini, Nello, Shawe-Taylor, John, Elisseeff, Andre, and Kandola, Jaz. On kernel-target alignment. *Advances in neural information processing systems*, 14, 2001.

[44] Cui, Tianyu, Kumar, Yogesh, Marttinen, Pekka, and Kaski, Samuel. Deconfounded representation similarity for comparison of neural networks. *Advances in Neural Information Processing Systems*, 35:19138–19151, 2022.

[45] LeCun, Yann, Bottou, Léon, Bengio, Yoshua, and Haffner, Patrick. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[46] Krizhevsky, Alex, Hinton, Geoffrey, et al. Learning multiple layers of features from tiny images. 2009.

[47] Darlow, Luke N, Crowley, Elliot J, Antoniou, Antreas, and Storkey, Amos J. Cinic-10 is not imagenet or cifar-10. *arXiv preprint arXiv:1810.03505*, 2018.

[48] Caldas, Sebastian, Duddu, Sai Meher Karthik, Wu, Peter, Li, Tian, Konečnỳ, Jakub, McMahan, H Brendan, Smith, Virginia, and Talwalkar, Ameet. Leaf: A benchmark for federated settings. *arXiv preprint arXiv:1812.01097*, 2018.